STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

NIKOLAUS BLÜMLEIN

Function-based Cost Estimation for Service Robot Prototypes in Early Design Phases

Universität Stuttgart

Fraunhofer IPA

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 35

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper

Nikolaus Blümlein

Function-based Cost Estimation for Service Robot Prototypes in Early Design Phases

FRAUNHOFER VERLAG

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-00, Telefax 0711 970-1399
info@ipa.fraunhofer.de, www.ipa.fraunhofer.de

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Herausgeber:
Univ.-Prof. Dr.-Ing. Thomas Bauernhansl
Univ.-Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Univ.-Prof. a. D. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper

Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW)
der Universität Stuttgart

Titelbild: © Dominik Kaltenbacher

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
ISSN: 2195-2892
ISBN (Print): 978-3-8396-0749-7

D 93

Zugl.: Stuttgart, Univ., Diss., 2013

Druck: Mediendienstleistungen des Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-2500
Telefax 0711 970-2508
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

GELEITWORT DER HERAUSGEBER

Produktionswissenschaftliche Forschungsfragen entstehen in der Regel im Anwendungszusammenhang, die Produktionsforschung ist also weitgehend erfahrungsbasiert. Der wissenschaftliche Anspruch der „Stuttgarter Beiträge zur Produktionsforschung" liegt unter anderem darin, Dissertation für Dissertation ein übergreifendes ganzheitliches Theoriegebäude der Produktion zu erstellen.

Die Herausgeber dieser Dissertations-Reihe leiten gemeinsam das Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA und jeweils ein Institut der Fakultät für Konstruktions-, Produktions- und Fahrzeugtechnik an der Universität Stuttgart.

Die von ihnen betreuten Dissertationen sind der marktorientierten Nachhaltigkeit verpflichtet, ihr Ansatz ist systemisch und interdisziplinär. Die Autoren bearbeiten anspruchsvolle Forschungsfragen im Spannungsfeld zwischen theoretischen Grundlagen und industrieller Anwendung.

Die „Stuttgarter Beiträge zur Produktionsforschung" ersetzt die Reihen „IPA-IAO Forschung und Praxis" (Hrsg. H.J. Warnecke / H.-J. Bullinger / E. Westkämper / D. Spath) bzw. ISW Forschung und Praxis (Hrsg. G. Stute / G. Pritschow / A. Verl). In den vergangenen Jahrzehnten sind darin über 800 Dissertationen erschienen.

Der Strukturwandel in den Industrien unseres Landes muss auch in der Forschung in einen globalen Zusammenhang gestellt werden. Der reine Fokus auf Erkenntnisgewinn ist zu eindimensional. Die „Stuttgarter Beiträge zur Produktionsforschung" zielen also darauf ab, mittelfristig Lösungen für den Markt anzubieten. Daher konzentrieren sich die Stuttgarter produktionstechnischen Institute auf das Thema ganzheitliche Produktion in den Kernindustrien Deutschlands. Die leitende Forschungsfrage der Arbeiten ist: Wie können wir nachhaltig mit einem hohen Wertschöpfungsanteil in Deutschland für einen globalen Markt produzieren?

Wir wünschen den Autoren, dass ihre „Stuttgarter Beiträge zur Produktionsforschung" in der breiten Fachwelt als substanziell wahrgenommen werden und so die Produktionsforschung weltweit voranbringen.

Alexander Verl Thomas Bauernhansl Engelbert Westkämper

Function-based Cost Estimation for Service Robot Prototypes in Early Design Phases

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von

Dipl.-Inf. Dipl.-Kfm. Nikolaus Blümlein

aus Ludwigshafen am Rhein

Hauptberichter:	Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl
Mitberichter:	Prof. Dr.-Ing. Heinz Wörn
Tag der mündlichen Prüfung:	05.11.2013

Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW)

der Universität Stuttgart

2013

Vorwort des Autors

Die vorliegende Arbeit entstand im Rahmen meiner Arbeit als wissenschaftlicher Mitarbeiter der Abteilung Roboter- und Assistenzsysteme des Fraunhofer-Instituts für Produktionstechnik und Automatisierung (IPA) in Stuttgart-Vaihingen.

An erster Stelle möchte ich Herrn Prof. Dr.-Ing. Dr. h.c. mult. Alexander Verl für die Übernahme des Hauptberichts, seine Betreuung während der Anfertigung der Arbeit und seine wertvollen Ratschläge danken. Mein Dank gilt auch Herrn Prof. Dr.-Ing. Heinz Wörn für die freundliche Übernahme des Mitberichts.

Zahlreiche Kollegen standen mir während der Dissertation mit hilfreichen Ratschlägen und ermutigenden Worten zur Seite und trugen so direkt und indirekt zum Gelingen dieser Arbeit bei. Ich bedanke mich bei Herrn Dipl.-Ing. Martin Hägele M.S., der mich maßgeblich bei der Themenfindung und Informationsrecherche unterstützt hat. Für seine konstruktiven Anmerkungen und kontinuierlichen Zuspruch bin ich Herrn Dipl.-Ing. Kai Pfeiffer zu Dank verpflichtet. Den Kollegen Frau Dr.-Ing. Dipl.-Inf. Birgit Graf, Herrn Dr.-Ing. Georg Arbeiter, Herrn Dipl.-Inf. Winfried Baum, Herrn Dipl.-Ing. Alexander Bubeck, Herrn Dr.-Ing. Ulrich Reiser und Herrn Dipl.-Ing Maik Siee danke ich für ihre stet Bereitschaft, mit ihrer fachlichen Expertise wertvolle Unterstützung zu liefern. Herrn Dr.-Ing. Christian Connette und Herrn Dr.-Ing. Arne Rost möchte ich ganz besonders danken für die intensiven und anregenden Diskussionen und Fachgespräche. Frau Luzia Schuhmacher M.A. und Frau Heide Kreuzburg gebührt ebenfalls ein besonderes Dankeschön für ihre Geduld und ihre wertvollen Ratschläge bei organisatorischen Fragen.

All den Kollgen, die durch ihre freundliche Art zu einer angenehmen und produktiven Arbeitsatmosphäre beigetragen haben, möchte ich ebenfalls danken, die schöne Zeit wird mir stets in Erinnerung bleiben.

Aschaffenburg, im April 2014 Nikolaus Blümlein

Kurzinhalt

Die frühzeitige Kostenabschätzung ist häufig eine Voraussetzung, um bei der Entwicklung komplexer Produkte deren Wirtschaftlichkeit beurteilen zu können. Die Verknüpfung zahlreicher technischer und wirtschaftlicher Faktoren stellt dabei für Systementwickler insbesondere in sehr frühen Phasen eine Herausforderung dar, in denen nur wenig mehr als die gewünschte Funktionalität bekannt ist. Da die Servicerobotik im Vergleich zu anderen Technologien noch relativ jung ist, verschärft der Mangel an Erfahrungswerten diese Problematik für Serviceroboterprodukte.

In dieser Arbeit wird ein Ansatz vorgeschlagen, der durch strukturale Verknüpfung von typischen Serviceroboterkomponenten, -funktionalitäten und statistischen Kostenmodellen den Kostenschätzungsprozess unterschiedlicher Kostenaggregatsstufen für Serviceroboter-Prototypen in frühen Entwicklungsphasen erleichtert. Zusätzlich ermöglicht diese Vorgehensweise die Abschätzung von Stückkosten für kleine Stückzahlen.

Kernelemente der Arbeit sind die Entwicklung einer Design-Struktur-Matrix für Serviceroboter und die Modellierung von Hardware- und Softwarekosten. Die anhand von Experteninterviews ermittelte Strukturmatrix verknüpft typische Roboterfunktionen mit Hardware- und Softwarekomponenten und ermöglicht dadurch die Ableitung der wesentlichen Bestandteile eines Prototyps aus gewünschten Fähigkeiten des Serviceroboters.

Der Stand der Technik wird bezüglich der gängigen Produktentwicklungs- und Kostenschätzungsmethoden sowie deren Einsatz im Bereich der Servicerobotik dargestellt. Neben allgemeinen Vorgehensweisen der Kostenschätzung werden hardware- und softwarespezifische Konzepte vorgestellt. Entwurfstechniken für neue Produkte werden in intuitive und strukturelle Methoden untergliedert und erläutert. Die Betrachtung dieser Techniken wird abgeschlossen mit einer Untersuchung, inwieweit sie innerhalb der Entwicklung von Servicerobotern Anwendung finden und diesbezüglich Verbesserungsmöglichkeiten bestehen.

Um Kostenschätzungen für Hardwarekomponenten vornehmen zu können, wurden im Rahmen einer empirischen Primärrecherche zentrale technische Eigenschaften und Preise typischer Komponenten als Basis der Modellentwicklung ermittelt. Die vorausgehende Auswahl der untersuchten Komponentenkategorien basiert auf Experteninterviews, der Analyse abgeschlossener Serviceroboterprojekte sowie einschlägiger Literatur. Für jede Kategorie wurden nach der Datenerhebung mittels semi-parametrischer Regression mehrere Kostenmodelle erstellt und das jeweils

geeignetste ausgewählt. Die Güte der Modelle wurde dabei mit Hilfe statistischer Vergleichsmaße beurteilt.

Für Softwarekomponenten bedient sich der Ansatz der analogiebasierten Aufwandsschätzung. Im Wesentlichen wird bei diesem Ansatz der Aufwand zur Erstellung einer spezifischen Software abgeschätzt, indem aus einer Analogie zu einem bereits bestehenden, funktional ähnlichen Programm Rückschlüsse gezogen werden. Zu diesem Zweck wurden mehrere Hundert Module der Open-Source-Softwareumgebung Robot Operating System (ROS) von Willow Garage analysiert. Hierbei wurde zunächst der Umfang von Softwaremodulen in Programmzeilen ermittelt. Anschließend wurde der jeweilige Umfang in ein weiteres Maß für Softwareumfang, sog. Function Points, konvertiert, um den Aufwand mittels Produktivitätsraten unabhängig von der verwendeten Programmiersprache schätzen zu können. Die verwendeten Konvertierungs- und Produktivitätsraten basieren auf heuristischen Daten aus Literaturquellen zur Softwareprojektierung.

Für jedes Kostenmodell wird im Rahmen der vorgestellten Methodik eine statistische Kostenverteilung errechnet. Dies ermöglicht neben punktweisen Schätzungen auch die Errechnung von Standardabstand und Wahrscheinlich-keitsbändern. Dies birgt den Vorteil, dass die Schwankungsbreite von Schätzungen expliziert werden können und somit die Schätzunsicherheit besser beurteilt werden kann.

Um den Schätzprozess zu unterstützen, wurde eine Softwareapplikation entwickelt. Diese stellt eine grafische Benutzeroberfläche zur Verfügung, die den Benutzer durch die wesentlichen Schritte des Schätzprozesses führt. Nachdem der Benutzer Roboterfunktionen und -komponenten ausgewählt und parametriert hat, errechnet das Programm anhand der integrierten Modelle Kostenschätzungen einschließlich Unsicherheitskorridoren. Als zusätzliche Orientierung werden dabei für jede Komponente Parameterwerte basierend auf den vorangegangen Erhebungen vorgeschlagen.

Der Ansatz wurde an zwei unterschiedlichen Datensätzen verifiziert. Zum einen wurden für im Rahmen von EFFIROB entworfene Roboterskizzen Schätzungen erstellt und von Experten mit bestehenden Kostenschätzungen verglichen und beurteilt. Hierfür wurden fünf Szenarien für mögliche Serviceroboter ausgewählt und jeweils mehrere Kostenaggregate mit Hilfe der vorgestellten Methodik geschätzt. Die Ergebnisse wurden auf ihre relative Diskrepanz zu den Originalwerten untersucht und auf Plausibilität geprüft, indem zu jedem Szenario ein Experte, der

die ursprüngliche Schätzung vorgenommen hatte, zur Qualität der neuen Kostenprognose befragt wurde. Zum anderen wurde die Kostenschätzung für einen gebauten Prototypen durchgeführt. Für den Vergleich ursprünglicher und neuer Schätzungen sowie tatsächlicher Kosten wurde der Prototyp "Raser", ein Serviceroboter für autonomes Rasenmähen, ausgewählt. Anhand der Projektdokumentation konnten die ursprüngliche Kostenschätzung sowie die tatsächlichen Kosten ermittelt werden und einer Schätzung gemäß der vorgestellten Methodik gegenübergestellt werden. Für die Beurteilung der Ergebnisse wurde ein Experte, der maßgeblich an der Entwicklung des Prototyps beteiligt war, befragt.

Short summary

Early cost estimation is a common prerequisite for the profitability assessment of the development of complex products. Thereby, the combination of numerous technical and economic factors poses a particular challenge for system developers when little more than the desired functionality of the product is known. In the field of service robotics, this issue is acerbated by the lack of empirical values and experience as the according technologies are comparatively novel.

The presented work proposes an approach which facilitates the estimation process for various cost aggregates of service robot prototypes in early development phases. To this end, typical components and functionalities are structurally mapped and related to statistical cost models. Additionally, the presented approach offers the possibility of unit cost estimation for small lot sizes.

Core elements of this work are the development of a design-structure-matrix for service robots and the modeling of hardware and software costs. The structure matrix determined by expert interviews typical maps robot functions to hardware and software components, thus facilitating the derivation of essential prototype parts from desired service robot skills.

The status quo is presented regarding current product development and cost estimation methods as well as their application in the field of service robotics. In addition to general practices of cost estimation, hardware and software specific concepts are illustrated. Design techniques for new products are divided into intuitive and structural methods and explained. The consideration of these methods concludes with the examination of their application to service robot development and possible improvements in this regard.

In order to develop a model for cost estimation of hardware components, essential technical features and prices of typical components were identified via empirical research. The preceding selection of examined component categories was based on expert interviews, the analysis of finalized service robot projects and relevant literature. Several cost models were developed for each category using semi-parametric regression. The most suitable model was chosen from these based on their quality according to statistical comparative measures.

For software components, the approach applies the principle of analogy-based estimation. Essentially, this approach estimates the effort required to implement a specific software application by inferring it from the analogy to an existing and functionally similar program. To this end, several hundred modules of the open source software environment 'Robot Operating System' (ROS) by Willow Garage were analyzed. After determining the extent of each module, i.e. number of lines of code, it was converted into function points, another measure of software extent. Using the latter in conjunction with productivity rates, required effort can be estimated regardless of the employed programming language. The applied conversion and productivity rates are based on heuristic data from software development literature.

The presented approach also illustrates the calculation of a statistical cost distribution for each cost model. In addition to single point estimations, the distributions allow the computation of deviation and probability bands. The advantage of this additional information is the facilitation of the assessment of the fluctuation range and thus the provision of an indicator for estimation uncertainty.

In order to support the estimation process a software application was developed. It features a graphical user interface guiding the user through the steps of the estimation process. After the user has selected and parameterized the desired robot functions and components, the program computes cost estimations including uncertainty corridors based on the integrated cost models. For additional orientation, the program proposes default parameter values for each component based on the empirical research data.

The approach was verified using two different datasets. On the one hand, estimations for robot prototypes outlined in the EFFIROB study were calculated applying the proposed methodology and consecutively compared with the original estimates and assessed by experts. To this end, five service robot scenarios were selected and several cost aggregates estimated using the presented method. The results were examined regarding their relative discrepancy to the original values. Their quality and plausibility was gauged by interviewing experts who had conducted the according original estimations. On the other hand, costs were also estimated for a built prototype. For the comparison with the original estimate as

well as the actual costs, the prototype 'Raser' was chosen, a service robot for lawn mowing. The original cost estimation and actual costs were extracted from project documentation and juxtaposed to the estimations according to the presented approach. The results were critically reviewed by one of the developers involved in the prototype's development.

Table of Contents

Abbreviation Index

AD	Axiomatic Design
AGV	automated guided vehicle
AIC	Akaike's Information Criterion
BIC	Bayesian Information Criterion
BOM	bill of materials
CBD	Component-based
CBS	component breakdown
CE	cost estimate
COCOMO	Constructive Cost Model
DSM	design structure matrix
EFFIROB	Profitability Analysis of New Service Robot Applications and Their Relevance for Robotics
FP	function point(s)
GAM	General Additive Model
GCV	Generalized Cross-validation
GUI	graphical user interface
HRI	human-robot interaction
HW	hardware
IPA	Fraunhofer Institute for Manufacturing and Automatisation
LOC	lines of code
OP	object point(s)
pdf	probability distribution
PERT	Program Evaluation and
PLT	Programming Language Table
R&D	research and development
ROS	Robot Operating System
SEROCOST	Service Robot Cost Estimation
SLIM	Software Lifecycle
SPR	Software Productivity
SW	software
UFP	unadjusted function points
US	ultrasonic sensor
USD	US Dollar

VDI..... Verein Deutscher Ingenieure (Association of German Engineers)

WBS.....work breakdown structure

Symbol Index

a	software project coefficient
AFP	adjusted function point(s)
AIC	Akaike's Information Criterion
b	software project coefficient
BC	best case estimate
BCC_{Devel}^{SW}	best cast estimate for
BIC	Bayesian Information
C^{HW}	aggregated hardware costs
C_{Admin}^{HW}	hardware administrative cost
$C_{Install}^{HW}$	aggregated hardware
$c_{Install}^{HW}$	hardware installation cost per
C_{Labor}^{HW}	aggregated hardware labor
C_k	environment or technology
$c^{Prototype}$	prototype costs
C^{RD}	development and designing
$C_{Install}^{SW}$	software installation cost per
$c_{Install}^{SW}$	aggregated software
c_i^{unit}	material cost of robot unit of component instance i
D	deviance
DDC	development and designing
d_{month}	working days per month
DoF	degrees of freedom of model
DUC	direct unit costs
E	development effort
$E(x)$	expectancy value of x
e	relative average estimation
GCV	generalized cross-validation
K	development effort (SLIM)
k_j	lines of code per function point rate for language j
$k_{j_{BC}}$	best case lines of code per function point rate for
$k_{j_{WC}}$	worst case lines of code per function point rate for language j

$LOC_{i\,j}$	lines of code of package i in programming language j
MLC	most likely estimate
MC	manufacturing cost
N	number of data samples
n_c	number of component type
n_i	number of components in
n_{units}	number of manufactured
p	number of parameters (AIC)
PUC	product unit costs
R^2	coefficient of determination
r_i	reuse ratio of package i
RSS	sum of squares of residuals
S	expected software size
$s_{i\,j}$	function point size of package i's part coded in language j
S_s	software size output in LOC
TC	total cost
t_d	software delivery time (SLIM)
TDI	total degree of influence
t_{SW}	software development staff
$t_{SW\,i}^{D}$	software development staff time for implementation of new code for package i
$t_{SW\,i}^{R}$	software development staff time for acquiring and adapting reusable code for package i
UFP	unadjusted function points
VAF	value adjustment factor
$VAR(x)$	variance of x
v^{D_C}	development rate for non-reusable, custom code
v^{D_R}	development rate for
VIF	variance inflation factor
v^{R}	rate of acquisition and adaptation of reusable code
WC	worst case estimate

.....	worst case estimate for software development cost
.....	software developer monthly
.....	engineer monthly wage rate
α.....	system designing cost markup
β.....	ratio of manufacturing cost to total cost
γ.....	ratio of development and designing cost to total cost
ε.....	estimation error
θ.....	parameter vector of likelihood function
μ.....	mean value
σ.....	standard deviation

Figure Index

Table Index

1. Introduction

The field of service robotics is a comparatively young area that is expected to economically boom within the next decades into an economically significant multi-billion Euro market (Hägele 2011, pp. ix–xi). For this to come true, the cost-effectiveness of service robots must be assured. However, only few studies on economic aspects of service robots have been published, most articles seem to focus on technical feasibility and solutions.

The lack of cost consideration has led to many prototypical service robots that were able to fulfill a certain purpose but prohibitively expensive (Prassler 2010). However, if profit-oriented enterprises are to engage in this field they must be able to predict the profitability of those undertakings that concern the development and marketing of a service robot in order to make rational planning.

1.1. Problem

Estimating the development cost of a prototypical robot is difficult for several reasons. Most technical cost estimation methods require more detail knowledge than is available at early development stages thus basically reducing the appropriate selection of methods to expert opinion and comparison to similar products if possible. Also, service robots represent complex systems where the costs cannot simply be reduced to material costs as is often the case for simple products like nails or pencils; higher complexity entails an elevated development risk. Estimating the efforts and costs for developing a service robot requires expertise not only in a variety of technical fields such as electronics, systems engineering and software development but also an understanding of costing concepts and risk management. Furthermore, the estimator must be knowledgeable about feasible service robot solutions and their possible mapping to complex tasks. Bridging these diverse fields poses a significant challenge to single estimators. Furthermore, the cost of the estimation itself must not be prohibitively expensive as that would lead to abandonment of the attempt to predict project costs at all.

Often, the estimation of manufacturing costs is based on the hardware structure. But nowadays, the software development has become equally as important for

service robots as the study EFFIROB on profitability of innovative service robot applications has shown; because service robots are complex systems the software development costs are considerable (Hägele 2010, p. 337). This major cost block is incurred in preproduction phases whereas the direct software costs (i.e. per unit produced) in the manufacturing stage are negligible. Therefore the classical definition of production costs falls short for a complex product and cost intensive constituents of a system where hardware and software are equally important as it does not capture the indirect cost incurred in the development stage.

The dilemma is exacerbated by the phenomenon that up to 90% of the life-cycle costs are determined in planning and development phases although the majority of the costs are incurred during production as depicted in Figure 1.1-1 (Ehrlenspiel 2007a, p. 11). This means that solid decision-making is highly important in early phases in spite of bounded rationality; at the time detailed product knowledge is available the possibilities of influencing costs will have decreased significantly.

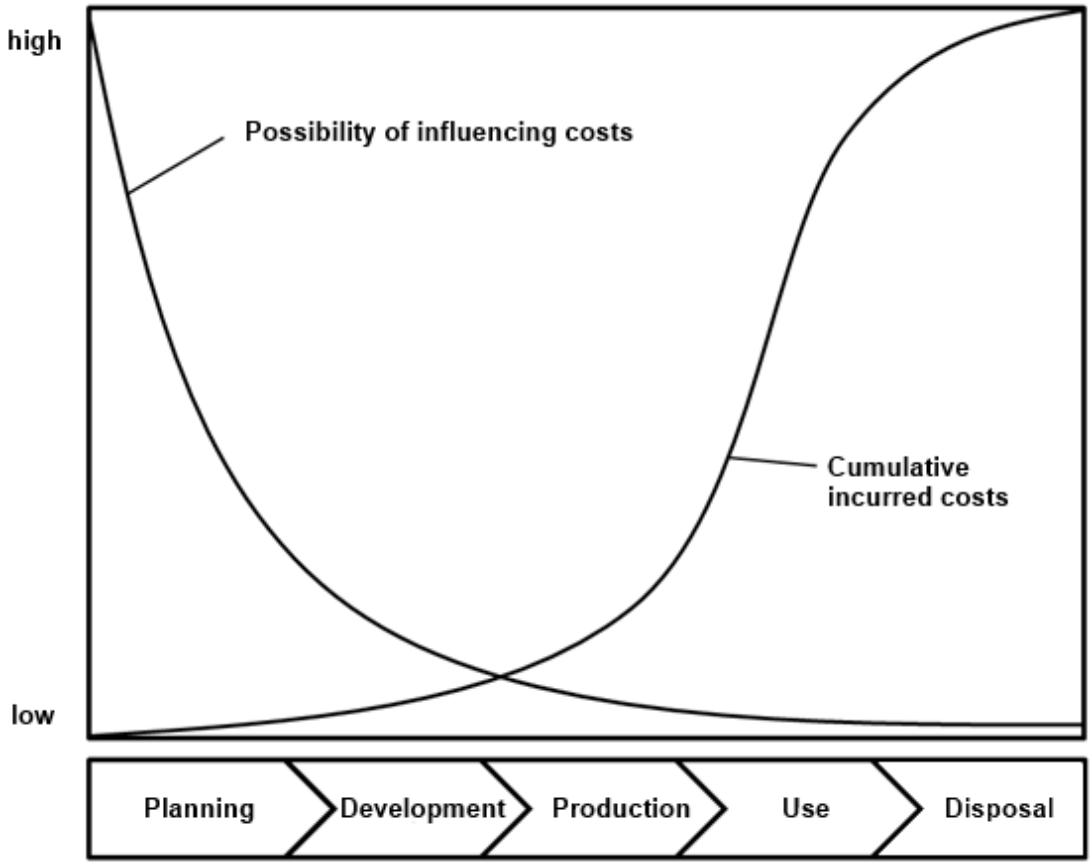

Figure 1.1-1 Influence on costs vs. incurrence of costs over product life-cycle

1.2. Motivation

Enterprises interested in serving the market for complex products as service robots require means to predict the profitability of such engagements (Trivailo 2012, pp. 2-5). Committing to the development of a service robot that is technically feasible but economically not viable could result in profound financial losses therefore early cost considerations are the developer's responsibility (Ehrlenspiel 2007a, p. 62). Ex ante cost estimation also becomes increasingly important for research institutes aspiring to receive public funding; as an example the substantiation of the profitability of proposed robot concepts was mandatory in a recent call for proposals in the field of service robotics by the German Federal Ministry for Research and Education (BMBF 2012). Cost estimates can also be used for comparison of variants, recognition of cost reduction potentials, production controlling or supplier cost examination i.e. they serve a variety of controlling and management purposes.

Therefore, cost estimation methods and tools that are reliable and produce realistic results on one hand and allow uncomplicated, quick and traceable estimates on the other hand could provide substantial support to service robot experts. Porter states that "an approach for reducing the complexity of the forecasting process is highly desirable" in the context of developing competitive strategies in emerging markets (Porter 2004, p. 234). Bridging the gap between the current lack of pragmatic prognostic methodologies and applications in the field of service robotics contrasted with the obvious benefits of sound cost estimation is the driving motivation behind this thesis.

1.3. Thesis Objectives

The primary objective of this thesis is to provide support for cost estimation in early planning and development stages of service robot prototypes. To this end, a pragmatic methodology is presented drawing on statistical data and expert knowledge formalized in cost models and structure matrices. The integration of the approach in a software tool is aimed at allowing quick and flexible yet reliable cost estimation when little more than the desired functionality of the service robot is known. The further the project advances, the more accurate the estimates the tool can deliver due to the increment in available information on the solution; the tool

can thus be used as an accompanying project management tool. These estimates can be used for decision making and assessments processes in product and project management and controlling.

Estimating prototype costs with the methods proposed gives the estimator a more comprehensive perspective on potential cost impacts of a specific service robot concept because it creates the possibility of comparing and aligning the results with estimates derived by different methods e.g. ad hoc expert opinion. It also provides the possibility of quick comparison of alternative variants by simply changing the selection of basic components which are used as an input to the cost estimation.

The approach approximates probability distribution of estimated costs thus allowing the indication of estimation ranges. These cost corridors improve the assessment of uncertainty inherent to the estimation as the additional information enables the estimator to judge magnitude and probability of deviation from the estimated value.

A secondary objective pursued in the presented approach is the derivation of unit costs for a specific service robot concept. To this end, various cost constituents estimated for prototype development are combined to extrapolate two forms of unit costs: direct unit costs i.e. costs that are incurred per robot manufactured and product unit costs which are construed as the direct unit costs plus the costs of development not attributable to an individual unit. One limitation of the presented approach regarding this objective is that it can only be applied directly to small number of produced units as economies of scale and mass production are not considered here.

1.4. Thesis Outline

Chapter 2 Preliminaries introduces the required terminology used in the presented work by giving definitions and explanations on essential service robot and cost subjects. Furthermore, the general assumptions made in the scope of this thesis are laid out. This chapter also points out the estimator's requirements concerning an estimation approach.

In Chapter 3 State of the Art and Related Work current methods and applications of cost estimation and product development are discussed. Moreover, their use in relation to service robotics and respective shortcomings are indicated.

Chapter 4 Function-based Cost Estimation Approach gives an overview of the complete approach. The estimation process is outlined in several steps and the methods for collecting data and constructing required dependency matrices and cost models are explained.

Chapter 5 Function-based Structure Estimation details the categories used for describing service robot functionality and structure. Furthermore, the combination of these categories into a structure matrix is presented. The structure matrix is the employed means to map the dependencies between components and serves the derivation of an approximate robot structure.

In Chapter 6 Component-based Cost Estimation, the modeling of prototype costs is explicated. These costs are based on the individual component cost models of the service robot structure. Cost constituents are established and the pertinent cost model derivations are formulated.

Chapter 7 The Software Tool SEROCOST presents the basic concept and features of the software application that has been developed within the scope of this work for direct and uncomplicated use of the proposed methodology.

Chapter 8 Experimental Validation explains the validity of the proposed cost estimation; because the presented approach is not amenable to statistical validation in a strict sense, use case estimations are computed and subsequently assessed by individual experts. Consequently, findings on estimation suitability and usefulness are discussed.

The thesis closes with Chapter 9 Conclusion. It contains a summary and discussion of the contributions and an outlook on further extensions and potential improvements of the presented approach.

2. Preliminaries

This chapter serves the clarification of the scope of the presented work. Essential terms and concepts are described, the general underlying assumptions explained and objectives of primary addressees of the cost estimation approach outlined.

2.1. Terminology

2.1.1. Service Robot

A service robot is defined as "a robot [...] that performs useful tasks for humans or equipment excluding industrial automation applications" (ISO 8373, p. 3); a robot is an "actuated mechanism programmable in two or more axes [...] with a degree of autonomy [...], moving within its environment, to perform intended tasks" (ibid., p. 2). The robots considered in this approach are ground-based mobile service robots according these definitions. Although it is generally applicable to a wider range of robots (or even engineering solutions outside of robotics for that matter) the high exigencies for time and effort of data acquisition required for deriving cost functions necessitate a focus on specific areas; against this background aerial and marine platforms are beyond the scope of this work.

Figure 2.1-1 Care-O-bot 3 by Fraunhofer IPA

Figure 2.1-2 PR2 by Willow Garage

Further restrictions stem from limiting the parameter ranges of the considered hardware components which are outlined in Chapter 2.1.3; effectively, these constraints limit the considered service robot to small and medium-sized robots

weighing no more than 650 kg.[1] This restriction to this subset is deemed justifiable as it covers the majority of service robots: in 2007, Piperidis et al. reported that only 27% of service robots weigh more than 10 kg (Piperidis 2007, p. 1). Popular examples of medium-sized service robots like the Care-O-bot 3 (186 kg) by Fraunhofer Institute for Manufacturing Engineering and Automation shown in Figure 2.1-1 or the PR2 (220 kg) by Willow Garage depicted in Figure 2.1-2 (Willow Garage 2009c) stay far below maximum theoretical weight considered in this approach.

Service robots interact with their environment which is why the term 'robot system' sometimes includes the environment (e.g. Dalgaard 2010, pp. 10–13; Lindemann 2005, p. 10). In the scope of this work, 'system' refers only to the robot itself; the environment is only considered insofar as structural components external to the robot are absolute prerequisites for its functionality (s. Miscellaneous Environment Hardware in Chapter 5.1.2.2).

2.1.2. Skill

At the time of writing this thesis, no general application or definition of the terms 'skill' existed but it has been used in combination and sometimes in distinction with the terms 'function', 'skill', 'action', 'task', 'application', 'mission' and others (e.g. Bischoff 2009, p. 7; Smits 2010, p. 6). For this reason, it is deemed to be of importance to clarify the term's use in the scope of this work; it is not the intention of the author to create a universally valid definition.

In this work, the term 'skill' is interpreted as a class of intermediate granularity that comprises purpose-oriented activities frequently occurring in a service robot's mode of operation, i.e. each skill could be decomposed into further sub-skills of a finer granularity.

For a concrete service robot application area specific skills can be detailed. However, the aim of the presented cost estimation approach is to cover a variety of service robots which requires a certain degree of generalization. For this reason a skill is defined here as a generic ability to deliver an expected functional performance. As

[1]The weight constraint stems from restricting the maximum weight of the platform to 350 kg and its maximum payload capacity to 300 kg; s. Chapter 5.1.2.2.

an example, the specific tasks 'cut wire of thickness x' or 'grind metallic surface' require specific skills but can be subsumed under the generic skill 'process object'. The list of skills developed in this work is not regarded as exhaustive rather than a compilation of robotic abilities that are demanded of many service robots, based on the experience of robot engineers and current literature on service robots (s. Chapter 5.1.2.1).

2.1.3. Component

Components are the building blocks of a service robot. In the scope of this work, these do not only comprise hardware i.e. physical equipment but also include software components. Similar to the generalization issue described for skills, abstract categories for components are essential to the approach being applicable to a range of different service robot concepts. In the scope of this work, several terms are used to clarify the level of concretization.

Component type is the most general expression applied in these considerations. A component type represents a generic category of components which serves similar technical purposes and characteristics and thus can be described by a common set of parameters. In contrast, *component* as the most specific term describes an actual element within a component type category specified by a determined set of values for the component type parameters. As an example, all robot arms (component type) are "interconnected sets of links [...] and powered joints [...]" (ISO 8373, p. 7) and allow positioning of an end-effector within its working space but a variety of implementations (components) differing in size, shape, number of degrees of freedom etc. exists.

Furthermore, the term *component instance* is used which stands for an entire set of identical components i.e. the determining parameter values of a component instance are the determining parameter values of a specific component and additionally the quantity of these components per robot unit. This term was introduced to clarify the distinguishing between differently parameterized components of the same component type category. Figure 2.1-3 illustrates the component terminology with an example for cameras.

Each component type is assigned an individual cost model which permits the cost estimation per cost instance using the respective parameter values as model input. As these cost models have been determined by statistical regression analysis the data required to build the models was collected beforehand. The dilemma in this context is that the component types must be generic enough to fit a variety of components but this abstraction complicates the selection of representative data because it increases the size of the population to be surveyed. In order to improve model accuracy and reducing the population size the ranges for component type parameters were restricted within plausible ranges for the service robot types considered. Examples are the maximum payload for robot arms which was set to 10 kg or platform maximum weight at 350 kg; detailed explanations of component-related restrictions are given in Chapter 5.1.2.2.

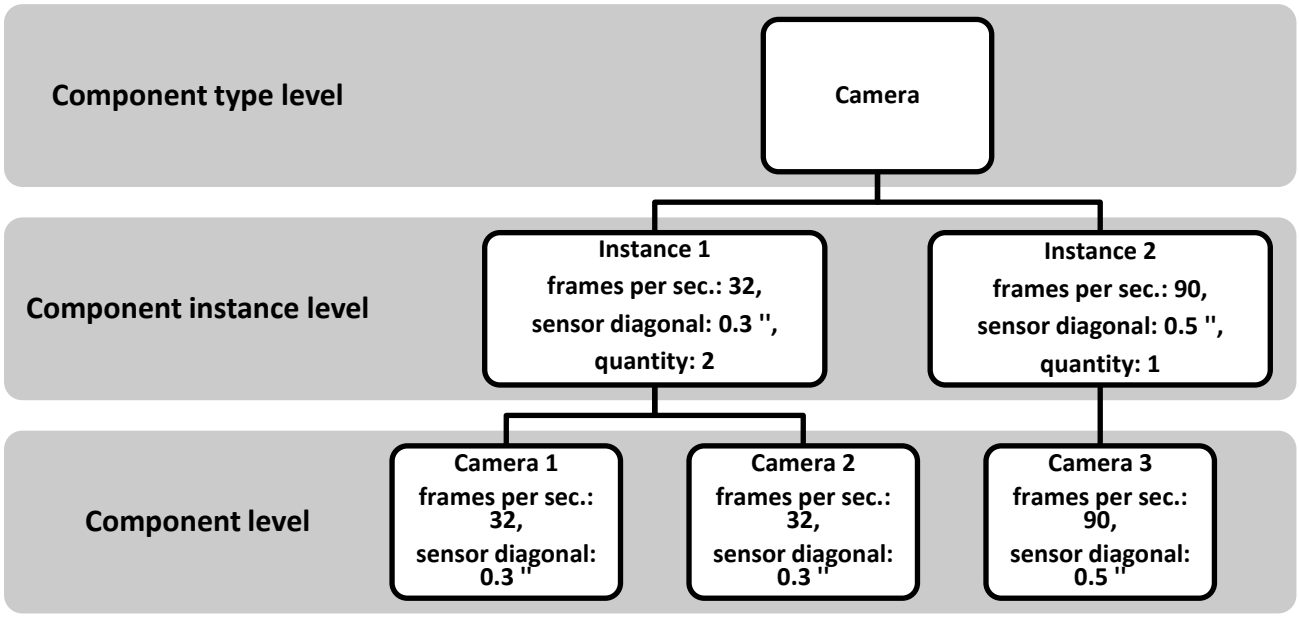

Figure 2.1-3 Camera example of the component terminology

2.1.4. Cost Types

Costing theory offers a wide range of cost terms. From a business-oriented point of view, costs are the consumption of goods and services incurred with the purpose of creating products or services, evaluated in monetary terms (Coenenberg 2009, p. 24).

Costs can be classified according to a number of characteristics depending on the focus of the specific consideration (Coenenberg 2009, p.25, pp.57-95; Plinke 2000, pp. 23–39). They are the dominant metric for planning and controlling the economic success of an enterprise. Classifications of costs help to fairly assign costs to specific business-oriented decisions according to the input involved – the principle of causation – in order to assess the economic viability of the latter.

In this work the considered costs are classified by factor consumption i.e. labor and material costs and by allocation unit i.e. costs per produced unit or costs incurred by the product type. A detailed explanation of cost types is given in Appendix 10.1.

2.1.5. Early development stages

In order to clarify the classification of early product development stages at which the estimation approach in this thesis is aimed, several current models of generic development processes are outlined in the following section.

Kossiakoff and Sweet regard the product development process as an eight phase system development life-cycle (Kossiakoff 2003, p. 52). The different phases can be subsumed under the three main development stages concept development, engineering development and post development as depicted in Figure 2.1-4 (ibid.; also cf. Dalgaard 2010, pp. 94–98).

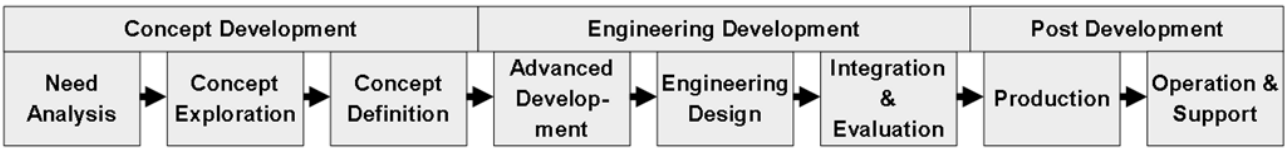

Figure 2.1-4 System life-cycle model of systems engineering by Kossiakoff and Sweet

The concept development stage relates to the first phases of product development and is concerned with issues of principal requirements, feasibility and product characteristics. The engineering development stage comprises phases of detailing the product concept by identifying and analyzing remaining technical problems, engineering component and subsystem solutions and incorporating them in a final product. Post development consists of the production and related engineering processes and after sales services and improvements. For each phase, applying an iterative method of the four successive steps requirement analysis, functional

definition, physical definition and design validation is suggested (Kossiakoff 2003, pp. 69–70).

Ulrich and Eppinger propose a similar generic product development process comprising the six phases as depicted in Figure 2.1-5 (Ulrich 2012, pp. 11–32).

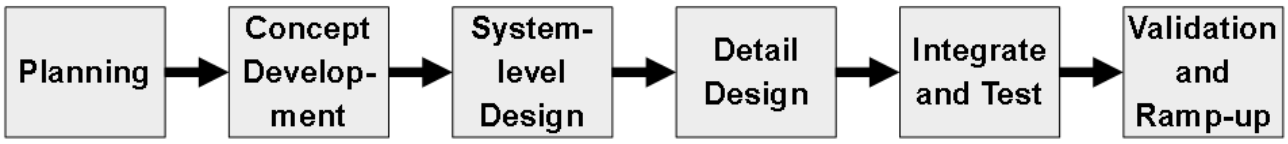

Figure 2.1-5 Generic product development process by Ulrich and Eppinger

The first two stages represent early development activities where technical and economical feasibility are focused. The following three phases concern the concrete product design in a 'top-down-top' manner i.e. first the overall system design is decided upon then the detailed solutions are developed and subsequently integrated into one whole. The final phase of this model incorporates final adjustments and production start. For each phase, executing a variety of activities related to marketing, design, manufacturing and other functions is suggested (ibid., p. 14). As Dalgaard points out this model basically reveals the same structure as the system life-cycle model by Kossiakoff and Sweet (Dalgaard 2010, p. 98).

The *VDI guideline 2221* proposes a comparable general procedure for development and construction which consists of seven steps structured in four overlapping phases as displayed in Figure 2.1-6 (VDI-Richtlinie 2221, pp. 9–13).

The procedure starts with a task, i.e. an application the product to be designed is aimed at. After establishing the customer's demands and related requirements in the first step the overall product functionality is determined and decomposed into subfunctions arranging them in a structured form. Subsequently, technical solutions for these functions are investigated. These solutions are then organized in feasible modules, i.e. they are mapped to real elements and groups vital for the implementation and the interfaces between them are specified. In the next step the essential concretization through detailed design of these significant modules takes place. In the penultimate step of the procedure the partial designs are finalized and integrated into the overall product design. Finally, the results of the product development and construction are documented so that product implementation,

features and usage can be used in successive processes, particularly in production. The procedure is flexible because at any given step a backward or forward iteration to other steps is possible; nevertheless, no step should be left out (ibid., p.9, p.11).

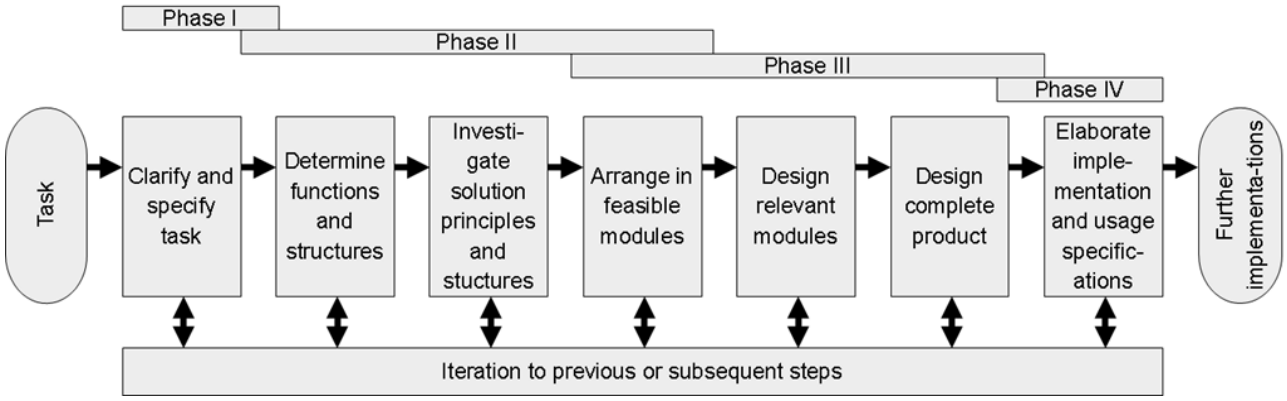

Figure 2.1-6 General procedure for development and construction according to VDI 2221

Although the VDI procedure does not directly address the production and post production stage it bears profound similarities with the two other models presented here. They all start at the refining of task requirements and functions, continue with the search for solution ideas which are consequently decomposed into parts and partial designs and finally integrated into one product.

In the scope of this thesis, early development stages are understood as those where application, main functionalities and approximate structure of the service robot are determined but no detailed design yet. This notion coincides with the first three steps of each of the development process concepts described above.

2.2. General Assumptions

In order to guarantee the assessability of the estimation model's quality the underlying assumptions must be disclosed and justified. The estimation approach in this work stands on the premises described in following paragraphs.

2.2.1. Manual Assembly/Small-series Production

The assembly of the robot units is executed in a small-series, manual fashion, i.e. no assembly lines or other mass production means are employed. This is in line with the fact that most service robots of today are produced in small numbers and mainly assembled manually (Botthof 2011, p. 29). This assumption also entails that no

explicit distinction between functional model and prototype development is made in this work.[2]

2.2.2. Existing company

The assumption of an existing company entails the premise that the robot development enterprise is not founded specifically for the purpose of the robot project but is already in business (not necessarily in robotics) and thus does not need to render capital investments in production facilities. This assumption also signifies that the necessary manpower is already employed at the company; variations in employment due to service robot development are not considered.

2.2.3. Prototypical Robot

The robot whose costs are to be estimated is assumed to be a prototype. This assumption entails the exemption of series production issues (e.g. patent claims, safety restrictions, etc.). There are several reasons for this assumption. Firstly, legal matters tied to series production can form a major constituent but are largely locally determined. Taking these complex and highly variable issues into account is beyond the scope of this approach. Secondly, larger series production of a finalized product offers many possibilities to drive down unit costs, e.g. using automated assembly lines, buying materials in large quantities or redesign of components (Ehrlenspiel 2007a, pp. 153–159). The study of cost-efficient production is a separate field of study. Implying its potential cost impacts requires complex models that could result in inapplicability or at least being impractical for early cost estimation purposes.

2.2.4. Off-the-shelf Hardware Components

The hardware components used in the robot are assumed to be parts commercially available on the market and no custom products, i.e. off-the-shelf (OTS) components. This means that no costs for the development of individual hardware components are considered, or that if parts are developed individually their costs are similar to OTS components. The assumption is necessary because developing separate cost models for prototypical components is beyond the scope of this

[2]For the possible distinction between functional model and prototype s. (Masing 2007, p. 500).

thesis. The assumption appears to be justified by the current availability of competing commercial products for each of the hardware component type categories investigated; also, custom parts need to lie in a comparable price range in order to be competitive.

2.2.5. Component-based Software Development and Software Reuse

Similar to the assumption of using available hardware technology it is assumed that existing open-source robot software is reused and adapted rather than building the software from scratch. Also, the software paradigm of component-based development is implied which is an appropriate match for the modular structure of robot systems and is considered the most viable approach to handling the involved complexity (Brugali 2007, p. 13; Brugali 2009, pp. 84–96). This assumption entails that no fundamental research is undertaken, i.e. cost for research isolated from specific products or components is disregarded.

2.3. Estimator Requirements

The presented cost estimation approach and the pertinent software tool are aimed at individuals who need to assess the economic viability or profitability of a service robot as a planned product in early development stages e.g. robot scientists, product engineers or project managers; in the scope of this work these are referred to as estimating persons or estimators. For early phase cost estimation and the corresponding cost models to be useful the key demands are that "they should be acceptable to intuition and experience, should be simple and transparent with traceable logic and ground rules, and have an applicable data base" (Meisl 1988, p. 100). This section details the requirements from the estimator's perspective and how they are met.

2.3.1. Intuitive Application

The cost estimation process should be intuitive and fairly easy to apply. If the cost estimation is too time-consuming and complicated it creates costs in itself which increases the likelihood of the process being avoided altogether (Wierda 1988, p. 190). Intuitive use is particularly important for developers who are technical experts and only have minimal exposure to cost issues because the solutions they design determine a majority of the product's costs (ibid., pp. 189–190).

In order to enable quick and comprehensible usage of the presented approach a software application was implemented that allows use of estimation data and models without forcing the estimator to research technical or economic details for frequently used robot components. The software not only calculates cost estimates for components but also provides suggestions for structural dependencies between components and functions, thus supporting decisions on the robot's structural design. Furthermore, default values are provided for every parameter and are based on expert judgments, statistical analysis and further research of current technical literature and software. Default values can be overridden by the user.

2.3.2. Traceability of Estimation

In order to render the estimation comprehensible the estimation process and its results must be traceable for the estimator. For this reason, the development of every cost model for the different cost constituent is explained and the results displayed separately for each component as well as in aggregated form so that the estimator can retrace the calculation.

The software tool also permits the adaptation of all cost parameters which enables the estimator to calibrate cost results according to his experience and to conduct sensitivity analyses thus fostering the estimator's understanding of the outcome.

2.3.3. Applicability

The data base of the structure and cost models must be applicable to the field of interest i.e. the development of service robot prototypes. To this end, lists of typical service robot skills and components were compiled and validated by robot experts at the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA). These catalogs were deliberately designed to be of a generic nature so that the individual items can accommodate a wider range of conceivable robot designs. In order to avoid over-generalization the component configuration can be tuned to the concrete design idea by adapting technical component parameter values like weight, selection of software required etc.

The hardware and software component cost models are based on data research specifically conducted within the field of service robotics i.e. the components that

were chosen for analysis are currently used and integrated in working service robots.

2.3.4. Uncertainty Assessment

All estimations are subject to uncertainty. As an early phase estimation has the purpose of supporting development decisions it is important that the estimator be able to assess the degree of uncertainty inherent to the estimation. Unfortunately, the true accuracy can only be evaluated after the real costs are known so they can be compared to the estimate i.e. from hindsight.

To permit the estimator assessing the level of vagueness, the presented approach provides means to calculate not only single point estimates but also range estimates which give the estimator information on the corridor of most likely costs; a wide range indicates a high degree of uncertainty. This additional information improves the estimation result by facilitating its interpretation.

3. State of the Art and Related Work

The concept of cost estimation can be applied to a variety of projects. It is usually part of the new product development (NPD) process and used to assess the profitability of a product idea or to derive business plans, budgets and schedules (Ehrlenspiel 2007a, p.62, pp.423-426). Against this background, different cost estimation approaches are presented in this chapter. As the structure or design of the product considered are central to cost estimation current methods of determining product structure are also outlined.

3.1. Cost Estimation

This section exhibits general cost estimation categories that can be applied to all kinds of costs e.g. project cost estimation where the estimation is extended beyond the component costs to the costs incurred on a system-wide level, e.g. overall design, administrative costs or management costs.

Furthermore, the most common approaches for hardware and software cost estimation are presented. There are two reasons why hardware and software components are treated separately:

1. The dimensionality of hardware versus software parameters are of different natures: Whereas most hardware parameters are physical, e.g. size, weight, forces, current etc. software cannot be described by such parameters as it is a logical concept, not a physical object.
2. Whereas software costs consist mainly of labor cost, hardware costs are a combination of material costs and labor costs. Thus, the different approaches for the respective domain differ in their focus.

It is noteworthy that in comparison to hardware cost estimation software estimation appears to be drawing more attention to researchers. Table 3.1-1 shows the number of search results for strings related to cost estimation supporting this impression.[3]

[3]Date of search: 06/06/2012

	"hardware…			**"software…**		
	Google	Google Scholar	IEEE Xplore	Google	Google Scholar	IEEE Xplore
…cost"	464,000	32,300	1,454	963,000	15,300	982
…cost	26,200	96	3	136,000	5,500	887
…cost prediction"	1,840	5	1	9,350	84	0
…cost	6	8	0	13,200	35	0
…development effort"	62,600	191	2	133,000	6,530	121

Table 3.1-1 Comparison of search results for estimation related terms

Costs can also be viewed from the customer's perspective. The initial cost is the price the user has to pay for the product and usually is higher than the manufacturing costs; otherwise the selling company cannot be long-term profitable. Demand-based cost estimation tries to answer the question "how much is the user willing to pay for a specific product?" whereas the cost estimation in the presented approach analyzes the cost of developing and manufacturing a certain kind of product, namely service robots. Thus typical market-oriented approaches like target costing, price-to-win, value analysis, preference matrix method or similar are not considered in the scope of this work.

3.1.1. Basic Approaches

As there is a large variety of different approaches for cost estimation a categorization of general methods facilitates understanding their advantages and drawbacks although the distinction between these categories is fuzzy. The majority of the approaches applied in practice are hybrid forms of the basic methods, e.g. expert judgment can be used as the basis of a parametric mode; some type of parametric model and work breakdown structure (WBS) or component breakdown structure (CBS) can be identified in most of the practiced methodologies (cf. Williams 1994, p. 4). These methods can be applied to the estimation of hardware and software development as detailed below, but they can also be applied in a more general fashion to estimate total project costs or those cost constituents that cannot be related to a specific part of the development project e.g. project management. The following are the most common categories of cost estimation approaches.

Expert opinion or *expert judgment* is the elicitation of subjective opinions from experts of the specific field. Their assessment is usually based on experience from previous projects that bear similarity with the project to be estimated. This approach is very common for early project effort estimation on a macroscopic level (ibid., p. 2).

Cost estimation by expert opinion is the most common estimation approach and offers several advantages (Molokken 2003, p. 39). Estimates can be very accurate if the expert is knowledgeable; some studies reveal that expert estimates are not inferior to other methods and can even outperform them in terms of accuracy (Hughes 1996; Molokken 2003, p. 42). Furthermore, implicit expert knowledge often is the only reliable source of data when documentation is scarce or fragmentary which often is the case for new technologies. Another benefit of most expert judgments is their low impact on project costs and quickness of delivery (Rush 2001, p. 132).

The problems associated with expert estimation originate from its subjectivity. Whereas good expert deliver accurate estimates, inexperienced ones may be far out in their calculations. Expert estimations tend to be too optimistic (Jones 2007, p. 67). With a large amplitude of estimation error it is difficult to assess an estimate's quality. In the same vein it is hard to judge an expert's proficiency by other measures than the success of his or her previous estimates.

The difficulties in assessing the estimate are associated with the lack of systematization of the approach. The expert relies on his subjective and partly tacit knowledge without disclosing every assumption he makes thus increasing the probability of bias (Roy 2003, p. 3); given the problems of subjectivism, full disclosure is either not possible (Polanyi 1985)[4] or requires considerable knowledge management efforts (Polanyi 1985; Nonaka 1995). This lack of structure reduces the comprehensibility and verifiability of the estimate (Rush 2001, p. 132). A further inconvenience of this lack in structure is its low reproducibility of results; different

[4]Polanyi hypothesizes that implicit knowledge can never be fully rendered explicit.

expert make different assumptions and experts leaving a company cause drain of estimation knowledge.

Parametric modeling, sometimes referred to as *algorithmic modeling*, implies the derivation and use of mathematical equations and algorithms to estimate costs. The model reflects the causality between certain input data, e.g. software size and output data, the cost estimate. The central assumption is that a number of important product features are the main cost drivers and thus determine the cost function (CBET 1990, p. 36). It is based on knowledge from historical data, research results and plausible assumptions where no data is available.

The model's complexity usually depends on the complexity of the underlying product. Simple linear models with only one input variable can be sufficiently accurate for plain products, e.g. cost as a linear function of the weight for cog wheel production estimation (Ehrlenspiel 2007a, pp. 432–433; Wierda 1988, pp. 190–191). For complicated products like large software developments more sophisticated, often non-linear models have been developed, e.g. Boehm's Constructive Cost Model COCOMO (CSE 2000).

Because parametric models are expressed by mathematical equations they represent highly systemized approaches. The derivation of the model and the underlying assumptions are usually documented thus rendering estimates repeatable, verifiable and comprehensible due to their transparency (Roy 2003, pp. 4–5). However, if documentation is lacking or disclosed e.g. for proprietary estimation models the validity of the model is hard to retrace (CBET 1990, p. 36). The mathematical structure also allows the application of different statistical measures which can facilitate the assessment of the estimation quality and its degree of uncertainty. The model's accessibility to computation also permits implementation of software tools enhancing and accelerating the estimation process.

As most models contain a number of adjustment parameters they can be adapted to different projects without major efforts. This renders parametric models flexible and usable for repeated estimation.

The main disadvantage of parametric modeling lies in the efforts necessary to create the model. Whereas parametric estimation is fast once the model is available it requires considerable amounts of research and resources to derive the underlying equations. In order to construct a model historical data must be accessible from which to extrapolate; if no data is available it must be acquired through costly research activities. Also, the applied assumptions, e.g. if the model is additive or multiplicative, linear or exponential, must be justified based on verifiable facts or at least plausible rationalizations which necessitate further investigative efforts.

Another drawback of parametric models is their inapplicability to radically new products and technologies. Although this issue applies to all estimation approaches as they are all based on some form of previous experience it particularly weights parametric models because the efforts put into creating a previous model are of little value for the new estimate (ibid.).

Estimation by analogy, *case-based reasoning* or *proxy-based estimation* is the comparison of similar previous projects to the object of estimation and is often closely related to expert judgment. The closer the compared project's characteristics are to each other the higher the quality of the estimation. This approach is often combined with parametric models (Keung 2008). Analogy-based estimation can be applied on product level or on component level in combination with the bottom-up engineering approach (s. below). Analogy-based estimation can be fast, intuitive and accurate if data on similar products is accessible to the estimator which often is the case for enterprises with long-term experience in the respective field (Shepperd 1997, pp. 737–738; Shepperd 1996, p. 174).

The difficulties in using analogies for estimation purposes lies in the identification of relevant differences between the new project and the analogy and in the recognition of their impact on costs. Differences can arise from a multitude of factors such as production facilities, material price fluctuations, personnel skills and many more. If crucial differences are not identified the risk of the estimation being overly biased is high and can lead to misleading conclusions. Furthermore, analogy-based estimation techniques are not applicable if no proxy can be found to compare the new project to, a common problem for novel technologies (Roy 2003, p. 9).

Cost estimation approaches can also be separated into *top-down* and *bottom-up estimation methods*. In top-down estimation the global properties of the product are used to arrive at a single estimate which can be broken down to more detailed levels, e.g. by dividing material costs from labor costs (Williams 1994, p. 2). Top-down approaches lend themselves to early cost estimation when no detailed information is available yet. Estimates can be fast but tend to be rather inaccurate (CSSE 2002, Chapter 5).

Bottom-up estimation, also referred to as *grass roots estimation* describes estimating processes starting at the component level with cost estimates for each component or project activity and subsequent aggregation of the part estimates to the overall estimate (ibid., Chapter 4). The estimation can be executed from a project perspective concerning activity costs, from product perspective with focus on components or in a combined form, depending on the aim of the estimate. These estimations are based on a WBS or CBS. Such a breakdown structure is the decomposition of the main project into its constituents to a determined level of detail.

In general, the advantages of bottom-up approaches reflect the disadvantages of top-down methods and vice versa. One advantage of the former is its potential for high accuracy if a comprehensive WBS or CBS is available because estimates are generated at a level of detail top-down approaches usually cannot deliver (Williams 1994, p. 2). A further effect of bottom-up aggregation of estimates is the averaging of the random error i.e. the random error decreases (Ehrlenspiel 2007a, pp. 453–456). The compilation of the breakdown structure also renders the cost estimate transparent and comprehensible.

The downside of bottom-up estimating is that it is a more expensive approach than top-down estimation because it requires detail knowledge to generate the breakdown structure (Trivailo 2012, p. 8; Williams 1994, p. 2). Also, the information necessary for a WBS or CBS is less likely to be available in very early development stages, particularly if the degree of innovation of the product to be estimated is high. Additionally, the risk of omitting crucial cost-driving elements remains, which can reduce the accuracy of the estimate.

3.1.2. Hardware Cost Estimation

Hardware costs can be interpreted in a narrower sense as material costs only or in a wider sense as all hardware-related costs which also include labor costs e.g. for assembly and installation. Accordingly, some cost estimation approaches focus on material costs only whereas other take on a more holistic perspective.

Weight-based costing as one of the material-only approaches is one of the simpler forms of parametric modeling in combination with analogy-based estimation: The cost per produced unit is calculated as unit weight multiplied with a weight-cost ratio derived from historical data or a product analogy. The ratio can be either constant or degressive with increasing weight (Ehrlenspiel 2007a, pp. 432–433). This cost estimation approach only yields reasonable results when the new product is very similar in design to the analogy or historical data used and its approximate weight is known. Also, it appears unlikely that the cost of a complicated product can be reduced only to its weight. On the other hand, it is frequently used due to its simplicity (Wierda 1988, p. 191; Roy 2003, p. 4).

Cost estimation by a *material cost to manufacturing cost relation* is a similar approach to weight-based costing with the difference that instead of the product's weight its direct material costs are considered as the basic parameter of the cost function. The underlying assumption is that the relation of material costs to manufacturing costs is constant for specific product groups (Wierda 1988, p. 191). This approach can be used in conjunction with the bottom-up *bill of materials* (BOM), a detailed list of the required materials and components for manufacturing a product (Reid 2002, pp. 457–458; Sun 2007). Evidently, this is only feasible once the product design has been decided upon and therefore the use of a BOM is not suitable for early product development phases. Furthermore, this approach tends to neglect labor costs.

Estimating costs via d*esign equation* combines technical and economic parameters into one closed-form cost equation (Ehrlenspiel 2007a, pp. 434–435; VDI-Richtlinie 2225 Blatt 1; VDI-Richtlinie 2225 Blatt 4). Typically, this is realized by identifying a number of product property variables and quantifying their cost impact, e.g. the resolution of a camera or the energy output of an electric motor (Eversheim 1985, p. 417). The advantage of this approach is that it has the potential to capture relations

between product characteristics and their impact on costs more precisely. Its major disadvantage is the effort required to create adequate design equations. Each product category requires a separate equation and thus the establishment of techno-economic relations, search for and selection of crucial variables and evaluation of the resulting cost function. For complex products, the design function can be complicated and results difficult to convey (Wierda 1988, p. 191). Also, labor costs are usually not considered.

A method to systematically derive and quantify relations between product properties and costs is *regression analysis* of the considered variables, i.e. the product property parameters. Regression can be linear, non-linear or a combination of both (Ehrlenspiel 2007a, pp. 436–437). Regression analysis is a methodologically sound and approved technique from the field of statistics. Thus it offers the advantage of being highly systematic and less prone to user bias. It allows the calculation of statistical measures which can be used to assess the uncertainty inherent to the estimation. On the other hand, regression analysis relies on data and assumptions made to be reliable and relevant; if this is not the case it can lead to false relationships and erroneous conclusions (Cook 1982); (Roy 2003, p. 5). The efforts of data acquisition and processing are extensive which renders the application of this method costly and time-consuming.

Although it is often stated that expert judgment is a common tool in estimating the costs for product development (Rush 2001, p.272, p.275) empirical studies on the matter are sparse. Keeney states that expert assessments are basically inevitable and "a judgment is initially required in determining that a problem is even worthy of attention" (Keeney 1989, p. 83). The elicitation of expert opinions often represents the only possibility to gather information in early development stages (Rowe 2002, p. 125).

Delphi methods offer the possibility to reduce the effect of the aforementioned disadvantages of individual judgments by combining the opinions of several experts in a structured way. It is assumed that the estimates will converge after several rounds of revising the experts' assessments (Rowe 1999). This method has been employed for a multitude of application purposes and proven a valid approach to forecasting (Linstone 1975). One major disadvantage of delphi approaches in

contrast to other expert judgment elicitations is that they require a considerable effort for the preparation of expert rounds.

Another form of statistics-based cost estimation is the use of *neural networks*. Their main building blocks consist of layers of artificial neurons which represent functions relating input and output data. The neurons are linked by so-called activation functions defining the output of a neuron which is the input to neurons on the successive layer; technical literature displays a multitude of different ways of constructing a neural network (e.g. Lawrence 1994; Hertz 1991; Gurney 1997).

The benefits of neural networks are controversially discussed: Some authors claim cost estimation via neural networks to outperform other forecasting methods (e.g. Liu 2003; Li 2006) whereas others doubt that it is of practical use: "Although neural nets do solve a few toy problems, their powers of computation are so limited that I am surprised anyone takes them seriously as a general problem-solving tool" (Dewdney 1998, p. 82). The main advantage of neural networks is their skill of solution optimization through self-adaptation of the applied model and the handling of very complex and non-linear data. The drawbacks are the large requirement for computational power and expertise in the construction and training of the neural network. Whereas the former is arguably abated to a certain degree by the development of potent computers and the relative decline of related costs, the latter represents a major impediment: not only does the complex process of developing a neural network bind considerable manpower but it requires knowledge which might not be available to every company. Furthermore, the results of an estimation by neural network can be hard to interpret as the internal weighting and connections of hidden neurons – the intermediate layers – are concealed from the user (Ehrlenspiel 2007a, p. 442; Roy 2003, p. 8).

A range of integrated software solutions is available for hardware cost estimation (Trivailo 2012). PRICE H and TruePlanning TrueH by PRICE Solutions and SEER-H by Galorath Incorporated are comprehensive suites that rely on large databases filled with data from projects and expert estimates and widely use parametric modeling and product analogies. Although these applications stem from military and aerospace background they are not restricted to these areas but can be applied to a wide array of different industries (ibid., pp. 11–12). The Advanced Cost Estimating

System (aces) by 4cost GmbH is a module of the 4cost suite allowing detailed parametric cost estimation. In contrast to PRICE and SEER products it does not rely on databases but on mathematical cost estimation functions derived from research over many years (4cost 2012). These suites offer comfortable out-of-the-box solutions but still require a lot of input data for fine-tuning and are thus less adequate for early phase cost estimation (Trivailo 2012, p. 15). Also, they are not free of charge thus incurring additional estimation costs.

Another category of tools available for hardware cost estimation is represented by "government of the shelf (GOTS)" (ibid., p. 5) handbooks, notably originating from U.S.A. aerospace and military programs. Although these are primarily designed for the mentioned fields the underlying principles and general recommendations can be applied to other areas with complex products such as robotics.

Some guidelines and norms primarily addressing cost accounting and cost planning can also be applied to cost estimation. Financial reporting and management accounting standards provide guidelines on how to assess the value of existing products within the company. Planned costs are extrapolated from knowledge of productivity ratios and unit costs gathered in earlier production periods of the product (Plinke 2000, p. 161); from the product perspective this form of cost estimation can be regarded as an analogy-based top-down approach. As traditional cost planning requires detailed cost information it is best suited for mature products and less appropriate for innovation development.

All of the mentioned methods are valid approaches for hardware cost estimation. However, the different methodologies make it hard for the robot developer to decide on the appropriate one and in contrast to more established industries like aerospace or automotive there is no integrated suite of methods available for service robotics. Thus, this work proposes a combination of regression analysis for hardware material as well as expert opinion and analogy-based approach for labor cost estimation. Regression analysis allows the estimation of material cost distributions based on documented component characteristics and is unbiased by subjective assessments. Since little information on labor costs is available in contrast to well-documented hardware components (material) costs, the analogy-based

approach falls back on experts' experiences in building prototypical service robots as a grounded basis for estimation.

3.1.3. Software Cost Estimation

Expert judgment and analogy-based estimation techniques are the most widely used method categories for software development cost estimation (Jorgensen 2007, p. 40). The methods within these categories range from estimates based on intuition and experience over case study analogies to more formalized approaches like delphi methods (Molokken 2003, p. 5; Jorgensen 2007, p. 39; Fink 2005, pp. 30–31).

The roots of the *wideband delphi method* lie in the research field of cost estimates in software development (Boehm 1981). This method is widely used in the software industry and has proven to be effective for software projects (Stellman 2006, p. 39; Stochel 2011, p. 351). Although the wideband approach is supposed to increase communication and interaction between the experts through additional moderated meetings the terms "wideband delphi" and "delphi" seem to be used as synonyms.[5] Thus the explanations of advantages and drawbacks given for hardware cost estimation using delphi methods apply analogously.

A more recent approach to ameliorating bias effects by tapping into the knowledge of several people at once is the *wisdom of crowds approach*. According to Surowiecki this method improves estimation quality by reducing negative effects like peer pressure and bias through dominant personalities in the expert group (Surowiecki 2005). Its basic claim is that the behavior of a group can be described as the behavior of a single entity potentially revealing higher intelligence than even the smartest group member thus leading to better estimates. For this to become true the expert group members forming the so-called 'wise crowd' must be heterogeneous, independent, decentralized and have a means of coalescing their opinions into one decision. Though the wisdom of crowd approach is not specifically

[5]Stellman and Greene state that "The Wideband Delphi estimation method was developed in the 1940s at the Rand Corporation" but the expression "wideband delphi" originated in (Boehm 1981) which derives from the delphi method developed by the RAND corporation as a forecasting tool, s. (RAND Corp. 2010) for numerous articles on the topic.

aimed at cost estimation, Stochel reports this method to be superior to wideband delphi cost estimates for software projects (Stochel 2011, p. 538). Similar to delphi methods it is a more costly approach than single expert estimation because it requires substantial efforts of preparation. Furthermore, it is difficult to assess if a group of experts is a wise crowd or just an opinionated mob.

Analogy-based cost and effort estimation for software development is also a very common method for software development cost estimation (Mair 2005, p. 510). It has been found to be suited for early cost estimation phases when little detail information is available. Some approaches use the algorithmic determination of similarity measures to ensure comparability of analogon and estimation subject but these measures are disputed and introduce further inconveniences to the estimation process (Shepperd 1997, p. 738).

Although expert judgment and analogy-based estimation appear to be the most favored choice in practice, companies sometimes require stronger formalization for cost estimation. Yenduri et al. analyzed the results of a company's switching from expert judgment to parametric modeling, reporting higher accuracy of estimates after the change (Yenduri 2007). The majority of the more formalized approaches focus on algorithmic models (Shepperd 1997, p. 736).

Most of the parametric models are based on software size measures (Boehm 2000; Ma 2010, p. 1). The dominating metrics are *lines of code* (LOC), *function points* (FP), and *object points* (OP); the derived values are subsequently translated into cost or required effort quantifications (Leung 2002, pp. 309–311; Zia 2011, pp. 104–105).

Lines of code, delivered source instructions or source code statements can be calculated by counting the lines of meaningful code and omitting non-functional parts, e.g. comments or empty lines. The appropriate standard of what is to be considered meaningful code must be defined before LOC counting (Nguyen 2007, pp. 2–5; Kalb 1990-10-01).

The usage of LOC is a controversially discussed matter among software scientists (Touesnard 2004, pp. 1–10). Major advantages of LOC counting are its intuitive comprehensibility and easy automation of the counting process. On the other hand, its explanatory power is limited because LOC depend on further variables like the

code quality depending on the developers' experience, coding style, difference between coding languages and many more.[6]

The *function point metric* was introduced by Allan Albrecht as an alternative to LOC (Albrecht 1979; Albrecht 1983). It abstracts from physical code size which varies greatly depending on the applied programming language by capturing the code's functionality. The first step in function point analysis is to determine the number of unadjusted function points (UFP) by classifying software functions into the categories outputs, inputs, inquiries, internal files and external interfaces and consequently weighting each function depending on its complexity (Glinz 2004, pp. 56–57). The second step is to calculate the adjusted function points (AFP) by using the formula

$$AFP = VAF \cdot UFP \tag{3.1}$$

where VAF, the value adjustment factor is calculated by

$$VAF = 0.65 + 0.01 \cdot TDI \tag{3.2}$$

TDI stands for total degree of influence and can range from 0 to 70; it is derived by summing impact weights from 0 to 5 for 14 different system characteristics (ibid., p. 57).

Different variations of the function point concept exist; as of 2012 there are five recognized standards (Wikipedia 2006); also there are some extended versions like *feature points* or *full function points* (Leung 2002, pp. 311–312). Advantages over LOC are the metric's detachment from specific coding languages and focus on software functionality instead mere physical size. One major drawback is that function point analysis requires deeper understanding of the analyzed software than LOC and thus must be executed manually by trained experts rendering the code analysis more time-consuming and costly.

Object points estimation is the most recent of the mentioned metrics, the concept was presented by Banker et al. in the early 1990s (Banker 1991). Object point methods are based on an object-oriented software development perspective and depart from traditional sequential programming paradigms; it is claimed that LOC

[6]Discussing LOC in depth is beyond the scope of this work. For a comparison of advantages and downsides see e.g. (Sneed 2010, pp. 229–262; Wikipedia 2003).

and FP metrics do not accommodate well the characteristics of object oriented programming (Chidamber 1994, p. 477; Minkiewicz 1997, pp. 2–3). Object points are calculated by counting program windows, reports, third generation language (3GL) modules and rules and weighting them (Stensrud 1998, pp. 2–5). Object points are similarly correlated with development effort as function points but are allegedly less complicated to count which is why they are particularly suited for early design phases (McConnell 2006, p. 86); nevertheless a basic idea of the software layout must be available in order to calculate the object point estimate.

No matter which size metric is applied, parametric models are used to derive effort estimates from software size, commonly expressed in person time and currency units. Additionally, many models imply fine-tuning parameters varying in type and number from model to model. Macro-adjustments aim at creating a single multiplier based on adjustment parameters like skill level, programming tools and paradigm. This factor is then applied to the overall project estimate (Jones 2007, p. 336). Micro-adjustments also rely on different adjustment parameters but usually apply each derived factor separately.

McConnell reports 21 adjustment factors used in Boehm's COCOMO II model with impact ranges indicating the potential to decrease or increase development effort; the ten factors with the highest impacts according to Boehm's research are listed in Table 3.1-2 (McConnell 2006, p. 66). In a similar vein, Jones enumerates 25 similar factors having potential impact on software productivity (Jones 2007, pp. 342–343).

Development Factor	**Potential effort decrease, %**	**Potential effort increase, %**
Product complexity	-27	74
Requirement analyst capability	-29	42
Programmer capability	-24	34
Time constraint	0	63
Personnel continuity	-19	29
Multi-site development	-22	22
Required software reliability	-18	26
Extent of documentation	-19	23
Business area experience	-19	22
Use of software tools	-22	17

Table 3.1-2 Factors influencing software development effort based on COCOMO II

Applying concrete, company-specific values for calibrating an estimation model can significantly increase the estimation fidelity but the problem with most adjustment factors is that they introduce a level of subjectivity, especially if the parameter values are difficult to express quantitatively, e.g. programmer experience and thus they can create a deceptive impression of accuracy (McConnell 2006, pp. 47–49).

Although a large number of cost models exist, many of them are proprietary and thus cannot be assessed or compared (Boehm 2000, p. 178). However, a few parametric models have been received with critical acclaim, Putnam's Software Life Cycle Management (SLIM) model and the different versions of the already mentioned Constructive Cost Model (COCOMO) by Boehm being among the most popular (Leung 2002, pp. 314–315).

Central part of Putnam's model is the so-called software equation

$$S_s = C_k K^{1/3} t_d^{4/3} \tag{3.3}$$

where S_s is the total software size output in LOC, K the development effort in person years, t_d the software delivery time and C_k the environment or technology factor which can be derived from historical data (Putnam 1978, p. 355). The model is derived from the Norden-Rayleigh curve describing manpower as a function of project time and captures project manning changes over the lifecycle of a software project. Originally aimed at estimating the size of software it can also be used for effort and schedule estimation (Leung 2002, pp. 314–316). Although the model is based on solid empirical evidence it has been observed that some of the assumptions do not always hold in practice (Boehm 2000, p. 180).

The COCOMO models also have a power function as the basic regression formula at the center of their concept:

$$E = aS^b \tag{3.4}$$

where E is the development effort in person months, S is the expected software size in thousand lines of code (KLOC) and a and b are coefficients whose values are derived for different types of software project characteristics (Leung 2002, p. 314). The COCOMO family comprises different versions of Boehm's cost estimating model: The first version COCOMO 81 was published 1981 and became one of most widely-used parametric cost estimation models (Boehm 1981; Boehm 2000, p. 202). It is

subdivided into basic, intermediate, detailed COCOMO, each suggesting different values for the coefficients a and b.

As software development paradigms and processes changed significantly the model grew less applicable to modern software projects which is why its creators realized the need for an overhaul (CSSE 2011). The new version, dubbed COCOMO II, aims at addressing issues like software reuse, object-oriented programming introducing scale-factors for precedentedness, development flexibility, architecture and risk resolution, team cohesion and process maturity. It is also subdivided into three parts called Applications Composition, Early Design and Post-Architecture, each addressing different stages of the development process. Although COCOMO II is widely-used in practice critics expound the problems of high sensitivity and requirement of concurrent approaches (Musilek 2002; Oriogun 1999, p. 61).

A wide range of software tools is available with many differences regarding their characteristics. Some programs are free of charge for personal use; most are marketed on a one-time or periodic fee basis. Furthermore, there are tools related to one specific estimation method and those that cover a whole spectrum of approaches. Another differentiation criterion is their degree of integration: some tools are stand-alone programs and others can be integrated into software management suites. Table 3.1-3 exhibits examples of estimation software along with their characteristics classification.

Software	**Distributor**	**Free version available**	**Method-specific**	**Suite integration**
SEER-SEM	Galorath	No	No	Yes
4cost-aces	4cost	No	Yes	Yes
QUEST	ProjectExperts	No	No	No
Construx	Construx Software	Yes	No	No
COCOMO II	University of Southern California	Yes	Yes	No
ArchANGEL	Bournemouth	Yes	Yes	No

Table 3.1-3 Examples of software estimation programs

SEER-SEM is a comprehensive suite of estimation methods and models, knowledge bases and visual data representation possibilities; it is widely used in aerospace and defense industries (CSSE 2002). 4cost-aces is a module based on parametric estimation that does not rely on a knowledge database but on cost estimation relationships developed by expert engineers (4cost 2012). SEER-SEM and 4cost-aces are examples of larger management suites that go beyond software cost estimation but also address issues of life cycle costs, hardware costs and project management in general.

QUEST and Construx Estimate are both exemplary applications tailored specifically for the estimation of software project efforts and support a variety of estimation techniques (CSB 2012). Whereas QUEST appears to be relying more on questionnaire techniques for elicitation of expert assessments, Construx Estimate has a stronger focus on parametric estimation techniques.

COCOMO II and ArchANGEL are spin-offs from university research centers and can be used free of charge. COCOMO II is based on the popular model of the same name described above (CSSE 2011); ArchANGEL is the further development of ANGEL Plus and is based on analogy estimation techniques (Shepperd 2002).

Each type of methodology has its advantages and drawbacks depicted in Table 3.1-4. Reproducibility refers to how well similar results can be obtained if another expert were to repeat the estimation process, traceability refers to how well the results can be understood, interpreted and traced back, and the required expertise indicates the necessary knowledge of estimation method and experience in estimating.

	Expert judgment	**Parametric models**	**Analogy-based estimation**
Reproducibility	Very low to high	High	Medium
Traceability	Medium	Medium	High
Required estimation expertise	Low to medium	High	Medium

Table 3.1-4 Assessment of estimation methods

The estimation quality of expert methods highly depends on the knowledge of the estimating experts. If this inherent information is not rendered explicit the results may lack reproducibility and traceability.

Parametric models yield reproducible and traceable results due to their high degree of systematization. As they are based on long-term statistics and heuristics, the models provide plausible estimation values. However, most parametric models require in-depth knowledge from previous projects on model-relevant factors like productivity or development time (as in SLIM estimation) for calibration. These might not be readily available for a new product type like a service robot in contrast to more established industries e.g. automotive or aerospace.

Analogy-based estimations permit a faster, arguably less accurate prognosis but also require comparable software. The quality of the results strongly depends on the comparability of the analogon used for comparison. As they tend to be less systematized than parametric models the expert's bias may have stronger impact on the estimation results.

In order to strike a reasonable compromise between estimation effort, usability and accuracy this work proposes a combination of parametric and analogy-based modeling. By referring to existing robot software as an analogy and using approved development heuristics, the lack of data from previous projects is abated. The use of language-specific productivity value permits more accurate estimations than a purely LOC-based approach.

3.2. Product Design Methods

Although the designing process for a service robot is not the focus of this thesis at least some notions of its principal design must be available in order to estimate the costs of developing it. The structure of a complex product is its primary cost determinant and depends on its purpose. Therefore, current methods of the derivation of product structures from the intended product application are presented in this section, i.e. the transition from functional requirements to the product structure. The different methods can often be combined in the search for product design ideas thus complementing each other.

Product descriptions can vary in their degree of specification from rough product sketch to minutely detailed designs. As the aim of the presented approach is prototype cost estimation in early development stages only those methods that are adequate for conceiving approximate product structures are presented here; detail engineering usually takes place at later development phases.

3.2.1. Intuitive Methods

Intuitive methods for generating design ideas are commonly applied to problems with imprecise structures, i.e. problems that cannot be solved by algorithms due to their openness (VDI-Richtlinie 2221, p. 34). The related methods are usually group creativity techniques aimed at finding innovative solution ideas. When applied to the product design process in early planning and development phases these methods are adequate for generating concept ideas without detailed engineering.

The most widely used intuitive method is *Brainstorming* (Hender 2001, p. 1). It is usually set up in small groups of five to 15 people who ideally stem from different disciplines with the aim of gathering as many ideas for solving a given problem as possible (Ehrlenspiel 2007a, p. 60). The technique is based on two guiding principles, namely the principle of deferred idea evaluation or judgment and the principle of striving for large quantities of ideas (Osborn 1963, p. 124). The former means that ideas are not assessed during the Brainstorming session in order to avoid group pressure and inhibition, the latter postulates that a great amount of ideas increases the probability of arriving at an excellent solution. Brainstorming is recommended for development phases where structures of product functions and thereto related solutions need to be generated (VDI-Richtlinie 2221, p. 34).

The *6-3-5 method* developed by Rohrbach is a derivation of Brainstorming also referred to as 'Brainwriting' (Rohrbach 1969). After communicating the problem to group of six experts, each of these experts is to produce three problem related ideas. The ideas are passed on to the next expert who elaborates on them. This passing-on takes place five times so that every expert has worked on every idea. This method is particularly suited for function and solution structuring but can also be applied to detailing design solutions.

Synectics is a technique that attempts to tap into subconscious thought processes by addressing seemingly irrational or emotional aspects of a problem and connecting different and apparently irrelevant elements (Gordon 1961, p. 3). Its core principle is to "trust things that are alien, and alienate things that are trusted" (ibid.) i.e. the detachment of known solutions and ways of thinking and embracing new perspectives. This approach is implemented by finding analogies for technical solutions to a given problem outside and consequently looking for possibilities to fit these analogies to the original problem.

3.2.2. Structured Methods

Whereas the intuitive methods are very difficult to integrate into a systematically computer-supported framework, the structured methods lend themselves more readily to support by automated spreadsheets, matrix representations and similar means due to their inherent systematization. As this allows the deployment of cost-estimating software these methods are arguably more suitable to the presented approach which aims at assisting and facilitating early phase cost estimation by providing a structured knowledge framework.

The *Morphological Analysis* or *General Morphological Analysis* is a method that attempts to investigate all possible solutions to a multi-dimensional non-quantifiable complex problem in order to find the most adequate one by means of comparison (Ritchey 1998, p. 1). The central concept is the use of a so-called morphological matrix or Zwicky-Box (Ehrlenspiel 2007a, p. 61; Ritchey 1998, p. 3). This matrix is constructed by setting different product aspects against each other along usually two or three dimensions, e.g. by mapping required product subfunctions to components as possible subsolutions or by mapping product features to possible implementation forms as depicted in Figure 3.2-1 for the schematic example of designing a lamp.

	Solution	**Solution 2**	**Solution 3**	…
Power	Battery	Generator	Photovoltaic	…
Light	High	Medium	Low	…
Size	Small	Medium	Large	…
⋮	⋮	⋮	⋮	⋮

Figure 3.2-1 Schematic example of a morphological matrix

After construction, the matrix is checked for internally consistent combinations in order to reduce the set of theoretically possible configurations which is referred to as "cross-consistency assessment" (Ritchey 1998, p. 6). The remaining arrangements represent plausible solutions and can serve as a basis for initial product design decision making e.g. by marking a preferred arrangement as indicated in Figure 3.2-1. Morphological methods have been applied to a variety of problems and found to be a useful approach to solution finding (Levin 2012).

Design structure matrices (DSM) can be used for modeling system and are particularly well suited for decomposition and integration purposes (Browning 2001, p. 292). The matrix approach can provide information on the relations between different components of a system but the concept is not restricted to the component domain; functions, processes, human resources adherent to the development and production of a system can also be dimensions of a structure matrix (Braun 2007; Browning 2001, p. 293). The dependency between domain elements presented in row and column heads are marked in the matrix cells i.e. the intersections of rows and columns. The dependency types allowed in the matrix cells must be determined beforehand. Relations can exist within a domain and between domains. The former are captured by intradomain matrices, the latter by interdomain matrices; both can be aggregated in a so-called Multi-Domain Matrix (Maurer 2008).

Intradomain matrices are square and also symmetric if only symmetric relations (the cell entries) are allowed. The cells on the diagonal left blank; self-reflexive dependencies could be noted there but this is atypical for the use of design structure matrices (Maurer 2007, p. 77). Figure 3.2-2 exhibits an intradomain matrix with a symmetrical relation between items A and B.

		Domain 1		
		A	**B**	**C**
Domain 1	**A**	-	x	
	B	x	-	
	C			-

Figure 3.2-2 Exemplary intradomain matrix

		Domain 2			
		Φ	**X**	**Ψ**	**Ω**
Domain 1	**A**	x			
	B			x	
	C				

Figure 3.2-3 Exemplary interdomain matrix

Interdomain matrices are not symmetric as the number and type of items in row and column heads can differ (s. Figure 3.2-3).

The planning matrix of the *Quality Function Deployment* (the 'roof ' of the *House of Quality*) where customer requirements and product features are correlated is a related concept and can be interpreted as a DSM, too (Browning 2001, p. 293; Clausing 1994, pp. 94–96).

Axiomatic Design (AD) is a systematic method aimed at rendering design processes more creative and efficient at the same time by providing rational thought processes and tools (Suh 2001, p. 5). The basic idea is to map different four domains of the designing process in a successive. The transitional mapping between the four domains is depicted in Figure 3.2-4 (ibid., p. 11).

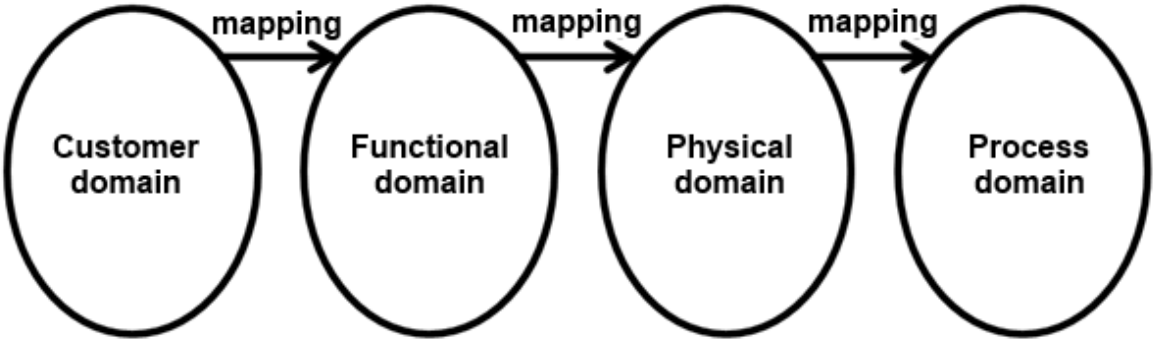

Figure 3.2-4 Mapping of design domains

The customer domain describes the desired features of a product from the customer's perspective; in the functional domain these descriptions are translated into functional requirements and constraints. The physical domain comprises the design parameters with which the requirements are to be met. The process concerns the producing process and the pertinent variables (ibid., p. 10).

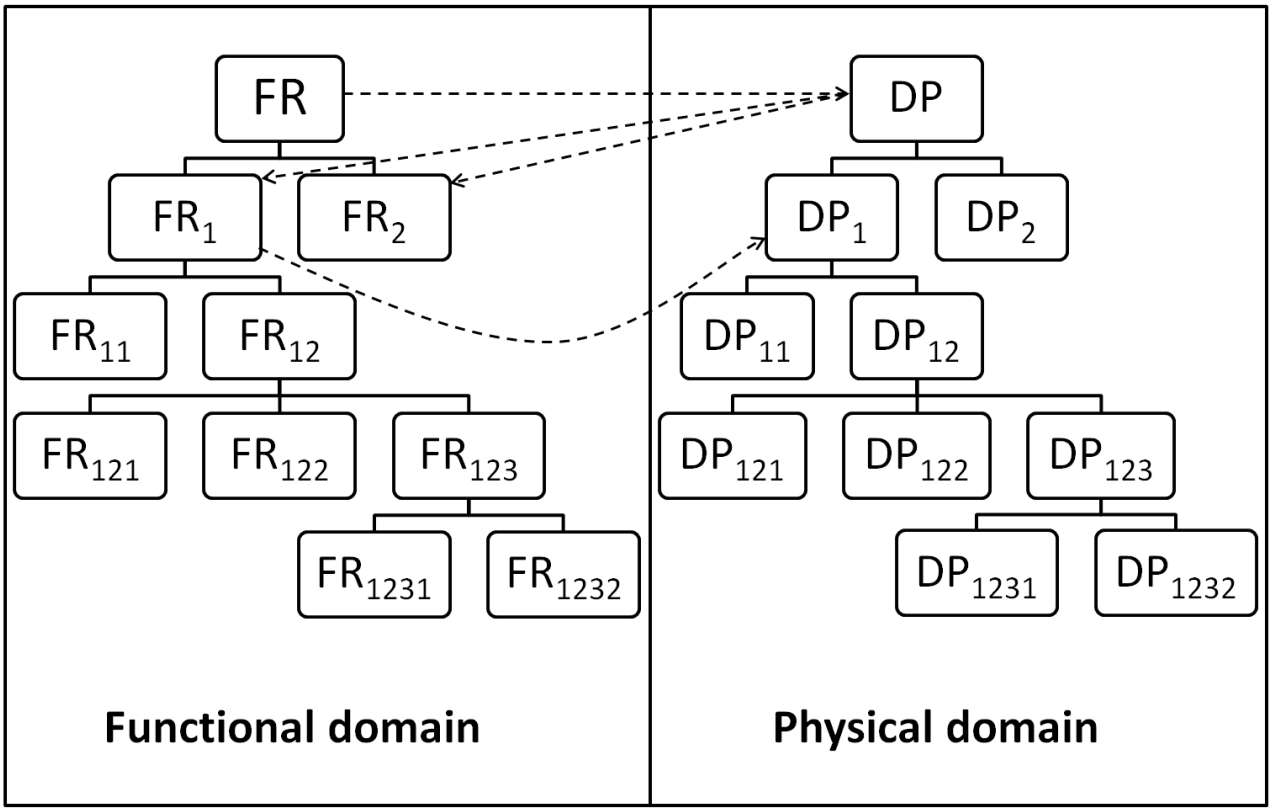

Figure 3.2-5 Zigzagging decomposition in AD

The main focus of the approach lies on the functional and physical domain and the mapping between them. In order to proceed systematically two axioms are applied. The independence axioms postulates functional requirements to be formulated as a minimum set of independent requirements i.e. decoupling of the product functions and subsequently mapping each one of them to a design parameter (ibid., pp. 16–17). Top-down decomposition of the functional requirement and the respective mapping of design parameters – a process dubbed 'Zigzagging' by its creator – leads to more a detailed product structure; an example is depicted in Figure 3.2-5 (ibid., pp. 29–32). The information axiom stipulates that the design with the least information content be chosen i.e. the simplest design fulfilling the requirements; however, most of the applications of AD focus on the first axiom (Kulak 2010, pp. 6707–6710).

As a conclusion it can be stated that most of the systematic approaches to determining a product structure are based on matching desired product functionality to product features and top-down structure decomposition. This commonality appears logical and reasonable because little more than the product's

purpose is known at the initial phase of the development process and a sensible product design must be aligned with the pertinent requirements.

3.3. Method application to Service Robotics

Although a variety of suitable methods is available for systematic cost estimation only few of them have been applied to the field of service robotics as of 2012. Noticeably, the majority of mentions of cost aspects within the context of service robots relate to low-cost designs where 'low-cost' is employed in relatively vague terms. However, many claims of such a design are not substantiated by any cost data; in most cases where cost data is given no attempt is made to estimate the costs of a service robot before construction but the resulting material costs are listed with hindsight. A possible explanation for the marginal treatment of cost issues could be that most of the works mentioning costs in relation to service robots focus stronger on technical than economic innovations, e.g. the usage of superior algorithms in order to compensate for inferior i.e. 'low-cost' sensor performance.

Shue et al. describe the construction of a "low cost semi-autonomous sentry robot" and refer to the usage of low cost components as the means to reduce costs but give no detail on any incurred costs (Shue 2012). Psarros et al. present "the design and development of a semi-autonomous low-cost underwater service robot" using conventional materials and components but provide no cost data whatsoever (Psarros 2009). Similarly, Zhou et al. illustrate the "design and implementation of [a] low-cost service robot" and justify the expression with the usage of standardized and inexpensive components; however only total acquisition costs are given without indication of their composition (Zhou 2010). Han et al. discuss "an efficient and low-cost robot grasping system" and report to have reduced the pertinent cost by using less accurate and less inexpensive components but only total costs are stated and compared with price estimates for selected robots, e.g. Willow Garage's PR2 (Han 2011).

A number of publications with a more detailed treatise of costs can be classified as bill of materials (BOM) approaches where costs are implicitly defined as the costs of the physical components of the robot. Mundhenk et al. describe a "low cost, high performance robot design utilizing off-the-shelf parts" and list the costs for the

components of a prototypical robot as depicted in Table 3.3-1 (Mundhenk 2003, p. 296).

Although software design is outlined no resulting costs for its development are mentioned. Likewise, Wolfe et al. outline the development of "a low-cost independently-mobile reconfigurable modular robot" and indicate the detailed component costs (Wolfe 2012). Labor costs are explicitly excluded. Piperidis et al. present "a low cost modular robot vehicle design" and specify a price lower than 1,000 € as the target price for a simple educational and research robot (Piperidis 2007). Only material costs are detailed for the estimation of the robot price; software development is addressed without reference to the resulting costs. Another example of low costs being claimed on basis of component costs is the "low cost indoor mobile robot localization system" by Lopez et al. where the costs for localization device and marker tags are stated but no further elaboration of development costs is given (Lopez 2011).

The existing software effort estimation techniques are applicable to the service robot software as it is subject to the same basic principles of software design. However, cost estimation for service robot software development is neglected in the aforementioned approaches and appears to be disregarded in general. Although the importance of software development within the field of service robotics has become evident only few explicit considerations of the adherent costs have been made (e.g. Kim 2005; Kim 2009). Kang et al. point out that virtual prototyping can significantly reduce development costs but limit themselves to this claim without further elaboration (Kang 2005).

The profitability study for innovative service robot concepts (EFFIROB) is explicitly aimed at estimating life cycle costs for service robots from the user's perspective (Hägele 2010, pp. 46–64). The scope of this study comprises methods for the estimation of both hardware and software costs. Hardware cost estimates are point estimates and based on component costs which is why this method can be classified as a simplified BOM approach.[7] The individual component costs for a selected range

[7]Simplified because only the main components are considered as no detail design was available.

of component categories are listed in a component catalogue. Software costs are given as point estimates based on code size of those ROS stacks which the estimating expert deemed comparable to expected software development; consequently, this estimation method is analogy-based. Although the point estimating methods yield no information on the likelihood of the estimate or the margin of error the EFFIROB exhibits the most encompassing cost estimating framework for service robots so far.

Part	**Quantity**	**Cost per unit (USD)**	**Total Cost (USD)**
Rocky 3742 Motherboard	2	500	1,000
512 Mb Memory (PC 133)	2	36	72
Flashram (256 Mb)	2	47	94
Unibrain Fire-I Camera	1	100	100
Notebook Hard Drive (40 GB)	2	107	214
Traxxas E-Maxx	1	369	369
1 GHz Pentium III CPU	4	92	368
SONY NP-F960 7.2 V 38.8 Wh	8	129	1,032
Power Supply (TI Power Trends + Parts)	1	100	100
Extra Parts, misc	1	200	500
TOTAL			3,849

Table 3.3-1 Example of a bill of materials

In this light, it is deemed necessary to compile a comprehensive and specific methodology for early cost estimation in order to facilitate the development of cost-efficient service robots. The approach thus proposed in this work is presented in the following chapter.

4. Function-based Cost Estimation Approach

Regarding the areas of hardware cost estimation and software cost estimation separately, a variety of theoretically applicable methods are available as outlined in Chapter 3. Although these methods have been applied in a wide array of industries, only very few comprehensive cost assessments can be observed in the field of service robotics. Studies addressing service robot cost issues often focus on material or hardware component costs for low-cost solutions; software costs seem to be widely neglected. Additionally, most of these approaches analyze costs from the retrospective and thus cannot be regarded as estimation methods.

In terms of holistic approaches that try to estimate both hardware and software costs the lack of integrative methodologies applied to service robots is even more evident: The only study that addresses both aspects in relation to service robotics within a cost estimating framework is EFFIROB.

While the EFFIROB study poses a valuable contribution, the approach applied therein has several shortcomings:

1. The cost estimates for service robots considered are point estimates i.e. single values. However, point estimates do not convey any information on the uncertainty of the estimation or its margin of deviation. Range or corridor estimates, however, permit the intuitive assessment of uncertainty by the width of the pertinent interval – large intervals reveal a higher probability of error.
2. The hardware component costs provided are based on singular expert judgment. In addition to the problem that these are point estimates there is no consideration of cost variation as a function of different component parameters e.g. weight, payload etc.
3. There is no systematic link between desired functions and the related selection of components which renders the hardware configuration subjective to personal preferences and may lead to neglecting possible alternative design solutions.
4. The software cost estimation is based on an average reuse ratio and does not account for different reuse ratios among software components. This leads to strong bias especially for software parts which only require installation but no

modification. Additionally, the basis of the software analogy, ROS, has undergone substantial changes since 2009 which suggest an updating of the estimation basis i.e. the code sizes of ROS components.

On the other hand, the general procedural method of decomposing a robot's functionality, mapping the corresponding functions to hardware and software components and aggregating resultant costs is a valid and expedient approach which is corroborated by the fact that the German Federal Ministry of Education and Research declared profitability analyses based on the EFFIROB approach as compulsory for service robot related project proposals (BMBF 2012).

Therefore, this work uses the EFFIROB study as a methodological starting point and extends it in several ways as to eliminate or ameliorate mentioned shortcomings. These enhancements comprise the following methods:

1. Structural decomposition is realized with structure matrices derived in dialogue with service robot experts and current classifications of robot functionalities. The relations thus captured permit systematic mapping of robot functionality to hardware and software components and provide the estimating person with suggestions for least requirements as well as possible enhancement and substitution possibilities
2. Hardware component costs are estimated by parametric cost models based on regression analysis thus avoiding subjectivity.
3. Software costs are estimated using an analogy-based approach. Data on the chosen analogy, the Electric Emys[8] release of the Robot Operating System by Willow Garage, is extracted by analyzing package-wise code size and reuse ratios of 469 packages in 89 stacks.

[8]As of April 2012.

4. For all cost constituents, uncertainty is indicated by estimating probability density functions and thereby determining range estimates. Unknown underlying shapes of probability density functions are modeled using the three-point estimation technique from Program Evaluation and Review Technique (PERT). This technique is considered a best practice approach for addressing cost uncertainty (Galway 2007, p. 9).
5. By applying established costing theories different cost types are considered. The parallel utilization of direct costing and burdened costing grants a more comprehensive perspective on the costs estimated.
6. In order to facilitate the estimation process a flexible and extendible software application is implemented which integrates the different cost models and provides the estimating person with default values for parameterization and numerous possibilities to enter individual values.

4.1. The Estimation Process

The aim of the presented approach is to enable the robot developer to arrive at a reasonable cost estimate for a prototype from an initial point at which an approximate idea of the desired functionalities is available but no detailed design of the system. By following a sequence of software-supported steps the estimation process can be facilitated and accelerates as the requirement for substantial background knowledge on behalf of the estimating person is substantially eased. An overview of the three main steps is given in Figure 4.1-1.

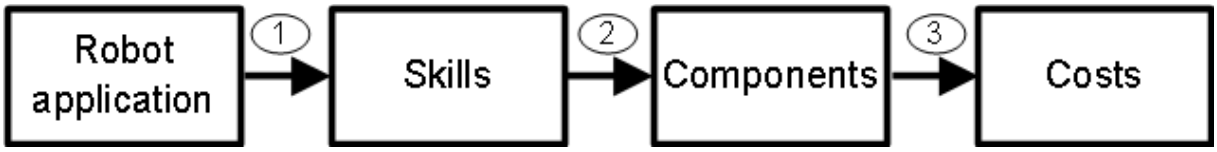

Figure 4.1-1 From robot application to costs

These steps can also be interpreted as traversing several conceptual domains of robot design similar to Suh's axiomatic design domain model (s. Chapter 3.2.2).

The estimation process starts with deriving required specific functions or skills from the known application purpose i.e. the developers have to decide what actions the robot is supposed to perform. The estimator must map these specific functions to

the generic skills used in the presented approach (s. Chapter 5.1.2.1) if the software tool is to derive suggestions for a conceptual technological structure from skill requirements. Alternatively, the estimator can also directly indicate components that make up the robot structure. Finally, the costs for the robot's development based on the structure of the previous step are estimated using heuristic cost functions. Figure 4.1-2 illustrates the cost estimation process as it is supported by the implemented software tool functionality (s. Chapter 7).

Initially, the developer of the robot must decide what actions the robot should perform, i.e. the specific robot application must be known. Once the purpose of the robot is decided upon, the application is broken down into tasks which directly correspond to specific functions.[9] This can be achieved by applying various creativity techniques (s. Chapter 3.2.1). Since the number of possible applications and specific function combinations is virtually infinite structured methods are deemed less adequate at this stage which is why the estimation software tool does not support this step.

The specific skills must then be mapped to the ones suggested in the skills catalog of the presented approach. These skills represent a non-exhaustive, generic set of typical robotics skills that can be found in most service robots (s. Chapter 5.1.2.1 for details on skills). Depending on the desired functionality the estimator selects those skills demanded of the robot as the input to the derivation of the technological structure. The transition from specific to generic robot skills is the entry point for the usage of the implemented software tool and marked by the dashed horizontal line in Figure 4.1-2.

From the selection of skills the set of necessary hardware and software components can be deduced by using knowledge on structural dependencies between these categories. These dependencies are extracted from the database where this knowledge based on expert information has been stored a priori (s. Chapter 5.2). After checking for interconnections, two separate lists of hardware and software components are presented to the estimator. Each of these lists contains suggestions

[9]An application represents a general field of work, tasks are more specific duties within this field, s. (ISO 8373, p. 17).

for components required and potential substitutes as well as components that can enhance the robot's performance. The effect of relating items of the categories skills, hardware component and software components via stored structural dependencies is supporting the estimator's task of creating an approximate robot design on which cost estimates can be based. Also, potential alternative solutions and possibilities for variants are indicated.

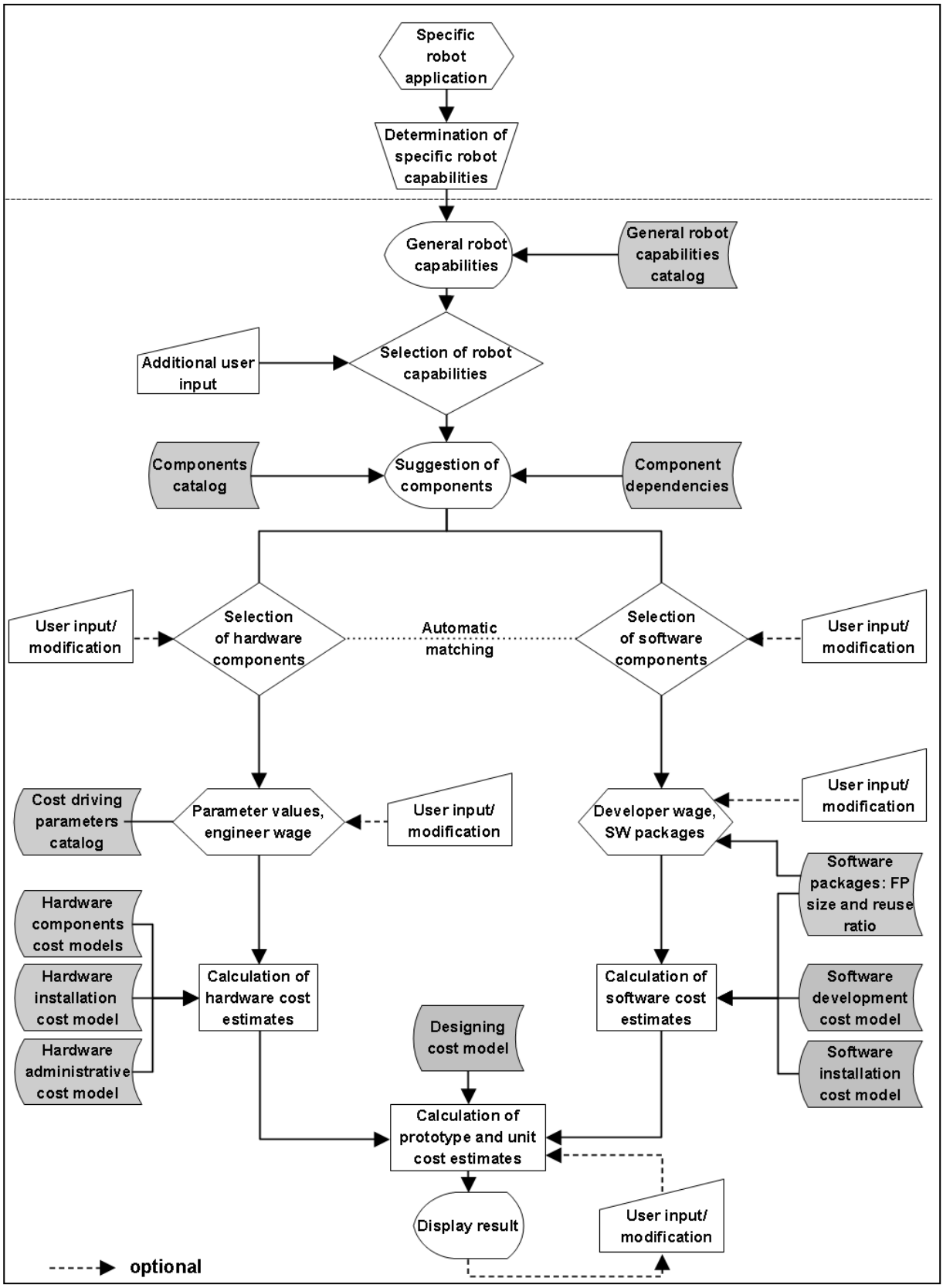

Figure 4.1-2 Service robot cost estimation approach

The selection of components serves as the base of cost estimation for the robot as each component has been assigned a specific cost function of its main cost driving parameters which are stored in the database along with typical average values of these parameters. By modifying the parameter values the estimate is fitted to the concrete robot solution without forcing the estimating person to develop individual cost models. The database also holds default parameter values in order to provide orientation to the estimating person in case no information on the specific parameter is available; if the default settings are used the averaging problem applies i.e. averages are usually not as exact as detail specifications for a concrete product or component. These defaults are based on expert knowledge and statistics on current market data.

In addition to component specific costs the costs of the system designing process which are not tied directly to components but rather the robot development on the whole are estimated by employing a proportional cost model.

Aggregating the previously calculated cost constituents, the cost estimate for the prototypical development of the robot is calculated based on the selections, pertinent cost models and additional input from the estimating person. In addition to prototype costs, direct and burdened unit costs are also derived.

4.2. From Functions to Structure

It is the declared aim of the presented work to facilitate the estimation of costs for robot development with only an early concept of the robot's desired behavior at hand. In order to avoid the necessity for detailed engineering in early concept phases generic knowledge on structural relations between skills and components - the main constituents of a service robot design in this approach - must be made available to the estimating person.

As developing service robots is a complex field of overlapping disciplines the extensiveness of knowledge required for the pertinent activities is vast. For knowledge essential to the estimation of the respective development costs to be systematically applicable it must be structured. In its structured form it can be rendered available to computer-aided processing thus formalizing and facilitating the estimation process. Structural knowledge in the context of the presented

approach is the knowledge about interdependencies between parts of and their relation to the overall structure, i.e. the service robot.

The determination of a product's structure i.e. product design engineering is a discipline of its own and cannot be covered to its full extent in the scope of this thesis.[10] The objective is not the derivation of detailed design or the qualitative assessment of a structure but to arrive at an information level where main cost-driving blocks can be identified.

The typical approach to modeling a complex system is to

1. decompose the system into subsystems,
2. analyze the (internal) relationships between subsystems, and
3. analyze the (external) relationships with its environment (Browning 2001, p. 292).

The decomposition of a complex system and its internal and external relations must be restricted to a level of granularity that is sufficient for the purpose at hand as the coverage of all details is too overwhelming or even counter-productive for human individuals and probably not feasible in early design phases (Maurer 2006; Maurer 2007, pp. 69–71; Browning 2001, p. 302). The differentiation between system and its environment also requires "choosing an arbitrary boundary for the system of interest" (Browning 2001, p. 292). The operating environment can be considered part of the system (e.g. Dalgaard 2010, pp. 10–13); however, in the context of the presented work the robot system is restricted to the mobile unit as outlined in Chapter 2.1.1.

Matrix-based approaches are among the most commonly used techniques for structuring knowledge on complex systems and relations (Maurer 2007, pp. 53–54; Browning 2002, pp. 429–430). Because they are well-suited for the illustration of dependencies between components and also complementary to computer processing matrices are used in the presented work for deriving a component-based

[10]For a detailed discussion of product engineering cf. e.g. (Lindemann 2005; Kossiakoff 2003).

architecture (Browning 2001, p. 292). Details on the structure matrix layout used in this work are presented in Chapter 5.

4.3. From Structure to Cost

With the help of structure matrices the user can determine what kind of components are required for the planned service robot but in order to derive cost estimates more information on the details of the component format are necessary. For this reason cost models are developed for each component category. Parameter values can be declared for each component thus refining the cost estimate; the overall costs for prototype and robot units are calculated as different aggregates of the component cost estimates plus the cost estimate for system design activities i.e. costs which cannot be attributed to a single component but rather the system as a whole. The latter cost constituent is modeled based on average industry data and several assumptions as data availability concerning this particular type of costs is sparse.

Hardware component material costs are based on regression models derived from technical data and prices of components currently available on the market. For each component, up to three main cost driving parameters are identified by comparing different regression models. Labor costs are considered based on average installation effort per component as indicated by robot experts and administrative efforts tied to each component instance.

The cost estimation for software components follows a different approach because there are no software suites that can be used in service robots without major adaptation and thus the dominant cost constituent is the development effort for robot software. These expenses are estimated by analyzing current robot software – the software analogy – related to each software component category and inferring development effort from their size and reusability. Heuristic values for development productivity are based on literature and expert opinions. By selecting packages from the analyzed software analogy that relate to the software component types of the structure matrix the software estimation can be refined to the specific planned service robot.

4.4. Data Collection and Processing Methods

The knowledge on service robot structures as described in this chapter must be based on data systematically retrieved and verified – the grey areas in Figure 4.1-2 mark those elements of the approach that required individual research. As different types of data needed to be acquired different methods were applied to collecting data for the construction of cost models. The general methods and their combination aimed at converting data into knowledge supporting cost estimation are described in this section; results of the data collection and the integration into cost models are presented in the respective section of Chapter 5 and Chapter 6.

The basic techniques employed in this work are regression analysis (hardware components), static code analysis (software components) and expert opinion elicitation (structural dependencies, validation). Research in third-party databases and technical literature provided additional information on average burden cost. These techniques were combined in order to create a service robotics knowledge database for early phase cost estimation. The application of each method is detailed in the following sections.

Obviously, the adequate values for an accurate estimate will vary from company to company depending on its individual situation. A typical example of such discrepancies is the wage rate of engineers and software developers: a company located in the People's Republic of China is subject to different wage structures than a European enterprise. To compensate for these issues the software tool supporting the estimation approach in this work allows the user overriding of any default parameter value. Some types of information are more difficult to survey than others; in the case where no or too little information is available the collected and derived values of the presented work provide the estimator with default values for component parameters thus offering support for quick estimates in early development stages.

4.4.1. Regression analysis

The presented approach applies regression analysis for the development of cost models for hardware components. The steps required for this method – data collection, modeling, model selection and adaptation, model validation and

assessment – are described in the following subsections. The general procedure of regression analysis is depicted in Figure 4.4-1 (Backhaus 1996, p. 8).

4.4.1.1. Data Collection

As data constitutes the starting point for the regression analysis it has to be collected first. To this aim, sampling of hardware component data is used. Sampling is the selection of units from a collectivity to be studied, the basic population. The selected units must be representative of the basic population in order to allow significant deduction of information on the properties of the domain examined (Sachs 2002, p. 99).

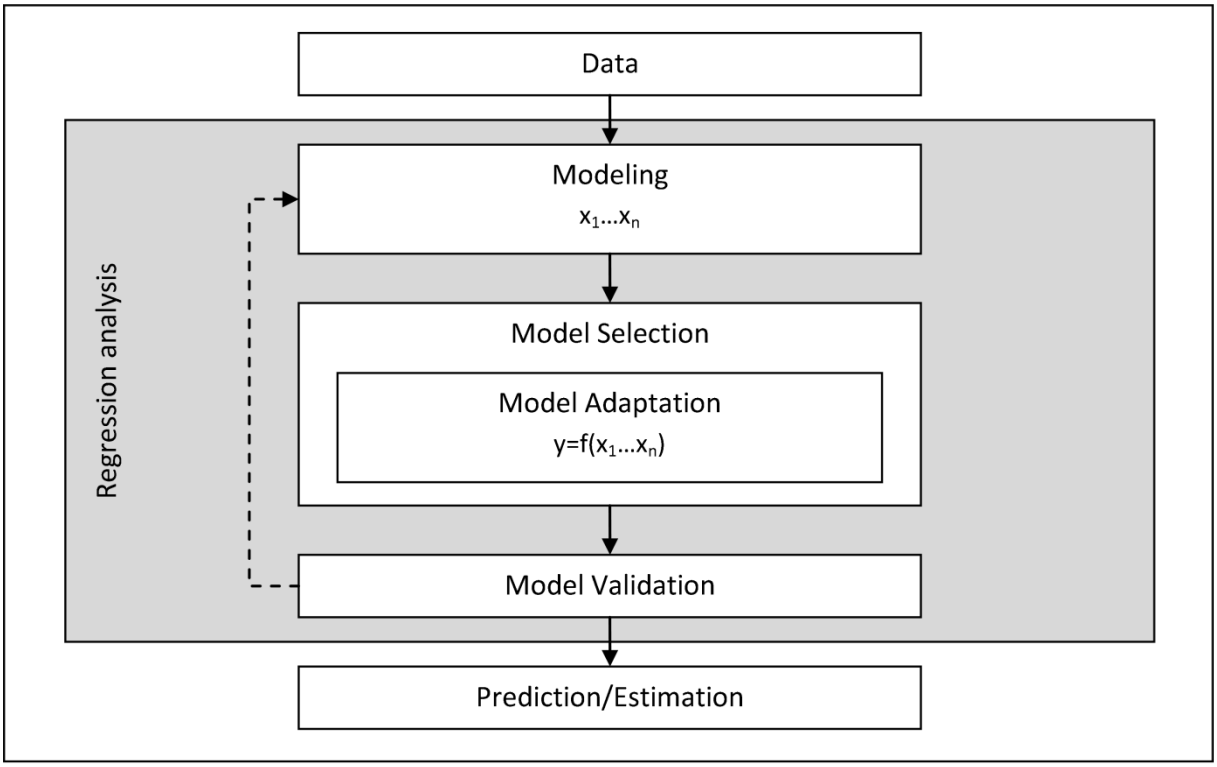

Figure 4.4-1 General steps of regression analysis

Sampling is used to retrieve information on the typical parameters for each hardware component category and their impact on prices. As the categories are designed to be collective terms the underlying basic population is not known. The collection and analysis of the entirety of elements belonging to each category would require massive market research efforts beyond the scope of the presented work. Therefore, the sampling approach is deemed a justified alternative.

A problem of the basic population being unknown consists in the impossibility to prove the representativeness of the samples selected. Thus representativeness of the samples selected can only be assumed. However, the selected units were cross-checked with experience from practical use in service robot development at the Fraunhofer IPA to ensure a sufficient degree of relevance.

For each hardware component category a list of specific component parameters was aggregated from a presample of five to ten products pertaining to the category. These product characteristics were then filtered for product relevance (is the feature important for the product's functionality?), number of mentions (threshold 60%) and statistical evaluability (nominal features were excluded) in order to determine the potential input variables for the cost model. Subsequently, the parameter values for the selected characteristics and single unit prices were sampled for a larger number of products.

4.4.1.2. Modeling

After the collection of data, the regression model must be formulated i.e. the input variables for the model have to be determined. The objective of this step is to create a model with a minimum set of independent variables $x_1, ..., x_n$ that reflects the influence of the real influencing factors on the dependent variable y (the component cost) as correctly and completely as possible.

Given that little information is available at early product development phases it is essential to reduce the number of estimation input parameters i.e. independent variables: the more input data the estimating person is expected to collect in order to perform an estimation the higher the probability that the estimation will be foregone. Also, collecting and editing large amounts of data for the development of the estimation models significantly increases the cost and time of the estimation process as well as those of the model development. On the other hand, improper and exaggerated reduction leads to loss in estimation quality. Thus, a compromise must be found between usability of the estimation model, model construction effort and estimation accuracy.

4.4.1.3. Model selection and adaptation

In order to estimate the hardware component cost a functional relation between the technical parameters identified in the previous step and the cost must be established. The general form of such a function is

$$y = f(x_1, \dots, x_k) + \varepsilon \tag{4.1}$$

where y is the dependent variable cost, $x_1, \dots, x_k$ are the independent k variables for the technical parameters and ε is the estimation error. ε is the cost component not explained by the model (Fahrmeir 2009, p. 19).

The most common regression model type is the linear regression model of the form

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon \tag{4.2}$$

which assume a linear relationship between independent and dependent variables. Another central assumption of linear regression is that the error term is normally distributed around the expected value 0. The coefficients $\beta_0, \dots, \beta_k$ are often estimated using the method of least squares on a dataset of at least k+1 observations. The major problem that can arise when applying linear regression is that the estimation error can be very large if the underlying data does not reveal a linear relationship i.e. the linearity assumption can be too restrictive.

Non-parametric models attenuate this issue by postulating solely a continuous and smooth regression function without assuming a simple linear form (ibid., p. 41). The original values are fitted to a smoothed curve using smoothing kernels, splines or wavelets to satisfy this condition. This permits a more exact mapping of relationship between independent and dependent variables at the price of increased computational effort and more difficult interpretation (Fox 2000, p. 2). In unrestricted non-parametric regression the expected value of the dependent variable under given independent variable values can be modeled as

$$E(y|x_1, \dots, x_k) = f(x_1, \dots, x_k) \tag{4.3}$$

The assumptions concerning the estimation error ε remain the same.

Because a large variety of different non-parametric models exist a detailed description is foregone here.[11] The presented approach employs the generalized

[11]For a detailed description of different non-parametric models see (Fahrmeir 2009, pp. 297–398).

additive model (GAM). An additive model can be formulated as the sum of smooth functions for the independent variables

$$E(y|x_1, \dots, x_k) = \alpha + f(x_1) + \dots + f(x_k) \tag{4.4}$$

i.e. instead of using one function for all independent variables a function is determined for each variable separately. This renders the model more restrictive than the general form but is more flexible than linear regression which is why it can be regarded as a fair compromise between accuracy and modeling effort (Fox 2005, p. 116).

The cost estimation models were calculated using the sample data and the *gam* module from the R *mgcv* library (RDCT 2012, pp. 2631–2820). To find the most adequate model, backwards selection was conducted i.e. for each component category a complete model with all independent variables chosen for data collection was constructed and subsequently reduced models were developed by removing independent variables. Furthermore a linear model was calculated using the R *lm* module of the *stats* library. In order to select the most adequate model based on an assessment of its predictive quality the complete model, reduced models and the linear model were compared applying several statistical measures.

The *coefficient of determination* R^2 is a measure for how well future outcomes are likely to be predicted by the constructed model. It indicates the proportion of explained variance by the model and is defined as

$$R^2 = \frac{\sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} = 1 - \frac{\sum_{i=1}^{n} {\varepsilon_i}^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}; 0 \leq R^2 \leq 1 \tag{4.5}$$

where $\hat{y}_i$ is the i-th estimated value out of n estimations, $\bar{y}$ the observed mean of the respective n observations and y_i is the value of the i-th observation. ε_i is the residual or estimation error of the i-th estimation i.e. the unexplained variation of the model. R^2 values close to 1 indicate a good fit of the data whereas R^2 close to 0 indicate an inadequate fit. Nevertheless, the informative value of the coefficient of determination is limited as it does not indicate if the model was specified correctly. Also, R^2 does not show possible correlations between the independent variables (Fahrmeir 2009, p. 99). Thus the coefficient of determination is only interpreted in conjunction with further indicators.

Akaike's Information Criterion (AIC) is a common criterion for model selection (Fahrmeir 2009, p. 161; Acquah 2010). It indicates the fitness of a model and also accounts for model complexity including a penalizing term for the number of model parameters. It is defined as

$$AIC = -2\ln(\hat{\theta}) + 2p \tag{4.6}$$

where $\hat{\theta}$ is the parameter vector with those values yielding the maximum of the likelihood function for the estimated model; p is the number of parameters of the model. The information criterion can only be used in comparison with other models; a single AIC value does not reveal anything about the model. Comparing AIC values the model with the smaller AIC value is preferred.

Because the AIC has a systematical bias towards models with a larger number of parameters for increasing sampling sizes, another information criterion countering this effect was considered in the model selection. The *Bayesian Information Criterion* (BIC), defined as

$$BIC = -2\ln(\hat{\theta}) + p \cdot \ln(n) \tag{4.7}$$

allows the reduction of mentioned bias by incorporating the sampling size n in the calculation. As with the AIC, smaller values for the BIC indicate a superior model.

Cross-validation is another method for assessing a model's predictive quality. The observation data is partitioned into two subsets: one for the estimation of the model – the training data – and the other for validating the model i.e. examining how well the model predicts the dependent variable values of the second subset. A large discrepancy in the fit between training data and validation data indicates overfitting of the model to the training data. For the purposes the presented work the *generalized cross-validation* of the R *gam* algorithm was used which is defined as

$$GCV = \frac{nD}{(n - DoF)^2} \tag{4.8}$$

where D is the deviance, n the number of data and DoF the degrees of freedom of the model (RDCT 2012, p. 2660);[12] the smaller the value of the GCV the better the fit of the model.

[12]For a detailed discussion of the GCV criterion see e.g. (Golub 1979).

The *partial F-test* can be used to evaluate a model's performance in comparison to the complete model. This is realized by trying to disprove the null hypothesis that the complete model does not offer a significantly better fit of data than the reduced model given a specific level of significance i.e. that their variance is the same. The F value is calculated as

$$F = \frac{\frac{RSS_{reduced} - RSS_{complete}}{p_{complete} - p_{reduced}}}{\frac{RSS_{complete}}{n - p_{complete}}} \; ; \; p_{reduced} < p_{complete} \tag{4.9}$$

where $RSS_{reduced}$ and $RSS_{complete}$ are the sums of the squares of residuals of the reduced and complete model, respectively, $p_{complete}$ and $p_{reduced}$ are the number of parameters of the models and n is the number of data points. If the calculated p-value of F is below the significance level α the null hypothesis is rejected. In this work, α was set to 0.05.

The *relative average estimation error* can also be employed as another model quality criterion. It is calculated as

$$e = \frac{1}{n} \cdot \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\sum_{i=1}^{n} y_i} \tag{4.10}$$

where $\hat{y}_i$ is the estimated and y_i the observed value of the i-th sample and n is the number of observations. As e is calculated from observed values it has limited informational value on the general estimating quality of the model; an overfitted model in particular with small e can still be of poor quality for predicting random values. However, if used in conjunction with further model evaluation techniques this measure can serve as another quality indicator.

Depending on the results of the comparison applying all of the described measures the most adequate cost model was selected for each hardware component category.

4.4.1.4. Model Validation

In order to validate the chosen model the fulfillment of its basic assumptions must be verified. The advantage of a non-parametric model in this context is that linearity is not assumed and thus need not be proven. Apart from the cross-validation

already mentioned in the previous section several validity checks were conducted for each model.

The independent variables were examined for multicollinearity i.e. the degree of their linear dependency was determined. Multicollinearity is to be avoided as it renders a model more sensitive to variation of the correlated input parameters and thus introduces bias. The *variance inflation factor* is a measure for collinearity and is defined as

$$VIF_j = \frac{1}{1 - R_j^2} \tag{4.11}$$

where j is the index of the j-th input variable of the model and R_j^2 is the coefficient of determination of the regression equation

$$X_j = \alpha_0 + \sum^{k} \alpha_i X_i + \varepsilon \tag{4.12}$$

i.e. the VIF is calculated for each input variable. If the VIF is larger than 10 the multicollinearity of the variable is considered high (Neter 1989, p. 409; for a critical review cf. O'Brien 2007).

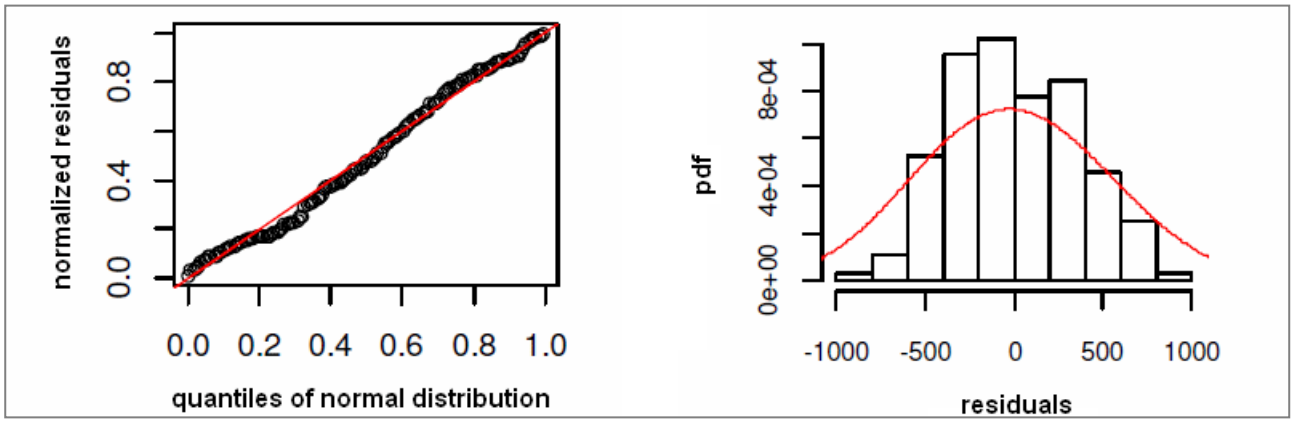

Figure 4.4-2 Example of a Q-Q plot

The sample data was also manually checked for outliers. Possible outliers in the input data were scrutinized for plausible explanations and excluded from the data if found to be abnormal. The exclusion of these values reduces the number of available data for model estimation but renders the resulting model more robust and is thus deemed justifiable for the purpose of early cost estimation.

The assumption of the residuals being normally distributed was examined using Q-Q plots which show the probability distributions of the normal distribution quantiles

compared with the normalized residual quantiles; an example is depicted in Figure 4.4-2 (adopted from Schönberg 2012, p. 134).

4.4.2. Static Code Analysis

Static code analysis is the analysis of software code without executing the code, i.e. it is the analysis of code structure. In the scope of the presented work it was employed to extract the size of relevant code packages as one of the crucial software component parameters in order to attribute software development efforts.

4.4.2.1. Code selection

Because the estimation of the presented approach takes place at an early development stage software implementation for the service robot project has probably not started yet; consequently, no or little code will be available for direct analysis. In order to derive estimates comparisons with existing software for similar projects i.e. service robot related implementations can serve as a plausible estimation basis.

Costs of software development depend on copious software project attributes including code complexity and size, user requirements, team skills, development methods and many more (Jones 2007, p. 24). More often than not, these parameters are unknown or only fuzzily specified in early development phases; a circumstance that renders the prognosis of necessary development efforts a daunting task.

Ideally, all relevant parameters of the analogy are identical to those of the robot software in development i.e. application area, complexity, experience of the development team, development tools, etc., in which case the development effort would be expected to be exactly the same. Usually, differences between separate software projects are common but comparing two similar projects still yields a reliable estimate of the development effort to be expected (Shepperd 1997). For this reason, an analogy-based approach was chosen for the presented estimation methodology in order to estimate the impact of software development on the overall costs. This entails analyzing comparable existing robot software – henceforth software analogy – in regard to its development effort and mapping the identified

effort quantifications to the specific robot system of which costs are to be estimated.

With experience from similar software projects at hand parallels can be drawn between required development efforts. If the estimator holds knowledge of incurred costs from past project she or he can extrapolate towards the new venture and thus achieve early cost estimation. Unfortunately, this approach does not lend itself to pioneering development areas where the development team has little or no prior experience and no information about efforts and costs to be expected.

The way out of this predicament is to find a suitable software analogy. A prospective software analogy has to meet certain general selection criteria (Shepperd 1997, pp. 737–739; Shepperd 1996, p. 171):

1. Source code must be available for code analysis, as otherwise derivations concerning software size and effort are impossible due to non-disclosure issues. This condition is per definition met by open-source software.
2. Software must be functionally related, i.e. built for same or similar principles of operation, otherwise the chosen software is no appropriate analogy.
3. Software development paradigms underlying the analogy’s design must be the same or similar as the paradigms for the planned software because the underlying paradigm has a strong influence on the software development productivity. Thus only comparisons between projects under similar development conditions yield sensible results.
4. The analogy's software quality should meet verifiable standard requirement so that the comparison also holds information on the quality level to be expected for a given development effort estimation.

Furthermore, as the presented approach shall cover a wide spectrum of robots and not one specific type a suitable software analogy must allow for flexible selection of necessary software packages and omission of irrelevant ones which can be formulated as two specific selection criteria:

5. The granularity of its modules should be equal or higher than the software categorization in Chapter 5.1.2.3 while covering all of the therein described functionalities in order to permit a more detailed customization of software components.
6. The software analogy must be aimed at software reuse, i.e. its components should be designed for being deployed in different robots with minimal effort.

Name	**Description**
Carnegie Mellon Robot Navigation Toolkit *CARMEN* (CARMEN-Team	Basic robot control and robot simulator
Coupled-Layer Architecture for Robotic Autonomy *CLARAty* by Jet Propulsion Laboratory (JPL 2008)	Domain-specific robotics software
Fawkes distributed by Japan's National Institute of Advanced Industrial Science and Technology	Component-based software framework for robotic applications; robotics technology middleware
Urbi by Gostai Technologies (Gostai 2012)	Components framework for robots and complex systems in general
Player project (The Player Project)	Robot framework, simulator and
Open Robot Control Software *Orocos* (The Orocos Project 2012)	Framework for component-based robot control software
Orca (Orca Robotics 2009)	Robot framework with focus on ease of use, originally based on Orocos
Mission Oriented Operating Suite *MOOS* (OMRG 2008)	Cross platform middleware for robotics research
Mobile Robot Programming Toolkit (MPRT 2012)	Library collection with focus on efficiency and reusability
Open Mobile Robot Architecture *OpenMORA* (OpenMORA 2012)	Complete robot architecture based on MOOS and MRPT
Robot Operating System *ROS* by Willow Garage (Willow Garage	comprehensive framework or "meta-operating system" for robots; reusable
Dave's Robotic Operating System (Austin 2009)	Basic software modules for modular programming and mobile robots

OpenJAUS (OpenJAUS 2012)	Software library and SDK implementing the Joint Architecture for Unmanned
RI-JAUS SDK by Jaybridge Robotics (Jaybridge Robotics 2012)	Cross-platform software development kit implementing the JAUS protocol for
Open Platform for Robotic Services *OPRoS* (OPRoS WIKI 2011)	Component based framework, GUI editors and simulator
Yet Another Robot Platform *YARP* (Fitzpatrick 2012)	Robot control SDK for flexible hardware interfacing and peer-to-peer
OpenRAVE (Diankov 2012)	Environment for testing, developing, and deploying motion planning algorithms in robotics applications

Table 4.4-1 Examples of open source robotics software

As of 2012, many different frameworks and tools for service robots exist. For the purposes of the presented approach, only open source projects were considered due to the necessity of access to the source code. Table 4.4-1 gives a non-exhaustive overview of available open source robot frameworks and tools as of July 2012 (cf. Niemueller 2010; Wikipedia 2009).

As the number of available frameworks is too extensive for an all-encompassing comparison the selection of an adequate software analogy is restricted in the presented approach to one framework, the Robot Operating System (ROS) by Willow Garage. The selection of ROS does not imply its superiority over other robotics frameworks; most of the mentioned frameworks appear to be suitable candidates, too. All of the above frameworks meet the requirement of open source code and functional relation to robotics. Many of them also meet the requirements of reusability and component-based design. The reasons why ROS was selected are:

1. Verifiable quality standards: The release of packages must adhere to a defined policy that postulates prerelease testing to assure operability and compatibility with other stacks (Willow Garage 2010).
2. High interoperability: There is no hard separation between frameworks as they often can be used in combination or have interfaces to each other.[13] Many of

[13]Orocos developer policy even strongly encourages combining efforts, s. (The Orocos Project).

the mentioned frameworks offer ROS interfaces and ROS is committed to integrability: "ROS is easy to integrate with other robot software frameworks" (Willow Garage 2009d).

3. Fine granularity: ROS offers a file system with different levels of granularity with several hundreds of stacks and even more packages. This fineness permits component-based software estimation on a detailed level.
4. Wide-spread usage: As of 2012, ROS has been integrated into a variety of heterogeneous mobile robot systems, e.g. PR2 by Willow Garage, Care-O-bot 3 by Fraunhofer IPA, Robotic Busboy by Intel Research's Personal Robotics project, TurtleBot and Husky A200 by Clearpath Robotics and many more.

Table 4.4-2 exhibits a comparison of selected frameworks regarding these aspects, 'n.d.' stands for not documented i.e. no description has been found in the respective project's documentation.

Project	**Open source**	**Component-based dev.**	**Quality control**	**Module granularity**	**Reusable design**	**ROS inter-operability**
ROS	✓	✓	✓	fine	✓	-
Orocos	✓	✓	n.d.	medium	✓	✓
URBI	✓	✓	n.d.	medium	✓	✓
OPRoS	✓	✓	n.d.	medium	✓	n.d.

Table 4.4-2 Comparison of robot software frameworks

Nevertheless, the analysis of several robot software frameworks is deemed highly desirable and expected to provide additional insight for the analysis of software development efforts in this area but is beyond the scope of this work.

4.4.2.2. Code sizing

In order to infer efforts estimates for the software development labor the code size must be measured. The most common metrics, lines of code (LOC) and function points (FP) were both combined in the presented approach by applying the so-called backfiring method. Backfiring is a technique developed by Capers Jones allowing the translation of source code statements in a specific programming language into adjusted function points (Jones 1995). This approach allows extracting function

points by LOC counting and subsequent extrapolation of FP using heuristic, coding language specific conversion rates.[14]

Usually, the ascertainment of function points is carried out manually by a function point expert which renders the process laborious and expensive. Backfiring allows the mathematical derivation of function points from lines of code but also with less accuracy. This is possible due to the apparent correlation between source code size and function points depending on the programming language used. Statistical data as a result of research into real-world software projects has been gathered by specialized enterprises like the SPR company over the last three decades yielding probability values for the ratio of source code statements to function points for more than 500 programming languages.

The cost estimation for software of the approach in the present work relies on the Programming Language Table (PLT), Version 2007d (SPR 2007).[15] This compilation provides statistical mean, lowest, highest and median source code statement to function point ratios for each programming language. Table 4.4-3 shows an excerpt of the table (adopted from ibid., p. 2).

	LOC per function point		
Language Name	**Low ($k_{j_{WC}}$)**	**Mean (k_j)**	**High($k_{j_{BC}}$)**
CLOUT	25.7	40.0	84.9
CMS2	98.4	106.7	116.5
CMSGEN	14.2	18.8	27.3
COBOL	98.4	106.7	116.5
COBOL .Net	35.8	71.1	190.6

Table 4.4-3 Excerpt from the Programming Language Table

The data collected, i.e. the numbers for LOC counted for the selected packages permits the calculation of a corridor of the approximate amount of function points

[14]Jones points out that only logical source code statements can be used for this method. The LOC used in this thesis thus exclude comments and empty lines.

[15]The function points are based on IFPUG standard 4.1.

for a given code. The expected size $E(s_i)$ of software package i is determined by the sum of the lines of code LOC_{i_j} of each package divided by the appropriate lines of code to function point ratio $k_j$for the respective programming language j taken from the PLT[16].

$$E(s_{i_j}) = \frac{LOC_{i_j}}{k_j} \tag{4.13}$$

$$E(s_i) = \sum_{\cdot} E(s_{i_j}) \tag{4.14}$$

The expected worst and best case size values were also calculated in the same way as $E(s_{i_j})$ but with the respective conversion rate $k_{j_{WC}}$ and $k_{j_{BC}}$ which are also provided by the PLT.[17] These values allow the calculation of corridors for software sizes instead of single point estimates.

4.4.3. Modeling estimation uncertainty

Estimates are always subject to uncertainty, particularly in early product phases. Its major source is the lack of detailed knowledge or definition of the final product; the more details on design, environment and mission parameters are set the lower the uncertainty level will be. This phenomenon is referred to as the "cone of uncertainty" (Meisl 1988, pp. 95–100); (McConnell 2006, pp. 35–37); (Stösser 1999). Figure 4.4-3 depicts a version of this cone where the estimation baseline increases over the course of time (Trivailo 2012, p. 6). Also, the tendency of exceeding the cost estimate in contrast to underrunning it is illustrated.

Table 4.4-4 exhibits typical error ranges for sequential software projects; although these indications stem from the software domain the inherent uncertainty concept can be extended beyond the field of software development (ibid.). The presented numbers illustrate that high accuracy of cost estimates cannot be expected in early

[16]The column "Nominal Value SS/FP" in the PLT holds the arithmetic mean of all observed values for the considered programming language; c_l is its inverse.

[17]$k_{j_{WC}}$is the inverse of "Calculated SS/FP Low",$k_{j_{BC}}$ is the inverse of "Calculated SS/FP High".

project phases because at this stage there are many unknown or undetermined factors that have impact on costs.

Commonly, point estimates are interpreted as the median i.e. the 50% of all actual values lie below and 50% above the estimate. As illustrated in Figure 4.4-3 (ibid.) and Table 4.4-4 (adopted from McConnell 2006, p. 39) this interpretation does not always hold true. Another problem with point estimates is that lower and upper limit of the estimation remain unknown to the interpreter. For this reason single-point estimates, i.e. "product X will cost 100€ per unit" would give the false impression of definiteness. Without an indication of its likelihood the estimate is of little informational value.

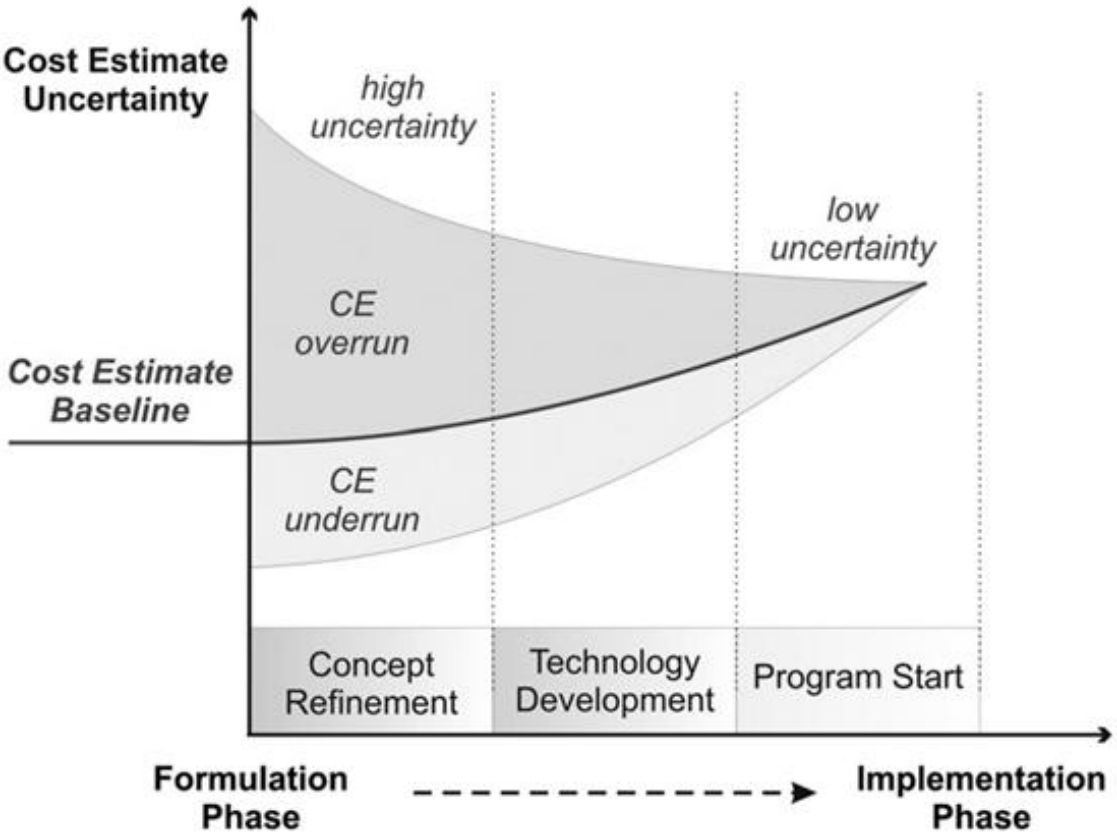

Figure 4.4-3 Conceptual example of the cone of uncertainty

Phase	Possible error on low	Possible error on high
Initial concept	-75%	+300%
Approved product	-50%	+100%
Requirements	-33%	+50%
User interface design	-20%	+25%
Detailed design	-10%	+10%

Table 4.4-4 Estimation error for costs and efforts in software development phases

To compensate for the estimation error a measure of inaccuracy should be denoted for assessing the probability of a large deviation from the estimate, often called cost risk (McConnell 2006, pp. 6–9; Galway 2007, p. ix).[18] Cost-risk analysis requires knowledge of the underlying error probability distribution. Since the actual probability density functions are unknown the prevalent approach is to assume normal distribution, arguing that many processes can reasonably be approximated due to certain statistical effects and characteristics like the central limit theorem or independency of process variables (Casella 2002, p. 102). However, the diverse cost constituents exhibit different probability distributions which is why asymmetrical distribution forms are also taken into account here.

4.4.3.1. Probability Density Functions

As aforementioned, the costs estimated in the presented approach are basically divided into material and labor costs. In order to achieve a differentiated view on the estimation uncertainty the distribution for each of these cost types is derived separately. The reason for this distinction is that costs for activities tend to be distributed asymmetrically whereas material costs can be assumed to be symmetrically, normally distributed (Browning 2002, p. 432; McConnell 2006, pp. 7–8).

Concerning hardware costs, the estimation error i.e. the model standard error expresses the uncertainty of the respective costs. The standard errors calculated with predict.gam are based on the Bayesian posterior covariance matrix in the fitted gam model (RDCT 2012, pp. 2747–2750). Because the standard error is assumed to be normally distributed for the regression model derivation the probability density

[18] Galway states that the term 'cost risk' only applies for exceeding estimates.

function of the estimated hardware costs is also assumed to be normally distributed; the mean represents the estimated value and the standard deviation equated with the standard error calculated by the model.

Labor or activities costs tend to be right skewed as work processes are truncated to the left because there is a limit to how efficient a process can be performed but the expansion of costs due to failures, delays, requirement changes and other factors is virtually limitless. As an example in the time period from 1994 to 2004 circa 75% of the software projects overshot the planned schedule or budget (McConnell 2006, p. 25).

To represent uncertainty in labor costs various asymmetric probability density functions can be used, the beta distribution being the prevalent one. The Program Evaluation and Review Technique (PERT) – an acknowledged planning tool for project management – applies the beta distribution to capture uncertainties concerning activity related costs (Keefer 1993). As this method has been applied over many projects in the last decades the beta distribution is used for modeling labor effort distribution in the presented work (e.g. Chen 2009; Xu 2007).

The continuous beta distribution defined on the interval [0,1] is parameterized by two shape parameters p and q, both required to be greater than zero. Its density function is defined as

$$f(x) = \frac{1}{B(p,q)} x^{p-1}(1-x)^{q-1} \tag{4.15}$$

with the beta function $B(p,q)$ defined as

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} = \int_0^1 u^{p-1}(1-u)^{q-1} du \tag{4.16}$$

and Γ denoting the gamma function. Outside of [0,1] $f(x)$ equals zero.[19]

The expected value of the beta distribution is

[19]The general beta distribution is defined on the general interval $[a,b], a,b \in \mathbb{R}$ but is more complicated. As the normalization of [a,b] to [0,1] and subsequent use of the beta distribution defined on [0,1] and denormalization of the results is mathematically more manageable without loss of information, only the beta distribution on [0,1] is considered.

$$E(X)_{Beta} = \frac{p}{p+q} \tag{4.17}$$

and the variance

$$Var(X)_{Beta} = \frac{pq}{(p+q+1)(p+q)^2} \tag{4.18}$$

The three-point estimation technique established in PERT allows the construction of a beta distributed probability density function by using only three data points: best-case estimate BC, most likely estimate MLC[20] and worst-case estimate WC. These three estimates can be based on heuristics or elicited from experts (Santillo 2005). The expected value and variance are approximated by the respective equivalents of the double-triangular distribution

$$E(X)_{double-triangular} = \frac{WC + 4MLC + BC}{6} \tag{4.19}$$

$$Var(X)_{double-triangular} = \left(\frac{(WC - BC)}{6}\right)^2 \tag{4.20}$$

The shape parameters p and q are determined by equating (4.3) and (4.5) as well as (4.4) and (4.6) and consequently solving the equation system to

$$p = \frac{E(X)^2 - E(X)^3 - E(X)Var(X)}{Var(X)} \tag{4.21}$$

$$q = \frac{E(X) - 2E(X)^2 + E(X)^3 - Var(X) + E(X)Var(X)}{Var(X)} \tag{4.22}$$

With the shape parameters determined the beta distribution can be calculated for the normalized data points.

4.4.3.2. Uncertainty Indicators

The modeled probability distribution functions permit the calculation of various statistical measures that can serve as uncertainty indicators. In the scope of this work standard deviation and quartiles are used for this purpose. The standard deviation expresses how large the variation from the expected value is; a large standard deviation in relation to the expected value thus expresses a higher degree of uncertainty. One advantage of this measure is its intuitive interpretability.

[20]Note that the most likely case is represented by the distribution's mode, not its expected value.

Another means of expressing uncertainty is the calculation of percentiles. A percentile marks a value below which a given percentage of observations and by extension estimations fall e.g. the fifth percentile of costs is the threshold below which five percent of the estimations would lie. Calculating several percentiles allows the estimation of probability ranges.

Figure 4.4-4 displays an exemplary beta distribution of labor costs. The dashed vertical lines indicate the 25th, the 50th (median) and the 75th percentile i.e. first, second and third quartile; the dotted vertical line represents the expected value. The interquartile range i.e. the range between first and third quartile lies roughly between 11,200 and 18,200 € and covers 50% of all values within the estimated range thus indicating a corridor for the likelihood of the estimate. The quartiles can also be applied for a staggered risk indication: The cumulated probability up to a quartile can be interpreted as the probability of the actual cost lying below the respective quartile; e.g. for the given example a 75% chance of costs lying below circa 18,200 €. Because the interpretation of the quartiles can be regarded as intuitive and comprehensible their calculation was incorporated in this work.

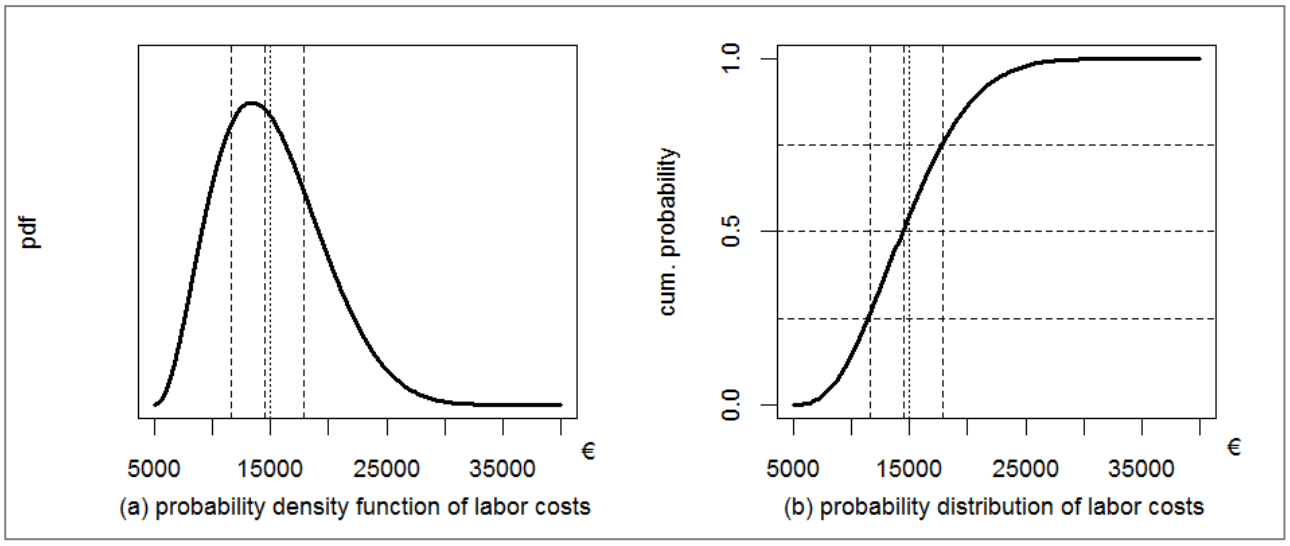

Figure 4.4-4 Exemplary beta distribution function

4.4.4. Expert opinion elicitation

Expert opinion elicitation or interviewing means retrieving information on a specific matter directly from professionals with experience in that particular field. This method often is the only available solution in cases where little or no data is

available otherwise: “If suitable data on the structure in question is not directly available, information acquisition can only be realized by executing interviews with experts” (Maurer 2007, p. 97). Expert interviews are one of the standard instruments of empiric social research (Flick 2007, p. 19; Schnell 2008, p. 299).

As little explicit information on the mapping of robot skills to components for service robots is publicly available expert elicitation was used to establish such systemic relationships. Furthermore, expert interviews were conducted regarding the expected reuse ratios for the considered software components and for hardware installation efforts.

For the derivation of classifications of skills, hardware components and software components the first step was the research into existing categorizations of robotic functions. Available literature was scanned for recurring expressions and descriptions of robotic skills. Based on these characterizations suggestions for item categorizations were developed. The existing categorizations considered are classifications employed in EUROP (Bischoff 2009, pp. 28–33), EFFIROB (Hägele 2010, p. 341), RoboEarth (Waibel 2011, p. 71) and the Handbook of Service Robotics (Siciliano 2008, pp. XXXVII–XLV). The categorizations were then presented to and discussed with three experts separately in order to evaluate their correctness. The criteria for items to be valid are applicability and importance to service robots as well as tangibility and understandability as they should allow their intuitive use for an early estimation of functions and components necessary.

For the structural mapping i.e. the entries of dependencies in the structural matrices a workshop was held with six experts for service robot development. The concept of the structure matrices, the categorizations of items, and dependency types were explained and pertinent documentation handed out. After clarification of any questions at this point the experts were requested to individually fill in direct dependencies on print-out sheets of six structure matrices for the mapping of task-to-task, task-to-hardware, task-to-software, hardware-to-software, hardware-to-hardware, and software-to-software. The individually completed matrices were then combined into one aggregate matrix for each matrix type.

The determination of software reuse ratios was conducted by individually presenting two experts on service robots a list of the considered ROS packages with

the request to allocate a percentage range for adaptation and customizing effort to each package. The percentage given designates the relation of altered code to total code size. Possible choices for code adaptation and customizing ranges were:

- >0% to 10% are reimplemented, 90% to 100% are reused as is with only minor adaptation
- 10% to 30% are reimplemented, 70% to 90% are reused as is with only minor adaptation
- 30% to 50% are reimplemented, 50% to 70% are reused as is with only minor adaptation
- 50% to 70% are reimplemented, 30% to 50 % are reused as is with only minor adaptation
- 70% to 100% are reimplemented, up to 30% are reused as is with only minor adaptation
- No modification, installation only, 100% reuse with no modification at all
- Package unknown and/or modification effort unknown
- Package not used

Ranges were chosen instead of single point estimates in order to allow for a degree of uncertainty and to speed up the ratio estimating process which was deemed necessary as 469 packages had to be evaluated separately. The calculated averages of the range indicated by the experts are used as the most likely value for the cost estimation.

For the installation costs of hardware and software components for each of the categories two experts were asked individually to estimate installation times based on their experiences.

For hardware installation costs, a list of the hardware component types was presented to each expert with the request to gauge the approximate installation time for every component type. For software costs, the experts were asked to estimate how long it takes to install ROS on a service robot of their knowledge. The basic idea is that once the software developed for the service robot subject to the cost estimation has reached a releasable state it will be installed as one suite similar to a framework like ROS. Installation of single packages was not considered in the

presented approach it is already included in the productivity rates of the software development stage, i.e. the software installation costs only apply to robot units produced after the prototype has been built. For both categories, values for minimum, most likely and maximum time required were elicited.

5. Function-based Structure Estimation

In order to support the estimating person in each of these steps the necessary knowledge must be made available beforehand. This chapter covers the aspects of mapping desired robot functionality to hardware and software components. First, the categorization into the three *domains Skills, Hardware Components*, and *Software Components* is explained and consequently how relationships between them are derived.

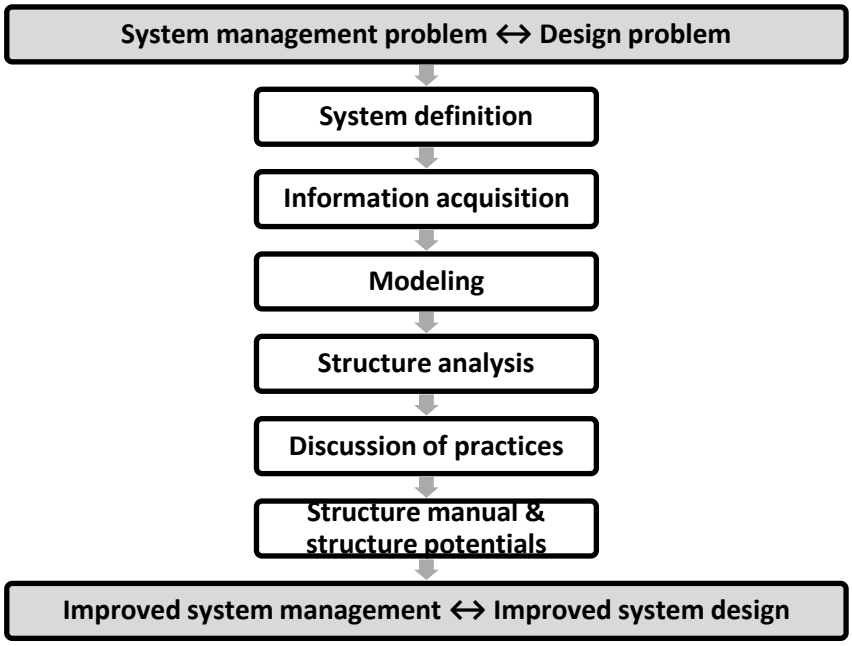

Figure 5-4.4-1 General structure management approach

From a structure management point of view this part of the approach covers the steps system definition, information acquisition and modeling concerned with the structure of a service robot. Figure 5-4.4-1 depicts the general steps of the complete structure management process where a handling problem or a design problem forms the starting point (adopted from Maurer 2007, p. 69). As handling problems are concerned with controlling issues for existing solutions It should be noted that it

is not the aim of the presented approach to evaluate the appropriateness of a specific structure to a given application problem but to indicate central structural elements which have significant impact on resulting costs which is why the latter three steps of the indicated structure management process are not considered further in this work.

5.1. System Definition

In order to derive a structure of a robot from which costs can be derived its scope, purpose, and functionality must be defined i.e. the domains of the design problem, the level of detail within the domains, and dependencies considered must be determined in advance (ibid.). The combination of these three preparatory steps constitutes the system definition.

5.1.1. Domain Specification

As outlined in Chapter 4 the presented approach is aimed at estimating costs starting from a point where little more than the robot application is known. In other words, the costs for fulfilling a desired functionality must be determined. The abilities to perform certain functional actions are skills of the robot thus skills constitute the first domain required for the determination of the robot structure.

Skills as abstract concepts do not incur costs themselves but have an impact on costs through their implementation. The evident manifestations of skills are the robot hardware components which embody the physical interface between functionality and physical working environment. Though less evident, the software required to coordinate and control hardware operation forms another important part of the realization of functionality. Thus hardware and software components were chosen as further domains of the robot structure because they represent the domains of product implementation (Kossiakoff 2003, p. 176).

Conceptually, both hardware and software are components and thus could be subsumed within one domain. The separation of these categories was undertaken for two reasons. Firstly, they were divided up in order to prevent the structure matrices to be constructed from growing too large because oversized matrices tend to become difficult to handle by experts (Browning 2001, p. 302; Maurer 2007, p. 37). Secondly, due to their different nature the cost allocations for these categories

exhibit discrepancies that justify separate treatment; e.g. material costs are a significant part of hardware components but negligible for software. Experience from the study EFFIROB showed good practicability of the distinction between hardware and software for cost estimation purposes (Hägele 2010, pp. 37–44).

Process or human resource domains are also potential candidates for the derivation of costs from system structure. Because the costs incurred within these fields can be attributed to the components by adequately modeling the cost functions these domains were not considered separately.

5.1.2. Domain Elements

For each domain the level of detail must be delineated: On one hand, the elements of a domain must be sufficiently detailed as to permit sensible mapping of the robot application and reasonable cost estimation; on the other hand they should not be large in numbers that the sheer quantity becomes too difficult to manage within an early phase estimation process.

Complex systems usually are constructed from a multitude of different components. At early design phases it is hardly practicable to exhaustively list all constituents of the final product (Maurer 2007, p. 67); even if it were feasible the large number of items would exceed the human capacity to handle complexity. Studies show that matrices containing only a small number of items can quickly overburden the user (Maurer 2006). Browning even finds that "individuals may have difficulty building DSMs with more than ten elements" (Browning 2001, p. 302). Complexity puts high demands on the estimator's abilities to collect and process information and plan accordingly; thus the amount of items should be reduced to a number manageable by the user (Dörner 2003, p. 60).

The approach towards usability in this work is to reduce the number of items for each matrix by establishing catalogs with typical categories for service robot *skills*, *hardware components*, and *software components* each containing no more than twenty items.[21]

[21]Although these catalogs have been cross-checked against expert knowledge the author does not claim the list to be exhaustive or exclusive.

As the items in the component catalogs represent generalizations from specific components they are subject to variations of manifold parameters, e.g. size, energy consumption etc. In order to reflect possible differences of characteristics in cost estimates the main cost driving parameters must be identified and the typical range of their values determined. Doing so allows the derivation of specific cost functions based on these parameters thus lessening the loss of exactitude due to generalization.

5.1.2.1. Skills

At the starting point of the estimation, the estimator has an approximate idea of the tasks the robot needs to perform which demand certain skills of the robot. Many of these skills are rather generic and occur in many different robot designs. Unfortunately, there is no standardized list of robotics skills.

Figure 5.1-1 shows different categorizations taken from EFFIROB, EUROP and RoboEarth (Hägele 2010, p. 341; Bischoff 2009, pp. 28–33; Waibel 2011, p. 71). Even though many concepts overlap in meaning, the differences in terminology indicate that many of the applied terms are not strictly defined.

For the estimation approach in this work a list of robot skills has been compiled from current literature. The study EFFIROB shows that the most common categories are perception, navigation, manipulation and human robot interaction (Hägele 2010, p. 340). A slightly extended terminology is used for the initial categorization of robot skills:

- *Perception* entails abilities aimed at discerning signals, objects or environmental conditions.
- *Navigation and Locomotion* is the umbrella term for skills that serve the orientating and moving of the platform in its environment.
- *Manipulation* encompasses all types of exerting physical influence on objects or environment.
- *Communication* describes those skills that enable the robot to engage in information-related interaction with its environment, be it humans, robots or computers. This includes the interpretation of messages and the derivation of consecutive actions.

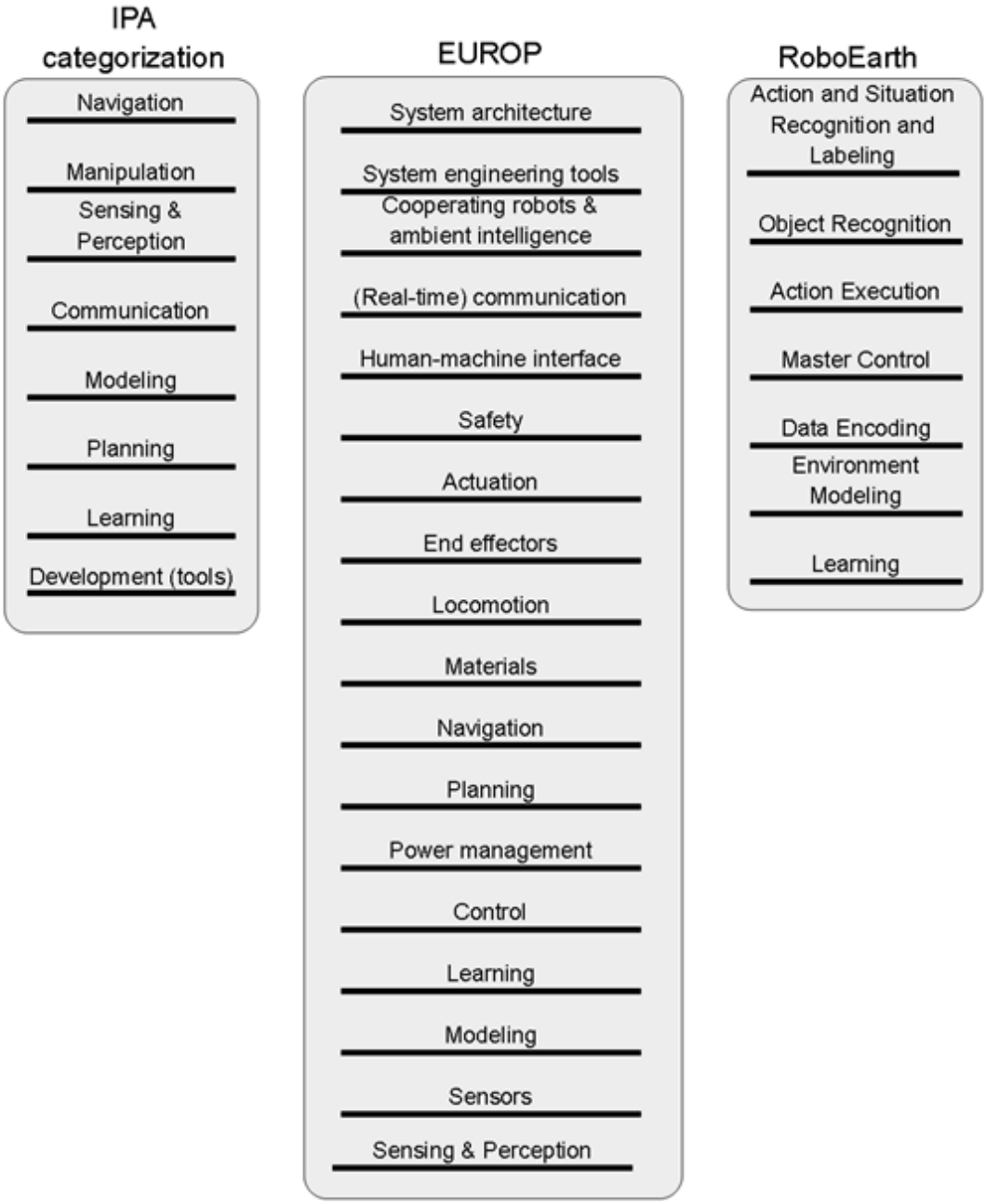

Figure 5.1-1 Different categorizations of service robot skills

These terms serve as a rough first classification of skills. Due to the plurivalent nature of skills the relation between categories and skills is many-to-many, i.e. one skill can fall into several categories.[22] As the mentioned categories are still too wide to facilitate function based cost estimation they are decomposed into skill classes.

[22]An example: Many sensors are obviously used for perception but also employed for navigation so it would be difficult to assign them to only one category.

The fineness of the skill classification is required to contain enough separate items to distinguish between functionalities but their number must not be so excessive that selecting from them would require more detailed knowledge of the robot's design than is available at early development phases.

For validation, the list was individually presented to three robot experts at Fraunhofer IPA and adapted according to their suggestions in order to align the categorization with requirements of applied robot design practice (s. Chapter 4.2). For the sake of clarity the resulting list is shown in Table 5.1-1; each skill is detailed in the following paragraphs.

Perceive Objects	Move Object
Recognize Objects	Process/Alter Object
Interpret Environment	Process/Alter Environment
Perceive evolutionary processes	Send Signals/Commands
Move to Location	Interpret Signals/Commands
Orientate in Environment	Receive Signals/Commands

Table 5.1-1 Service robot skills

In terms of axiomatic design (AD) the skills represent the domain of functional requirements – what the robot must be able to do – whereas the components are the manifestation of the design parameters – what is used to enable the robot to do what it is supposed to (Suh 2001, pp. 10–12).

The skill to *perceive objects* describes the ability to detect the presence of physical entities in the environment when in proximity of the robot without classifying the object. The scope of the detection depends on the sensor type employed. A simple implementation of this skill is a bumper that forwards the information of physical contact of the robot's hull to a colliding object.

Recognizing objects is an extension of perceiving objects in the sense that the robot is able to classify certain objects in its environment based on sensor information and a priori knowledge on the objects to be recognized.

Interpreting environment comprises capacities enabling the robot to deduce type and state of at least some of the environment's features, e.g. lighting conditions,

slopes, degree of "clutteredness" etc. The characteristics to be interpreted strongly depend on the robot's design purpose.

The *perception of evolutionary processes* entails the realization of changes, be they internally (e.g. tool abrasion) or externally (e.g. lawn mowing). The alterations can vary from simple changes (e.g. battery load status) to complex ones (e.g. reordering objects in tridimensional environment). This perception also encompasses detection of trends.

Moving to location is one of the most basic actions of a mobile robot. In the given context of this work it entails the avoidance of any harm to humans, the environment or the robot itself. The movement to a location is executed autonomously.

Orientation in environment describes the ability to derive the robots geographical and topological position within the environment the robot is deployed in.

Moving objects is the ability to purposefully change an object's position by carrying, pushing or pulling it to a desired destination. In the context of skills, objects are considered moveable and of smaller than the robot or at most equally sized.

Processing or altering objects encompasses all volitional modifications of objects, e.g. welding together of work pieces.

Processing or altering the environment is similar to that of objects. The difference lies in the environment being of a much larger extent than objects, e.g. floor or walls, thus necessitating different handling approaches.

Sending signals and commands represents the variety of abilities to relay external output to other entities, be it humans, computers or robots. These messages can take the form of acoustic or visual indications or be transmitted electronically as data packages.

Interpreting signals or commands encompasses all forms of higher level intelligent processing of messages from outside. This entails derivation of (re)actions e.g. replanning a route or stopping a current activity.

The skill of *receiving commands* means offering other entities possibilities of communication via one or several input channels.

5.1.2.2. Hardware Components

Since many components are possible candidates for service robots it is not attempted here to exhaustively list and categorize all potential hardware constituents. Instead, central and frequently incorporated hardware component types were compiled, using the hardware catalog from the EFFIROB project as a starting point. The extraction of initial component types from this source yielded 38 different component classes for the categories of robot arms, grippers, mobile platforms, tactile sensors and contact-free sensors (Hägele 2010, pp. 356–358).

As the EFFIROB study revealed human-robot-interaction to be one of the main functional requirements for service robot the classes *Keyboard, Microphone, Display, Touchscreen, Joystick (or comparable), Speakers* and *Signal Lights* were added to the preliminary category list. Furthermore, the general infrastructure classes *Wiring, Power Supply, Data Storage, Processing Unit* and *Data Interface* were included because elements of these are used in virtually any service robot.

The total of 50 preliminary component classes was then reduced through abstraction and combination of similar classes validated in dialogue with robot engineers at the Fraunhofer IPA. The reason for this reduction is that the structure matrices resulting from such a large number would have grown too unwieldy for data collection on structural dependencies via expert interviews and for uncomplicated usage by the estimating person (s. Chapter 5.2). This reduction is deemed justified as detailed design decision are usually made in product development stages succeeding the early concept phase on which the presented approach focuses; furthermore, the parameterization of components described in Chapter 6.2 avoids overgeneralization.

An overview of the resulting hardware component types is given in Table 5.1-2. Because the list is not exhaustive three placeholder categories were introduced for miscellaneous hardware components. Although no cost models are derived for these categories due to their heterogenity they are included in the structure

matrices to indicate possible relationships with other components thus avoiding the unintentional disregard of components not fitting into any of the specific categories.

Some constrictions were made for certain classes according to the service robot types focused in this approach (s. Chapter 2.1.1). These constraints allow the development of cost models to be better tuned to this kind of systems which are expected to be of higher accuracy than global category cost models. Each component is detailed in the following paragraphs.

Camera	Gripper
Ultrasonic Sensor	Miscellaneous End Effectors/ Tools
Laser Scanner	Input Peripherals
Radar	Output Peripherals
Binary Sensor	Power Supply
Force/Torque-Sensor	Controlling Unit
Gyro-Sensor	Safety Hardware
GPS System	Miscellaneous Structural Hardware
Mobile Platform	Miscellaneous Hardware
Robot Arm	

Table 5.1-2 Hardware component types

Digital *cameras* are sensors providing data on static or dynamic images, either monochrome or in color. They are typically employed as data source for object recognition, environment interpretation or self-localization. The prevailing technologies are charged-coupled device (CCD) and complementary metal oxide semiconductor (CMOS) which represent the categories analyzed in the scope of this work, line scan cameras were excluded. Three-dimensional camera systems (time of flight cameras) were not included due to lack of sufficiently large number of products but the increasing interest in these products is expected to change this situation in the near future.

Ultrasonic (US) sensors measure the transit time of ultrasonic sound waves from sensor to obstacle and back (end-to-end) thus allowing the measurement of distances to nearby objects. They can be used for object detection, obstacle avoidance and simple object recognition tasks. This component type encompasses only reflection-based US sensors excluding those that only measure object presence (these are subsumed under binary sensors below).

Laser scanners or *laser distance measuring devices* emit controlled laser beams and determine the time of flight (TOF) or phase shift of reflected beams from which the distance and shape of the reflecting objects can be derived. Due to their high precision and fast response time they are commonly used for safety zone control and self-localization. This component type includes one-dimensional and two-dimensional laser scanners; laser scanners certified for safety applications were also regarded as long as they do not require an additional evaluation unit. Three-dimensional laser scanners were excluded for lack of a sufficiently large number of products and their arguably low relevance for service robots.

Radar sensors measure the presence of and distance to objects by emitting radio waves and detecting reflections caused by obstacles in the wave's path. A potential application for radar sensors in service robotics is the safety control for the robot's workspace. This component type includes radar chip modules and enclosed radar solutions.

Binary sensors only distinguish between two states; this category comprises mechanical and contactless sensors with a binary output signal. Typical applications are object presence control and immediate environment collision avoidance. The sensors considered include optical, capacitative and inductive proximity sensors as well as push buttons. One necessary condition for an item to be included was that all active parts are integrated in one casing thus one-way through-beam light sensors were not considered. Magnetic proximity sensors were also excluded from data.

As the name suggests *force-torque sensors* perceive and measure forces and torques occurring in physical activities. Typical application purposes are joining operations with a sensor of this type between robot arm and end-effector or in the joints of the robot arm. The regression data includes force-only, torque-only and combined measurement sensors with one to six degrees of freedom. Rotating force-torque sensors were not considered due to their rare application in service robots.

Gyroscopes measure angular position, acceleration and velocity, *acceleration sensors* capture angular and translational accelerations. Because these two types of sensors are used for similar purposes and combined sensors exist they are combined

in one category; a typical application area of this sensor category is robot odometry. The considered sensors' degrees of freedom range from one to six.

GPS (Global Positioning System) modules allow the determination of one's position based on satellite signals. GPS can be used in the area of service robotics for minimizing the position error and navigation purposes. The products considered in this category encompass small integrated systems including GPS receiver, patch antenna and back-up battery. Separate reference stations typically used for differential GPS were not subsumed under this category as these would be part of the robot's environment and not of its internal structure (s. miscellaneous environment hardware below). Also excluded from this category were products optimized for usage in agriculture and automotives because they were not seen as fit for service robot due to their proprietary interfaces.

A *mobile platform* is one of the central components of a service robot and permits a robot to change its position in physical space as opposed to stationary industrial robots. Although many different platforms exist for operation in air, in or on water as well as on ground, only the latter were considered in this approach as ground-based platform make up the majority of service robot platforms; platforms for the different media are of vastly different design and thus unlikely to be reliably mapped by one cost model. Leg kinematics were also excluded due to their high complexity and little relevance for commercial service robots as of 2012. No restriction was made concerning the steering type i.e. the platform steering can be holonomic, differential or Ackerman-type. Some mobile platform stemming from the market for automated guided vehicles (AGV) are able to carry weights up to several tons but do not fit the description of service robots at the center of this work's attention; only about 27% of commercially available mobile robots weigh more than 10 kg (Piperidis 2007, p. 1). Thus the mobile platform considered here are of a maximum weight of 350 kg and a carrying capacity of 300 kg.

Robot arms are programmable devices also referred to as serial chain robots and can be described as a series of links and joints and can exhibit a resemblance to the

human arm counterpart (Siciliano 2008, p. 72).[23] Robot arms increase the reach of the service robot and can be used in combination with end-effectors for a variety of manipulation purposes. Since the focus of this work is on mobile service robots the maximum weight of robot arms considered here is constraint to 30 kg or less because their control and power supply requirements are appropriately low for mobile solutions. They can be powered by batteries as their power demand remains equal to or below 48 V and do not necessarily entail the use of a bulky control cabinet but can be controlled by an industrial PC or a similar controlling unit.

Grippers or *gripping modules* are a specific type of end-effectors aimed at grasping and holding objects i.e. they form the end of the kinematic chain that comes into direct contact with the object to be manipulated. The regression model is based on data for mechanical grippers with the number of gripper jaws ranging from two to three. Only electrically driven gripper modules were considered due to their high fitness for service robot purposes because no additional power transmission infrastructure is required as is the case for pneumatic or hydraulic solutions. Articulated hands were also excluded from regression due to the lack of data as there are only few robot hands commercially available; their high complexity is also likely to require a separate cost model.

Input peripherals allow input of commands or information to the robot by direct interaction, e.g. keyboards or buttons. Although many types of sensors can be interpreted as input devices only haptic input appliances like keyboards or joysticks were considered for the regression model as these represent the majority of input devices used for service robots and are not already subsumed under one of the other component types presented.

Output peripherals are one-way communication devices that render information externally available, e.g. warning signals, status information or feedback on user input. Due to their flexibility, robustness and intuitive interpretability visual communication devices arguably represent the most common output means. Thus,

[23]Robot arms are also referred to as manipulators; this term is ambiguous in so far as it can lead to confusion with the end-effector, e.g. gripper which is also part of the manipulating chain.

small monitors with screen resolutions up to 76,800 pixels were chosen as representatives for output peripherals.

Most of a robot's components require a *power supply*. For mobile robots the components supplying power are on-board power sources that allow robots to act without an energy tether restricting its movements. Due to its advantages in storing, space requirements, and suitability for indoor applications direct current electricity is the preferred power source. Hence, the regression model is restricted to direct current electrical power units and includes storage units i.e. batteries. The range of available batteries is extensive which is why the model estimation was restricted to a subset. For the batteries analyzed, the maximum capacity was set to 100 Ah and the minimum and maximum nominal voltage to 2 V and 15 V, respectively, as these numbers represent realistic limits for the type of service robots considered here. Because different hardware components can require different voltage levels voltage converters (DC-DC) with an output voltage up to 48 V were also subsumed under power supply components.

The *controlling unit* is the interface between robot hardware and robot software, the robot's "brain". As industrial PCs are highly versatile in their application and readily available off-the-shelf they pose an adequate solution for many service robot purposes and thus regression data is based on this type of controlling computer. This category also includes embedded PCs which are more compact and well-suited for smaller robots. Control cabinets typically used for industrial robots were not considered because they do not offer an adequate solution for mobile robots.

Safety hardware consists of those components whose primary function is to prevent the robot from harming persons in its working area. For the regression model safety switches and safety contact sensors of mat, bumper or band design were considered; safety components that had already been considered in other categories e.g. laser scanners were not included in the regression of the safety hardware cost model.

The placeholder category *for miscellaneous end effectors and tools* stands for devices that can be used for manipulation purposes by service robots but are too specialized to be considered typical service robot components. Possible examples of this category are a vacuum cleaning unit or cutting and machining tools.

Miscellaneous structure hardware is a collective term for those components not fitting into any of the categories mentioned above that can be built into the service robot. Possible examples are casing elements, internal frames or tool magazine for quick tool exchange.

Miscellaneous environment hardware covers those components that perform a supporting or auxiliary function in relation to the service robot but which are not integrated into the robot itself. Instead, they form a part of the robot's working environment. Possible examples are GPS reference stations, battery-charging stations or artificial landmarks like magnetic track tape or RFID tags.

5.1.2.3. Software Components

Similar to the hardware component categorization the software component types are classified according to their functionality. Because the EFFIROB study does not offer a catalog of generic software components similar to the hardware lists a classification was conducted based on literature research and experience values from the realization of mentioned study. The preliminary list consisting of 23 classes was separately presented to three robot engineers at Fraunhofer IPA for revision. The final list resulting from consolidation of the experts' input is displayed in Table 5.1-3. An explanation for each component is given in the following paragraphs.

Object Detection	Tool Control
Object Recognition	Arm Control
Object Modeling	Arm Path Planning 3D/6D
Environmental Modeling	Operational Interface (HRI)
Change Detection	Robot-to-Robot Coordination
Self Localization & Mapping	Communication Protocols & Messages
Platform Path Planning 2D/3D	Learning & Reasoning
Platform Control	Drivers & Primitives
Grasping & Grasp Planning	(Robot) Operating System

Table 5.1-3 Software component types

Object detection describes the detecting of an object without realizing the type of object. It is a basic feature commonly used for local path correction and safety purposes and implemented in most service robots.

Object recognition is the logical extension of object detection. It expresses the ability classification of a detected object in the robot's vicinity. In computer vision this task is accomplished by filtering the sensor data and looking for characteristic geometric features like edges, corners or other visual features like typical patterns or color gradients.

Object modeling represents the creation and interpretation of different objects as entities differentiated from the overall environment and is thus closely related to object recognition as object recognition usually relies on models of objects for comparison with its sensor data. Library of object models are subsumed under this category.

Environmental modeling is the creation and interpretation a model of the immediate surroundings and is similar to object modeling. The disparity lies in the environment being larger and often much more complex than most singular objects; it can be made up of lots of objects but can also consist of a large plane only e.g. the floor of a sizeable room. Environmental modeling can also entail the determination of topological relations between objects in the surroundings.

Change detection describes the perception of changes either in the immediate surroundings (external), within the robot (internal), or both. It requires the keeping track of states over the course of time. This function is vital if the robot is expected to manipulate its environment in a controlled and autonomous way. An application example for external change detection could be a floor-cleaning service robot which realizes when waste has been dropped on the floor and removes it.

Simultaneous localization and mapping (SLAM) is the construction and updating of a map of an a priori unknown environment and concurrently localize its own position. SLAM is being increasingly applied in service robotics due to its enhanced flexibility compared to navigating with predefined maps.

Platform path planning describes the determination of a collision-free itinerary for the robot platform in order to reach a specific point in its environment. If the path of a service robot cannot be a random pattern this skill is central to the robot's mobility. Paths for ground-based robots can be planned in two or three dimensions.

Platform control is the software enabling the robot to generate commands for the execution of platform movements and supervise the movement. Forward and inverse kinematics in conjunction with the required input from platform sensors constitute the foundation of this skill.

Similar to platform path planning *arm path planning* allows the determination of a collision-free itinerary for the robot arm in order to reach a specific point within its workspace. This task can be particularly demanding in dynamic and cluttered environments. If the path is planned as a succession of waypoints of each joint in space coordinates the planning is three-dimensional, if the planned waypoints also include orientational angles of the joints the planning incorporates six dimensions.

A robot fitted with a gripper needs to be able to grasp an object. This software generating commands for grasping movements including the planning of grip position is subsumed under the category of *grasping and grasp planning*. Depending of the shape of the object different grasping positions are possible; the according skill allows the robot to determine the most adequate one and execute the respective gripper movements.

If the robot is fitted with tools other than grippers the robot software used to generate commands for tool usage und supervise its correct functionality is represented by the category for *tool control*. As these tools can vary greatly (s. Chapter 5.1.2.2) this rather broad category spans a multitude of possible control software components.

Arm control is the analogon of the platform control category for the generation of robot arm commands and the supervision of the execution of arm movements.

The *operational interface* category covers all those software components which allow humans to interact with the robot. This includes the possibility to give commands to the robot and to receive information from it. The category encompasses graphical user interfaces (GUI), speech recognition, tele-operation and other conceivable types of human-robot interaction (HRI).

Robot-to-robot coordination components enable a robot to communicate with other robots and to synchronize collective actions. Elements of this category allow robots to cooperate and fulfill tasks specified for a team rather than for a single robot.

Communication protocols and messages are those software components that constitute the "language" of the robot i.e. the definition and implementation of standards for command and information exchange purposes. They allow internal communication processes between components of the service robot but also external communication with computers.

Reasoning is the inference of implicit information from explicit knowledge in a deductive way and derive corresponding decisions. *Learning* signifies acquiring new or modifying existing knowledge and therefore is closely connected to reasoning. To a certain degree, these software components allow a robot to integrate smoothly in its environment and accommodate unforeseen situations. This category entails reasoning algorithms, ontologies, expert knowledge databases, and similar components.

Drivers represent low-level interfaces to hardware or software components. *Primitives* are atomic instructions which can be combined to create more complex commands. Commonly, primitives are one of the main constituent of drivers.

The *operating system* provides the framework and common services for other software components to run. It handles basic operations like file management, command execution scheduling and memory allocation.

5.1.3. Structural Dependencies

Three different relation types are considered in the presented approach. Each robot skill requires certain hardware and software components in order to function properly; others might improve its compliance or extend its scope. Also, one kind of component can substitute another one in terms of functionality. Thus, the dependency types applied in this approach are "requisition", "substitution" and "enhancement" and are perceived as relations between two items.

Requisition specifies that item A requires item B. Requirements as entered in the structure matrices are considered the least requirement. Applied to skills, this means that the required item B or a better one is necessary for skill A to be satisfactorily executed; applied to hardware and software components, item B or a better one is required for the component A to function in a useful and desired way.

Skills can require hardware or software components but cannot be requirements of those categories.

Substitution describes item A's ability to functionally replace item B. This dependency plays a vital role in accounting for implementation alternatives. If item A can substitute item B but not vice versa this entails functional superiority of item A, e.g. a laser scanner can replace ultrasonic sensors but the inversion is not correct.[24] The declaration of least requirement and substitution dependencies allows the computation of design alternatives for the service robot. Skills cannot be substitutions for hardware or software components.

Enhancement represents item A directly augmenting or complementing item B's skills. Directly means the enhancement must be a proximate effect not an indirect one. An example: Object recognition using a laser scanner in conjunction with ultrasonic sensors might be superior to one relying only on ultrasonic sensors. The laser scanner does not enhance the ultrasonic sensors but the object detection. Skills cannot be enhancements of hardware or software components.

Because the dependencies defined here are unilateral two designators are necessary for each relation in order to represent the relation's direction. The according designators for the structure matrix cells are displayed in Table 5.1-4.

Designator	**Meaning**
R	<row> requires at least <column>
r	<row> is minimum requirement of <column>
S	<row> can be substituted by <column>
s	<row> can substitute <column>
E	<row> is enhanced by <column>
e	<row> enhances <column>

Table 5.1-4 Dependency type designators

[24]Ultrasonic sensors might be sufficient for a given application. In that case they are the least requirement but substitution could take place by components with higher functionality. Additional costs for higher (enhanced) functionality must be weighed against the performance demands.

Designator combinations for substitution and enhancement are logically possible, specifically for symmetric relations when mutual substitution or enhancement is possible and combinations of substitution and enhancement when a component can substitute but also increase another one's functionality. Per definition, least requirement dependencies cannot be combined with substitution or enhancement in the same cell: item A requiring item B cannot be substituted by item B; enhancement is considered optional in contrast to requirements being compulsory. The list of possible designator entries in the cells of the structure matrices is given in Table 5.1-5.

The structural knowledge on dependencies between skills and components facilitates the cost estimation as follows: After the identification of separate skills and components the relations and dependencies between them are automatically mapped and made available the estimating person via the estimation application SEROCOST (s. Chapter 7). This supports consideration of technological dependencies and an overview of possible structural alternatives without detailed technological concepts on the estimating person's part.

Designator	**Meaning**
rR	Mutual requirement
eE	Mutual enhancement
es	<row> can substitute and enhance <column>
Es	<row> can substitute and be enhanced by <column>
ES	<row> can be substituted and enhanced by <column>
sS	Mutual substitutability
eEs	Mutual enhancement, <row> can also substitute <column>
eES	Mutual enhancement, <row> can also be substituted by <column>
esS	Mutual substitutability, <row> can also enhance <column>
EsS	Mutual substitutability, <row> can also be enhanced by <column>
eEsS	Mutual enhancement and substitutability

Table 5.1-5 Possible designator combinations

5.2. Data Collection and Modeling

As explained in Chapter 4.2 structural matrices provide an appropriate means to present relationships within and between different design domains in a clearly

arranged way as means of complexity management (Maurer 2007, pp. 67–68). The domains considered in the presented work are skills, hardware components, and software components thus the interrelations between them are captured in three intradomain matrices and three interdomain matrices; the Multiple-Domain Matrix (MDM) combines all single matrices and gives an overview of all relations as depicted in Figure 5.2-1 (ibid., p. 60).

The MDM is a square matrix because the domains are arranged along horizontal and vertical axis as well. Those sections along the diagonal where rows and columns belong to the same domain constitute the intradomain matrices, the remaining sections the interdomain matrices. Because the interdomain in the lower left triangle of the MDM (dark gray areas) are the transposed matrices of the upper right triangle they do not contain additional information and thus do not need to be considered separately. The dependencies depicted are interpreted according to the rule "row affects column".

The definition of domains and domain elements determines the construction of the structure matrices and delineates the scope of the cost estimation. In order to provide support in deriving a list of robot components required for a specific application the structure matrices must be completed i.e. all those cells where relations between domain elements exist need to be filled out. To this end, a workshop was held where six experts in service robot development were requested to mark dependencies in each of the six structure matrices with the defined designators as they saw fit.

As the result of the workshop, six different versions of each structure matrix were filled out. The individual expert specifications were consolidated into one structure matrix of each type. The consolidation was conducted in three steps for each of the six structure matrices:

1. All cell entries were aggregated into an aggregation matrix where each cell could hold up to six different entries.
2. In each cell of the aggregated matrix the less frequently mentioned dependencies were removed so that only the most frequently mentions remained.

3. Each cell was checked for consistency; contradictions were eliminated manually based on the context of the two related items.

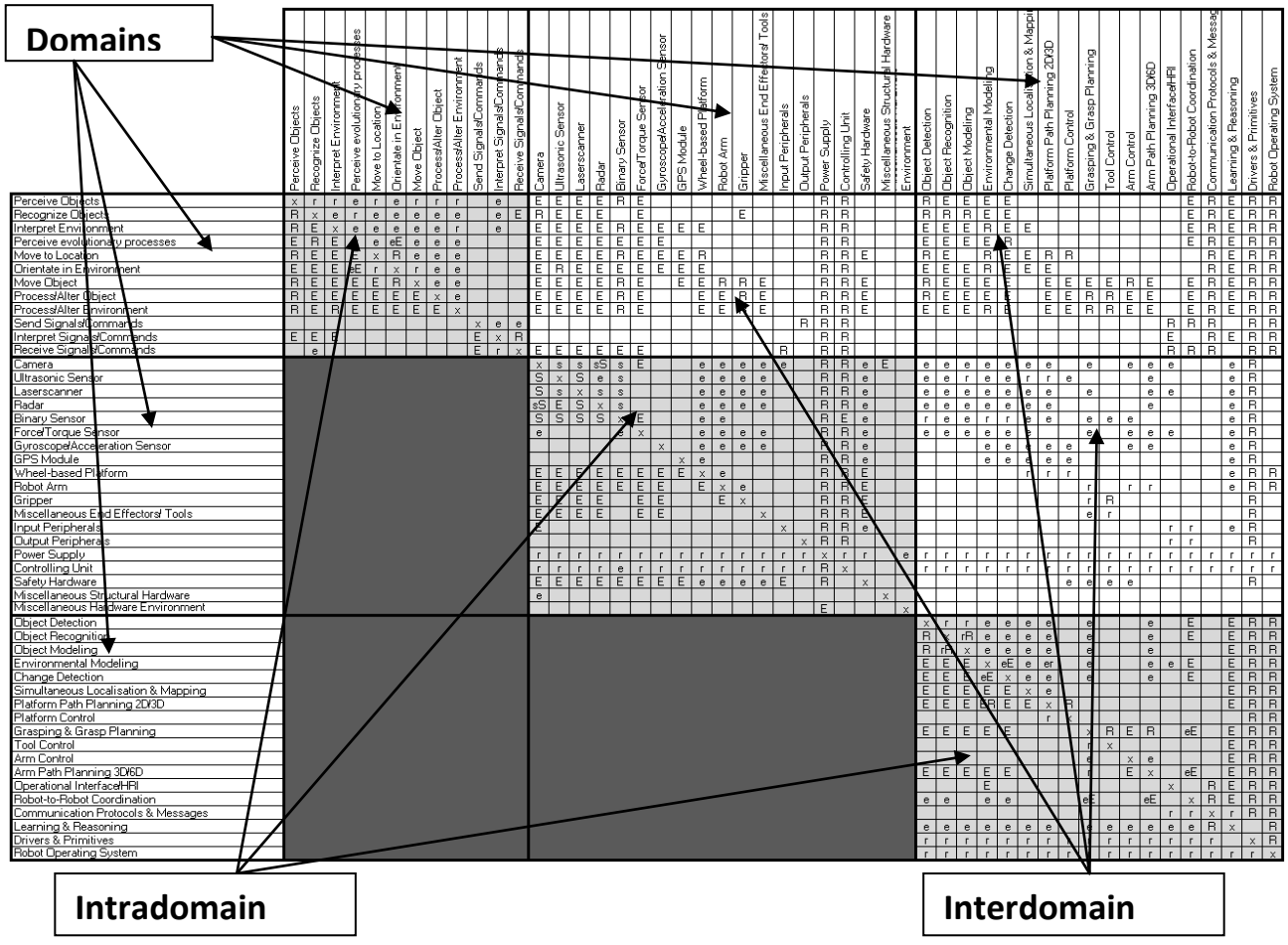

Figure 5.2-1 Multiple-Domain Matrix

In the resulting MDM 612 of 1,176 cells (transposed cells excluded) contained marked dependencies. One noticeable finding was that only seven combined entries remained after consolidation which indicates that in most cases one relation between domain elements is dominant. Furthermore, only eight substitution relations were specified.

6. Component-based Cost Estimation

In this chapter the principal cost model is laid out. The overall cost perspective is that of an entrepreneurial point of view, i.e. social or environmental effects and external costs are not considered.

6.1. Component-based Estimation Cost Model

The proposed cost estimation model for prototypical service robots is based on its main hardware and software components and contingent heuristics. The required components are a function of the skills the robot is supposed to feature; the required skills depend on the specific robot application. This modeling approach is top-down from the perspective of the technological structure (from general functionality to specific component) and bottom-up from cost perspective (from specific component to prototype cost).

The following paragraphs point out how the separate cost blocks for hardware and software components are modeled. The main advantage of estimating the costs for each component separately is the increase in accuracy as the overall variance decreases and individual components are easier to assess (Ehrlenspiel 2007a, p. 429); also, the overall estimate is rendered traceable by detailing its composition. Figure 6.1-1 depicts how the different cost blocks are combined to form the cost for the prototype.

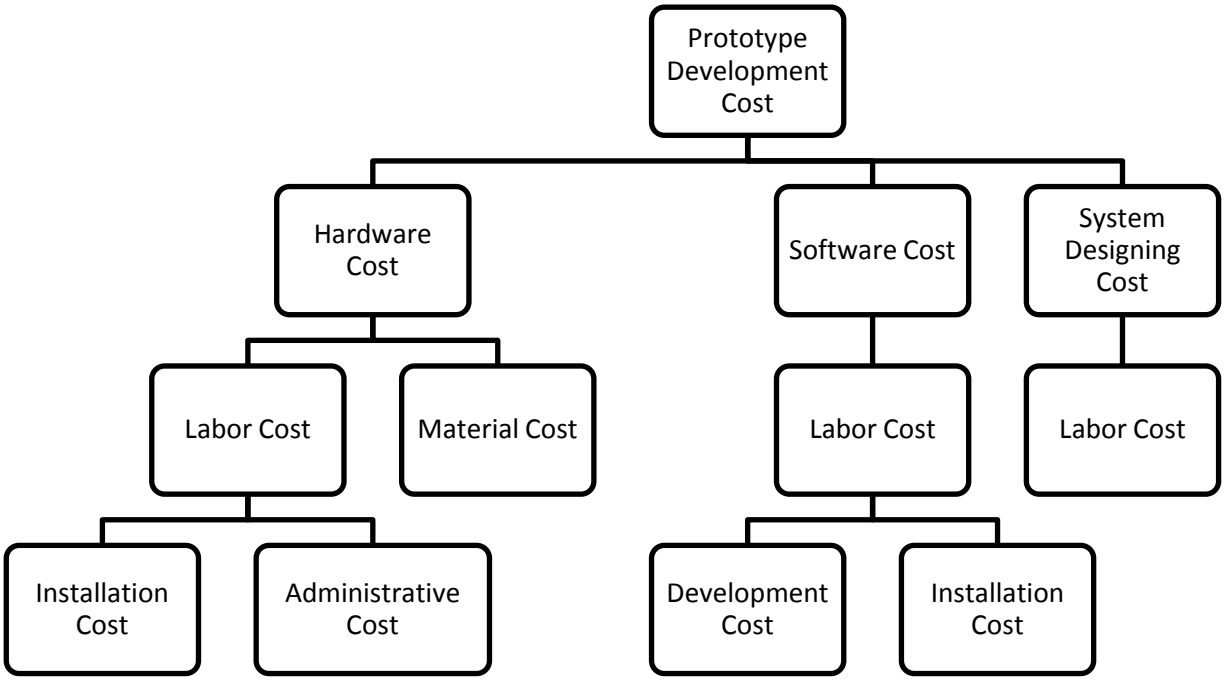

Figure 6.1-1 Cost structure of service robot prototype

6.2. Component Costs

Each cost constituent is estimated separately, similar to the Bill of Material (BOM) concept. BOM relies on detailed knowledge of final product's structure, i.e. the minute design and requirements of the final product is available. Thus, the bottom-up BOM approach does not lend itself to early phase cost estimation. Conversely to BOM, the presented approach extrapolates the list of components for a prototypical service robot yet to be built from the specified robot application, i.e. it is a top-down approach. To this end, categories for vital component types are established and cost functions attributed accordingly.

For each component a model of material and labor costs is constructed and the individual cost constituents are classified according to the cost types illustrated above. Direct costs at product unit level are calculated for one robot i.e. component quantities are related to one manufactured unit.

As some of the costs are unit variable whereas others are fixed, the following notation is used for cost designation: The lower case letter *c* represents variable costs calculated for one robot unit, the upper case letter *C* represent costs incurred for all regarded units, i.e. n_{units}. The number of units n_{units} includes the prototype and thus is greater or equal to one.

6.2.1. Hardware Component Costs

As outlined in Chapter 2.2.4, only OTS components are considered for hardware cost estimation, i.e. no proprietary hardware development costs are estimated. Thus the hardware component costs in this approach are the sum of the material acquisition costs, the installation costs for each component and the labor costs incurred by the related administrative processes e.g. selecting the appropriate component and finding a supplier.

6.2.1.1. Data Collection

The findings from the data collection processes described in Chapter 4.4 are outlined in this section. This data was used to create a hardware component cost model.

The hardware related labor cost can be divided into installation cost and the labor cost for administrative processes pertinent to the components. For the administrative labor a literature research was conducted because the inherent work steps are usually executed by different persons e.g. selection of component by engineer, purchasing by the buying department, handling and transport by logistics; the resultant increased effort of identifying the relevant processes and interviewing the responsible agents and the assumedly low impact on the total costs was deemed disproportionate to the arguably increased exactitude by expert opinions.

According to Ehrlenspiel e.a. "administrative costs for each additional part usually lie in the range of 1,500 to 2,000 [€] for purchased parts" as of 2007 (ibid., p. 138).[25] Adjusting this value by the inflation rates in Germany from 2007 to 2011 the range lies between 1,650 € and 2,200 €.[26] The average value of 1,925 € is used as a markup per component (not per component unit) and thus represents direct cost at the product level and indirect costs at the produced unit level.

[25]The German original (Ehrlenspiel 2007b) indicates a range between €1500 and €2000, whereas the translated English version states "1500-2000 [US]$".

[26]Aggregated growth for agreed wages in Germany 2007-2011: 9.95% (Statistisches Bundesamt 2012a).

In order to establish an estimation model for the cost of installation two expert interviews were conducted with robot engineers at the Fraunhofer IPA. To this end, the list of hardware component types was presented to each expert individually and each category explained. Consequently the expert was asked to estimate the minimum, most likely and maximum effort required for the installation of a typical component of each category. The installation process was explained to include mounting the device, connecting it to power supply and if applicable to data interfaces and verifying its correct operation.

For the minimum effort both interviews yielded the reply that the component installation would take "few hours" for any component. On further inquiry one reply was "three to five hours" and the other "two to three hours". One expert stated that the minimum amount of time required (in his case two to three hours) was also the most likely effort, whereas the second opinion stated was "between half and a full day" with workdays of eight hours. For the maximum time required the answers were "one week" and "one to two weeks"; in both cases these larger amounts were attributed to faulty initial installation or minor defects like broken cables and the related error finding efforts. In both interviews the experts voiced their concern that the exactitude of such estimations could only be limited as the components can vary to a considerable degree.

The process of collecting data on material acquisition costs was conducted in several steps. First, technical product information was collected for each product of the category samples. In order to identify potential input parameters for component cost estimation a list of product features and the prevalence of the features mentioned were compiled. The data was extracted from technical data sheets and product information of relevant websites. For this presample five products were randomly sampled for each hardware component category; if the technical parameters were inconclusive i.e. no dominant features were identifiable, five more products were sampled.

In order to identify the most relevant parameters they were filtered in a three step process. First, all those parameters which were mentioned in less than 60% of the component group samples were excluded. This first filtering is based on the assumption that relevant characteristics appear throughout the majority of product

descriptions of a category. The number of technical parameters considered as potential main cost drivers resulting from this presampling is presented in Table 6.2-1, a detailed list for each category can be found in Appendix 10.3.

The remaining parameters were divided into primary and secondary parameter groups. Primary parameters are those which define the components functionality to a large degree, e.g. the payload of platforms or the capacity of batteries, secondary parameters are features not central to the component's purpose e.g. the color of the casing. Although the distinction between primary and secondary parameter categories depends on the concrete application the dichotomization based on feature significance was deemed necessary for further reduction of the number of parameters. In a third step, the remaining parameters were filtered according to their statistic evaluability. All nominal parameters i.e. parameters which cannot be ranked where excluded.

With the list of preliminary cost estimation parameters determined, products matching the component type description were selected as a representative sample and the according selling prices researched i.e. material cost for components are represented by the market prices of the components. Where available, prices were taken from the manufacturer's website; otherwise they were requested via email or telephone call to the manufacturer or a supplier if the producer did not engage in direct sales. The prices are considered as per single unit order i.e. rebates on large volumes are not contemplated. Value-added tax was excluded from the prices because it is deductible for non-consumers. The currency unit for all components observed is Euro (€).

The categories *Miscellaneous End Effectors/Tools*, *Miscellaneous Structural Hardware* and *Miscellaneous Environment Hardware* are considered too heterogeneous in nature to determine a single cost model for each one of them. Therefore, no data was collected on products pertinent to these categories.

It should be noted that the spread for industrial component prices can be considerable and often depends on the buyer's negotiating power, quantity of products bought, availability, general market demand and many more; analyzing these effects is beyond the scope of this work.

Hardware Component Type	Number of technical	Number of significant
Camera	101	3
Ultrasonic Sensor	69	3
Laser Scanner	127	4
Radar	40	2
Binary Sensor	63	2
Force/Torque-Sensor	82	4
Gyroscope/Acceleration Sensor	120	5
GPS System	38	0[27]
Wheel-based Platform	66	4
Robot Arm	65	4
Gripper	54	3
Input Peripherals	75	2
Output Peripherals	93	3
Power Supply	76+23	3+3
Controlling Computer	83	4
Safety Hardware	41	2

Table 6.2-1 Component-wise counts of technical parameters[28]

The sample sizes depend on the availability of commercial products in each category and thus differ accordingly. A stark contrast can be observed between widely used products such as binary sensors and goods manufactured for smaller niche markets like robot arms. The number of sampled grippers is particularly low due to the technical constraints (s. Chapter 5.1.2.2). The category sample sizes are displayed in Table 6.2-2.

Applying the regression analysis techniques outlined in Chapter 4.4.1 each hardware component category was assigned a material cost model. An overview of the different models is given in Table 6.2-3. As shown, 15 out of 16 regression models reveal a better coefficient of determination for the regression model (RM) than for

[27]None of the technical parameters for the compared GPS modules proved cost significant, s. Chapter 6.2.1.3.

[28]Power Supply cell entries contain two values because DC-DC converters and batteries were modeled separately.

the linear model (LM) and 14 out of 16 yield a lower average estimation error which can be interpreted as sign that the regression models are a valid approach for reducing estimation uncertainty.

Hardware Component Type	**Sample Size**
Camera	333
Ultrasonic Sensor	143
Laserscanner	357
Radar	10
Binary Sensor	1,371
Force/Torque-Sensor	165
Gyroscope/Acceleration Sensor	52
GPS System	22
Wheel-based Platform	32
Robot Arm	15
Gripper	8
Input Peripherals	82
Output Peripherals	169
Power Supply	679+103
Controlling Computer	142
Safety Hardware	28

Table 6.2-2 Component-wise sample sizes

While the average estimation error of most models lies within the limits outlined in Chapter 4.4.3 it is still considerably high for some component types, notably for power supply products. Including more parameters could increase the accuracy of these models but conflicts with the lack of detailed technical data in early conceptual phases and thus reduces the applicability for early estimation purposes.

Hardware Component Type	**Final parameters**	**R^2 (LM)**	**Avg. Error % (LM)**	**R^2 (RM)**	**Avg. Error %**
Camera	frames per second, sensor diagonal	0.56	50.4	**0.65**	**42.81**
Ultrasonic Sensor	blind range, maximum operating distance	0.28	17.92	**0.28**	**17.8**
Laser Scanner	scanning angle, blind range, maximal operating distance	0.83	42.37	**0.93**	**23.45**
Radar	sensor volume	0.64	67.36	**0.93**	**27.58**
Binary Sensor	blind range, maximum operating distance	0.49	42.86	**0.58**	**37.95**
Force/Torque-Sensor	degrees of freedom, measurable moment M_z	0.79	**43.45**	**0.81**	44.98
Gyroscope/Acceleration Sensor	degrees of freedom, power consumption	0.85	68.91	**0.94**	**27.6**
GPS System	-	0[29]	25.55	-	-
Wheel-based Platform	weight, footprint size, maximum payload capacity	0.76	59.02	**0.96**	**29.21**
Robot Arm	weight, reach, maximum payload capacity	0.22	107.78	**0.85**	**28.41**
Gripper	jaw stroke	0.71	32.76	**0.74**	**31.27**
Input Peripherals	degrees of freedom, number of buttons/keys	0.09	59.63	**0.121**	**55.25**

[29]Equals zero per definition, as in this the estimated value is equaled with the mean of observed values. For this case, R^2 holds no meaningful information.

Output Peripherals	pixel array size	0.83	1131.43	**0.97**	**36.62**
Power Supply (DC-DC converter)	output voltage, output power	0.3	132.75	**0.43**	**59.19**
Power Supply (battery)	capacity, output voltage	**0.69**	**75.39**	**0.69**	**75.39**
Controlling Computer	processor type, volume, flash	0.72	42.42	**0.81**	**29.32**
Safety Hardware	weight, size of contact surface	0.948	74.08	**0.981**	**24.39**

Table 6.2-3 Comparison of hardware cost model performance

Two noteworthy cases required an adaptation of the methodology. The power supply category proved to be difficult to model as it comprises converters as well as batteries. As these are technically very different concepts no parameters were found which could be applied to one type without unduly distorting the estimation for the other. In order to keep the structure matrices synchronized for which data had already been collected these two types were kept in one category but each was assigned a separate cost model. For GPS modules no parameter remained as significant cost drivers because the price spread of the considered products was small since many products revealed very similar parameter values. Thus, the price for a GPS module is modeled as the average of the sample prices.

As a further result of the data collection process, averages and medians for each applied parameter were determined. These values provide the basis for parameter default values if the user chooses not to enter a specific value or no specific information is available. The concrete values are listed in Appendix 10.3.

6.2.1.2. Labor Costs

As each hardware component has to be installed in the service robot it also incurs labor costs. As components for service robots differ in an unwieldy number of parameters like size, technology, complexity and many more a simplified labor cost model is used in accordance with the objective of pragmatic and easy-to-use applicability (s. Chapter 2.3).

The function of estimated installation cost $C_{Install}^{HW}$ was constructed using the three-point estimation method on the averages of the expert judgments on minimum, most likely and maximum time effort required expressed in work day of eight hours working time. This yielded a mean installation time of 1.46 workdays per component with a standard deviation of 0.93 workdays. Thus the overall installation cost is estimated as

$$E(C_{Install}^{HW}) = n_{units} \cdot E(\mathrm{c}_{Install}^{HW}) = n_{units} \cdot 1.46 \cdot \frac{w_{Engineer}}{d_{month}} \cdot \sum_{\mathrm{i}=1}^{\mathrm{n_c}} n_i \qquad (6.1)$$

where n_{units} is the number of manufactured robots (including the development prototype), n_i is the quantity of the *i*-th of n_c component type instances per robot, $w_{Engineer}$ is the monthly cost rate of the person assigned to the installation and d_{month} is the number of working days per month.

Minimum and maximum values indicated in the interviews yielded averages of 0.41 and 6.25 workdays per component.

The overall hardware installation cost variance is

$$Var(C_{Install}^{HW}) = {n_{units}}^2 \cdot Var(\mathrm{c}_{Install}^{HW}) = \left(n_{units} \cdot 0.97 \sum_{\mathrm{i}=1}^{\mathrm{n_c}} n_i \right)^2 \qquad (6.2)$$

The installation costs are direct costs on the product unit level.

Components also cause secondary administrative work e.g. in the form of selecting and purchasing processes, inspection and similar activities. These are indirect costs and there are several ways to burden them on product units (s. Appendix 10.1.2.5). Also, some of these processes do not scale with each purchased component unit; the relevant cost object in most cases is either the component (in the sense of an OTS product differing from comparable ones) or batches of components as not each component unit is necessarily bought separately. It is not the aim of the presented work to capture all indirect costs but only those costs that are caused by the product's development and production. As the establishment of substantial knowledge on firm-specific details on administration organization, staffing, processes etc. required for the determination of holistic administration markups is deemed neither pragmatic nor required in the scope of this work a heuristic rule is

applied here for the estimation of the administrative costs related to hardware component.

Based on Ehrlenspiel e.a. each component is estimated to incur 1,925 € average administrative cost per component on the product level (s. Chapter 6.2.1.1) i.e. these costs are modeled as an absolute markup per component:

$$E\left(C_{Admin}^{HW}\right) = 1925 \cdot n_c \tag{6.3}$$

where n_c is the number of component type instances.[30] The variance is calculated based on the indicated range between 1,650 € and 2,200 € using three-point estimation as

$$\mathrm{Var}\left(C_{Admin}^{HW}\right) = \left(\frac{550}{6} \cdot n_c\right)^2 \tag{6.4}$$

It should be noted that the values for administrative costs for hardware components are only very rough, orientational values due to the lack of more specific data.

The overall hardware labor cost model is calculated as

$$E(C_{Labor}^{HW}) = E(C_{Install}^{HW}) + E\left(C_{Admin}^{HW}\right) \tag{6.5}$$

$$\mathrm{Var}(C_{Labor}^{HW}) = Var(C_{Install}^{HW}) + Var\left(C_{Admin}^{HW}\right) \tag{6.6}$$

With the minima and maxima adopted from data collected, the probability density functions $pdf_{Install}^{HW}$ and pdf_{Admin}^{HW} are calculated as beta distributions and subsequently convolved in order to compute their aggregate distribution pdf_{Labor}^{HW}.

The beta distributions are calculated with the *dbeta* algorithm of the R software library *msm*. Having estimated pdf_{Labor}^{HW} permits the calculation of probability quantiles which give the estimator additional information about the range of the cost estimate in addition to mean and variance.

[30]Disambiguation note: The markup is not calculated per unit of the component used but per component type. Example: If two identical laser scanners were used as components, the administrative cost is only incurred once for the type of laser scanner, not twice.

6.2.1.3. Material Costs

Applying parameter values from user input the expected value and variance of unit material cost c_{unit_i} for the *i*-th component instance can be estimated using the cost models derived by regression analysis:

$$E(c_i) = f_i^{RM}(p_1, \dots, p_{k_i}) \tag{6.7}$$

where k_i is the number of parameters for component *i*, f_i^{RM} is the generalized additive regression function for the same component. The regression model calculated with *gam* also computes the standard error i.e. standard deviation.

The default values for the parameter are set to the median of the surveyed samples pertinent to the specific parameter; the median was chosen due to its higher robustness toward outliers. Consequently, the expected cost of all hardware materials is estimated as

$$\begin{aligned} E\left(C_{Material}^{HW}\right) &= n_{units} \cdot E\left(c_{Material}^{HW}\right) \\ &= n_{units} \cdot \sum_{i=1}^{n_c} n_i \cdot E\left(c_{unit_i}\right) \end{aligned} \tag{6.8}$$

The overall variance is calculated as the sum of the component variances $Var\left(c_{unit_i}\right)$ calculated by the respective model

$$\begin{aligned} Var\left(C_{Material}^{HW}\right) &= {n_{units}}^2 \cdot Var\left(c_{Material}^{HW}\right) \\ &= {n_{units}}^2 \cdot \sum_{i=1}^{n_c} {n_i}^2 \cdot Var\left(c_{unit_i}\right) \end{aligned} \tag{6.9}$$

As the models assume a normally distributed estimation error the probability density function $pdf_{Material}^{HW}$ for material costs is calculated as a truncated Gaussian distribution. The area of the distribution is truncated at three standard deviations above and below the expected cost value; additionally, the lower bound is constraint to zero to avoid negative cost estimates. This truncation is necessary for the estimation of corridors as the normal distribution would allow theoretically infinite minima and maxima for the hardware cost estimate. The threshold of three standard deviations was chosen because with 99.74% the probability of the values

lying in the remaining interval is still sufficiently high.[31] The calculation of this probability density functions is executed with the *dtnorm* algorithm of the R library *msm* which adapts mean, variance and densities of the non-truncated distribution accordingly.

First, second and third quartiles are also calculated for $pdf_{Material}^{HW}$ in order to provide additional data on estimation uncertainty.

6.2.1.4. Aggregate Hardware Costs

With hardware labor and material costs estimated an aggregate hardware cost function can be constructed for the overall hardware costs

$$E(C^{HW}) = E(C_{Labor}^{HW}) + E\left(C_{Material}^{HW}\right) \tag{6.10}$$

$$\mathrm{Var}(C^{HW}) = Var(C_{Labor}^{HW}) + Var\left(C_{Material}^{HW}\right) \tag{6.11}$$

The probability density function pdf^{HW} is calculated by convolving pdf_{Labor}^{HW} and $pdf_{Material}^{HW}$ using the *convolve* algorithm of the R library *stats*. Figure 6.2-1 displays an example of the combination of previously calculated density functions.

6.2.2. Software Component Costs

Similarly to hardware costs, the software costs are calculated in a bottom-up approach. In distinction to the hardware components, costs are not based on individual cost models for each component type but on one general cost model which relies on the four parameters code size, productivity ratios for reusing and developing code, and reuse ratio for existing code. The reason for this different approach lies in the lack of data for a regression analysis approach: The development costs of existing software like ROS are unknown and there is no abundance of disclosed primary parameter values on the software as there is for hardware components. Thus, acknowledged software cost estimation principles are applied, i.e. the extrapolation of cost from code size, and extended with data collected from expert interview.

[31]The interval [μ-3σ; μ+3σ] covers 99.74% of sampled values from a normal distribution (Bronštejn 1995, p.633, p.974).

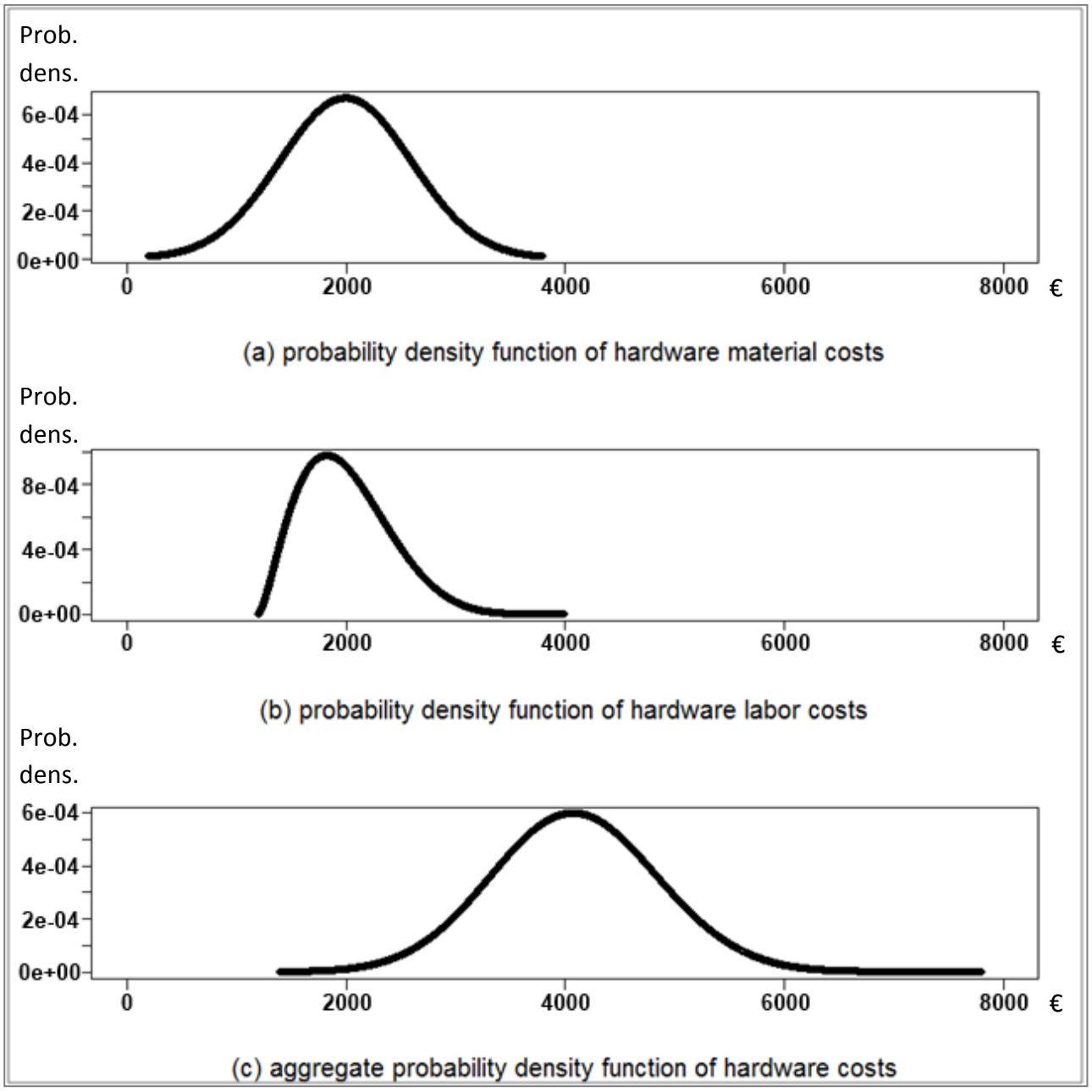

Figure 6.2-1 Aggregation of hardware probability density functions

6.2.2.1. Data Collection

For the code analysis, 469 packages from the ROS Electric Emys release and current Care-O-bot stacks as of April 2012 were chosen as the software analogy.[32] The number of meaningful code statements was extracted from each functional file

[32]For a list of all packages analyzed in the presented work s. Appendix 10.4.

pertinent to these ROS packages to be used as input for backfiring. Meaningful statements exclude comments or empty lines because they do not add functionality to the code.

Functional files are those constituting the implementation of the desired functionality and exclude those that serve the code administration itself. For example, ROS packages contain many *manifest.xml* files that serve to describe dependencies between packages (Willow Garage 2009e). These are usually not modified by third-party developers and thus were excluded from backfiring in the presented work. For similar reasons, all files that serve the code compilation, e.g. *makefile* or *CMakeLists.txt*, were also excluded. A list of files excluded can be found in Appendix 10.4.

Packages concerned with testing, documentation, and tutorials were also excluded from the analysis because the applied implementation rates cover the whole software development process including testing and documentation; considering mentioned packages would thus lead to an overemphasis of those activities. After data cleansing, the source code statements were counted for each file using the tool *cloc* (Danial 2012) and the resulting data fed into a database including count of lines and programming language.

Furthermore, each package was related to the software component types by a package-to-component relation matrix. The cell values of this matrix were not collected through expert interviews but through analysis of each package's documentation and assigning an appropriate value (Willow Garage 2008). Possible values are "package is required by component" or "package enhances component". Additionally, a package-to-package dependency matrix was created with the possible cell values "package requires package" and its inverse "package is required by package" (row to column). The values were determined using the *rosdep* utility, a small software application within ROS which provides information on software dependencies (Willow Garage).

As the number of lines of code varies highly with the respective programming language and ROS packages are programmed in many different languages[33] the collected code sizes were translated into function point estimates. To this end the latest version of the Programming Language Table (PLT) by Software Productivity Research was used (SPR 2012). The PLT contains statistic ratios for minimum, mean and maximum lines of code per function point for 550 different programming languages (SPR 2007). Applying the according conversion value the minimum, mean and maximum number of function points was calculated for each software package via backfiring. Table 6.2-4 lists the sums of function point averages per stack.

In order to tie function points to costs productivity rates are required i.e. the time one developer requires to implement one function point. Although the developer's productivity depends on many factors heuristic values can serve as default values to arrive at a plausible estimation. The heuristic values for software implementation rates employed in this work are based on Jones' research assuming that code is developed following the CBD paradigm (Jones 2007, pp. 134–145):[34]

1. Developing reusable code typically occurs at a default rate v^{D_R} of 6 function points per person month.
2. The default rate for developing non-reusable, custom code v^{D_C} is 10 function points per person month.
3. The default rate for acquiring and adapting reusable code v^{R} is 30 function points per person month. Such activities include the search for appropriate components and creating the necessary environment for reusing them e.g. installing all required miscellaneous software.

[33] 28 different languages were used for the considered 469 packages.

[34] These values are used as defaults in the estimating software tool SEROCOST pertinent to this work. As assumptions concerning these values might vary the user is offered the possibility to change any of these values thus rendering the estimation process more flexible and customizable to individual needs.

Stack	Sum
arm_navigation	3,891.4
arm_navigation_experimental	1,371.3
audio_common	43.4
bond_core	50.1
brown_remotelab	3.5
bullet	9.7
camera_drivers	191.5
client_rosjava_jni	75.7
cob_command_tools	382.6
cob_common	1,541.3
cob_driver	706.0
cob_environment_perception	293.4
cob_environments	0.3
cob_extern	88.2
cob_people_perception	22,215.2
cob_simulation	342.8
common	333.8
common_msgs	841.0
common_tutorials	202.5
control	225.3
diagnostics	49.3
diagnostics_monitors	65.2
documentation	56.3
driver_common	186.3
executive_smach	151.6
executive_smach_visualization	33.0
filters	28.3
geometry	217.0
geometry_experimental	187.4
geometry_tutorials	8.1
geometry_visualization	18.6
ias_common	570.4
image_common	56.0
image_pipeline	142.1
image_transport_plugins	26.7
imu_drivers	30.2
joystick_drivers	897.2
knowrob	4,633.0
laser_drivers	40.9
laser_pipeline	63.0
navigation	426.0
nodelet_core	56.6
object_manipulation	2,794.6
octomap_mapping	60.0

Stack	Sum
openni_kinect	119.0
orocos_kinematics_dynamics	162.9
perception_pcl	134.5
perception_pcl_addons	111.9
physics_ode	199.3
pluginlib	111.7
point_cloud_perception	1.6
pr2_apps	81.0
pr2_arm_navigation	75.0
pr2_calibration	1,063.6
pr2_common	964.2
pr2_common_actions	283.9
pr2_controllers	725.4
pr2_ethercat_drivers	273.7
pr2_gui	25.9
pr2_kinematics	45.7
pr2_mechanism	481.2
pr2_navigation	95.7
pr2_object_manipulation	3,555.7
pr2_power_drivers	119.6
pr2_robot	170.1
pr2_simulator	150.7
robot_calibration	6.4
robot_model	1,312.1
robot_model_tutorials	107.1
ros	537.3
ros_comm	1,792.6
ros_realtime	31.3
ros_tutorials	170.2
rx	540.0
schunk_modular_robotics	444.9
simulator_gazebo	3,344.9
slam_gmapping	17.7
sql_database	17.9
stage	7.0
vision_opencv	45.8
visualization	735.5
visualization_common	181.5
visualization_tutorials	19.9
warehousewg	361.8
web_interface	1,053.9
wifi_drivers	41.3
xacro	29.2

Table 6.2-4 Average function points per stack

These productivity ratios include all software development activities ranging from early requirement analysis to delivery and thus constitute rather slow development rates. If activities are skipped or heavily reduced e.g. by omission of thorough testing the rates will rise with accompanying quality losses.

Furthermore, the development team's experience has a strong impact on the software productivity rates (McConnell 2006, p. 63; Jones 2007, p. 340). For this reason the estimator may want to include his assessment of the team skill in an attempt to increase accuracy of estimates. In order to provide this possibility without introducing increased complexity to the estimating approach a set of adjustment factors depicted in Table 6.2-5 was extracted from Jones' research (Jones 2007, pp. 337–342). The default productivity rates for software development are multiplied with the selected factor yielding the applicable rates v^R and v^D.

Team capability	**Multiplier**	**Impact compared to average**
Excellent	1.55	+55%
Good	1.28	+28%
Average	1.0	0%
Below average	0.56	-44%
Poor	0.13	-87%

Table 6.2-5 Team capability adjustment factors

As not all of the software is to be rewritten according to the assumption of code reuse in Chapter 2.2.5 data on estimated reuse ratios was collected. To this end, a list of the 469 packages was presented to two experts at Fraunhofer IPA familiar with ROS in order to estimate reuse rates using the scale described in Chapter 4.4.4.[35] Several rules were applied in order to aggregate expert judgments into one reuse ratio for each package:

1. Where the expert could not estimate reuse because he was unfamiliar with it a default value of 0.6 was determined. This value is a heuristic rule for component-based development based on software estimation literature

[35]Both experts are members of the "Extended Authors (Stacks Released)" group, s. (Willow Garage 2009a).

stating that roughly 60% of the code stems from reusable components (ibid., p. 144).

2. If the expert indicated that a component was merely installed but not adapted the reuse ratio was interpreted as 100% and the packages were marked in the database so that they are exempt from the estimation software development cost.
3. If the judgments were identical the median between lower and upper interval limit were set as the reuse ratio e.g. the interval between 30% and 50% reuse is translated into a reuse ratio of 0.4.
4. If the expert opinions differed the average of the respective interval medians was set as the reuse ratio e.g. one judgment of 30% to 50% and one of 10% to 30% equal a reuse ratio of 0.3. There is one exception to this rule: If one expert indicated lack of knowledge (package not used or unknown, rule 1) and the other specified "installation only" (rule 2) no average was used but the package regarded as described in rule 2.

Figure 6.2-2 displays the absolute counts of all occurring reuse ratios thus determined. One aspect that can be drawn from the presented figure is the dominance of high reuse ratios. More than 50% of the package for which an estimate was given revealed an average reuse ratio of 0.8 or higher i.e. more than 50% are only slightly modified. Packages that are not modified at all but only installed represent another large fraction with 28%. For roughly a third of the packages analyzed the reuse ratio is 0.6. The reason for this high value is that for 132 packages the experts indicated that they did not know or use these packages so that their reuse ratios were defaulted to 0.6 (s. rule 1).

The finished software must also be installed on each manufactured robot. Thus experts were interviewed to determine the costs of an individual installation effort not related to software development. Not knowing the concrete implementation of neither the robot nor the robot software the experts were asked to estimate the minimum, most likely and maximum effort expressed in how long it would take them to install the ROS release version on a service robot like Care-O-bot 3.

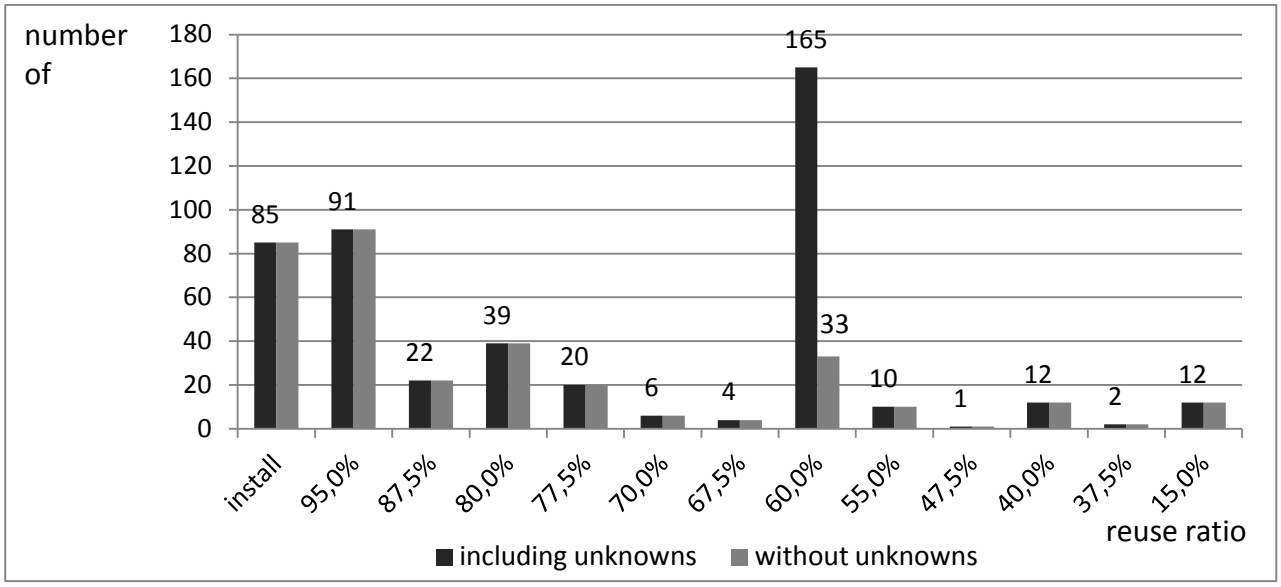

Figure 6.2-2 Distribution of package reuse ratios

Care-O-bot 3 was chosen as an appropriate analogy because it runs on ROS components, many software stacks have been released for it, the experts were familiar with its structure and it fits the definition of service robots outlined in Chapter 2.1.1. The installation effort to be estimated should also include brief testing of correct functionality. Both experts specified one workday as the most likely value and half a workday as the minimum. For the maximum one expert indicated "between two and three days", the other suggested three workdays; the worst case was thus set to three workdays.

6.2.2.2. Material Costs

Software per se is not physical but it requires storage and computing media. As these requirements are already covered by the hardware for the central computing unit (s. Chapter 5.1.2.2 on control unit) no material costs are accounted for in this regard.

In a broader sense of material costs the acquisition costs for bought software can be regarded as material costs as they can be depreciated according to many accounting standards.[36] As the presented work assumes the usage of freely available open

[36]E.g. UK Income Tax Act, UK Companies Act, German regulations on tax matters (EStR).

source code as the basis for the robots software development there are no expenses in the sense of this software material cost interpretation. Thus, material costs for software are considered to be nil.

6.2.2.3. Labor Costs

As software is reproducible in virtually any number it is not produced on a per unit basis. Thus the direct labor costs of software per service robot mainly consist of software installation costs. Although installation usually is automated for larger production series it appears plausible that the installation of software and the verification thereof for prototypical or early series service robots will be handled per individual unit in accordance with the assumption of small-series production (s. Chapter 2.2.1). According to data collected in interviews on software installation, the average time required for installing the finished software on the robot is one workday, in the best case half a workday and in the worst case three days (s. above). Using PERT calculation, one installation is thus expected to last

$$E(t_{install}) = \frac{0.5 + 4 \cdot 1 + 3}{6} = 1.25 \tag{6.12}$$

with a variance of

$$Var(t_{install}) = \frac{3 - 0.5}{6} = 0.4167 \tag{6.13}$$

The resulting estimated costs $C_{Install}^{SW}$ for the software installation including functional verification on n_{units} manufactured units (including the prototype) are thus calculated by multiplying the required time with the developer's daily wage rate. The wage rate is derived as the monthly salary $w_{Developer}$ divided by the average working days per month d_{month}.

$$\begin{aligned} E\left(C_{Install}^{SW}\right) &= (n_{units} - 1) \cdot E\left(c_{Install}^{SW}\right) \\ &= (n_{units} - 1) \cdot \frac{w_{Developer}}{d_{month}} \cdot 1.25 \end{aligned} \tag{6.14}$$

and its variance as

$$\begin{aligned} Var\left(C_{Install}^{SW}\right) &= (n_{units} - 1)^2 \cdot Var\left(c_{Install}^{SW}\right) \\ &= \left((n_{units} - 1) \cdot \frac{w_{Developer}}{d_{month}} \cdot 0.4167\right)^2 \end{aligned} \tag{6.15}$$

assuming that a software expert undertakes the process and that the software can be installed without modification at the production stage. These costs are incurred

at product unit level except for the initial prototype which is why n_{units} must be reduced by one if the prototype is included in the total number of produced units. The reason why the initial prototype is excluded from this form of installation cost is that the prototype will be directly connected to the software development process so that the working software is already installed when the development process is complete.

A large cost factor stems from software components that need to be developed first. Software development costs are activity costs which are not direct costs at the robot unit level but can be regarded as direct cost on the product level. Nevertheless, if software development costs are burdened on produced units, software costs constitute a considerable part of the production costs (s. Chapter 10.1.3 for comparison of direct and indirect costs).

Labor cost estimates are based on assessing the amount of effort that is put into activities. Thus an appropriate measure for effort has to be determined. Most software estimation approaches assume the size of software as the dominant actuating variable on the required efforts; the most common recognized measures for size being source lines of code statements (SLOC or LOC) and function points (FP) (Leung 2002, pp. 309–310; CSE 2000, pp. 3–7). In the presented work a combination of both LOC and FP measures is applied using the backfiring method.

As software development can be very complex the effort inherent to software projects depends on a multitude of parameters e.g. experience and size of the development team, time schedule, available tools, project complexity and many more; Jones lists 25 different factors affecting software development efforts (Jones 2007, pp. 341–342). For a model to be pragmatically applicable the number of parameters should be reduced without extensively compromising its informative value.

The parameters used in the presented approach are software function points, software reuse ratio, each collected per component of available robot-related software, and function point implementation rates depending on team experience.

The *function point sizes s* is the quantitative metric for the size of software as described above.

The *software reuse ratio r* is the relation of how much of the code is reused to the overall size of the software. This represents the application of the code reuse assumption (s. Chapter 2.2.5) in combination with the analogy-based estimation model for software (s. Chapter 3.1.3). In robotics, component-based software engineering (CBSE) or component-based development (CBD) are increasingly recommended and applied, as they allow systematic reuse of software elements and thus save time, effort and eventually cost (Brugali 2009; Brooks 2005, p. 135).

The *function point implementation rate v* describes the average time per function point that is required for rendering usable software code. Those packages that only require installation according to expert opinion do not impact the development cost but are listed among the software packages for completeness' sake.

As the developments costs are mainly employment expenditures the estimated overall outlay for software development C_{Devel}^{SW} can be approximated by the staff time required multiplied with the developer's monthly salary $w_{Developer}$. Staff time can be interpreted as the time one single person would require implementing the respective software.

$$E\left(C_{Devel}^{SW}\right) = E(t_{SW}) \cdot w_{Developer} \tag{6.16}$$

As software is assumed to be developed component-wise t_{SW} is the sum of the staff time required for each component. However, because components can share functionalities and thus use the same software parts the software development is treated on a level of finer granularity than the software components presented in Chapter 5.1.2.3, namely on package level. The ROS package structure allows a more detailed customization of analogy-based estimation.

Based on the selection of components on the higher level the necessary packages are retrieved from the database and the time for the total development extrapolated from the aggregated code size. This procedure also ensures that packages occurring in several components are only accounted for once.

$$E(t_{SW}) = \sum_{i} E\left(t_{SW_i}\right) \tag{6.17}$$

The staff time required for each package *i* is composed of two summands; one for the time required for acquiring and adapting reusable code $t_{SW_i}^{R}$ and one for developing new code $t_{SW_i}^{D}$.

$$E(t_{SW_i}) = E(t_{SW_i}^{R}) + E(t_{SW_i}^{D}) \tag{6.18}$$

This distinction is necessary because the development of software occurs at different implementation rates; adapting reusable software is usually faster than developing new code from scratch. Consequently, the costs per function point depend on the manner of implementing it – reuse itself tends to be cheaper than custom implementation. Writing reusable code however is more expensive than writing non-reusable code because of the need of greater care for debugging, documentation and interface compliance. The decision to make the custom code or parts of it reusable or not is subject to the developer's design philosophy.

Applying these values in conjunction with the size of the package derived by analogy-based estimation and the respective reuse ratio the staff time summands can be formed as

$$E(t_{SW_i}^{R}) = \frac{E(s_i) \cdot E(r_i)}{v^{R}} \tag{6.19}$$

and

$$E(t_{SW_i}^{D}) = \frac{E(s_i) \cdot (1 - E(r_i))}{v^{D}} \tag{6.20}$$

where $E(s_i)$ is the expected size of package *i* in function points, $E(r_i)$ is the expected reuse ratio of the package (i.e. the percentage of software which is assumed to be reused) and v^{D} is either v^{D_C} for custom code development or v^{D_R} if the newly developed code is to be reusable.[37]

The software reuse ratio r_i indicates the part of available software packages that must be adapted and fitted to the concrete application to be implemented; it is the relative amount of reuse as opposed to the amount to be developed (cf. Frakes 1996). As not all parts of the code are reusable the remaining $(1 - E(r_i))$ must be

[37]For simplicity's sake the possibility of combining development of reusable and non-reusable code is not considered.

rewritten. Software that is installed without any modification is not considered reused and thus not implied in the development costs.

The estimated time effort required for software development of package *i* is calculated as

$$E(t_{SW}) = \sum_{\mathrm{i}} \left(\frac{E(s_i) \cdot E(r_i)}{v^R} + \frac{E(s_i) \cdot \left(1 - E(r_i)\right)}{v^D} \right) \tag{6.21}$$
$$= \left(\frac{1}{v^R} - \frac{1}{v^D} \right) \cdot \sum_{\mathrm{i}} E(s_i) \cdot E(r_i) + \frac{1}{v^D} \cdot \sum_{i} E(s_i)$$

In order to determine the variance of the software development costs its probability density function must be established by calculating the aggregates of the variances of sizes and reuse ratios of software packages. The implementation rates v^D and v^R are assumed to be constants because the uncertainty towards implementation is represented by the aforementioned variances.

The wage rate $w_{Developer}$ is assumed to be constant in the presented approach; it strongly depends on regional influences and must be known to the estimating person beforehand (s. Chapter 4.3.2).

In order to establish the expected value and variance of reuse ratios the following construction model is used:

Minimum of reuse is always 0% and maximum always 100%. The most likely reuse rate MLC_{reuse_i} was derived by expert elicitation for each software package individually as described in the previous chapter.

Applying these values to (4.19) and (4.20) yields

$$E(r_i) = \frac{4MLC_{reuse_i} + 1}{6} \tag{6.22}$$

and

$$Var(r_i) = \frac{1}{6} \tag{6.23}$$

Expected value and variance of the function point size $s_i$of a software package are modeled as follows:

The lines of code counted for the component are converted to minimum, mean and maximum sizes using statistic conversion rates from the SPR Software Programming

Languages Table (SPR 2007). As the distributions of function points varies with the programming language and software packages can be composed of files implemented in different programming languages the expected function point size value of each software package is determined by calculating its language-specific distributions and consequently adding them. The cumulative expected value of the function point size s_i of software package i is

$$E(s_i) = \sum_j E\left(s_{i_j}\right) \tag{6.24}$$

where s_{i_j} denotes the function point size for the part of package i implemented in programming language j. It is calculated as

$$E\left(s_{i_j}\right) = LOC_{i_j} k_j \tag{6.25}$$

where LOC_{i_j} represents the number of lines of code written in language j which have been counted for software package i. The conversion rate k_j is the mean ratio of function points per line of code (s. Chapter 4.4.2.2 on backfiring).

Assuming that the language specific implementation sizes are independent from each other the overall variance of the package's size is

$$Var(s_i) = \sum_j Var\left(s_{i_j}\right) \tag{6.26}$$

The variance $Var\left(s_{i_j}\right)$ is determined applying equation (4.20), i.e. using the PERT estimation method:

$$Var\left(s_{i_j}\right) = \left(\frac{(WC_{i_j} - BC_{i_j})}{6}\right)^2 \tag{6.27}$$

The terms $WC_{i_j}$and BC_{i_j} denote worst case and best case for the function point size of the part of package i specific to language j, i.e. the largest and smallest sizes to be anticipated. They are calculated in analogy to (6.32) but with the respective conversion rates $k_{l_{min}}$ and $k_{j_{max}}$ for conversion to minimum and maximum function point size:

$$WC_{i_j} = LOC_{i_j} \cdot k_{j_{max}} \tag{6.28}$$

$$BC_{i_j} = LOC_{i_j} \cdot k_{j_{min}} \tag{6.29}$$

With the variances for code sizes and reuse ratios thus modeled and assuming that the reuse ratio per package is unrelated to the size of the package, i.e. $Cov(s_i, r_i) = 0$, the variance of the time effort per component is calculated as[38]

$$Var(t_{SW}) = \sum_i \left(\left(2\mathrm{Var}(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D} \right)^2 + \mathrm{E}(r_i)^2 \cdot \left(\frac{1}{v^R} - \frac{1}{v^D} \right)^2 + \frac{2}{v^R v^D} \cdot \mathrm{E}(r_i) + \left(\frac{1}{v^D} \right)^2 \right) \cdot \mathrm{Var}(s_i) + \mathrm{Var}(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D} \right)^2 \cdot \mathrm{E}(s_i)^2 \right) \tag{6.30}$$

The overall expected cost variance for software development is then calculated as

$$Var\left(C_{Devel}^{SW}\right) = \left(w_{Developer}\right)^2 \cdot Var(t_{SW}) \tag{6.31}$$

6.2.2.4. Aggregate Probability Density Function

For the derivation of the beta distribution for software development the minimum and maximum of the development cost must be modeled, too. They are determined by best case and worst case values for code size and reuse ratio:

$$BCC_{Devel}^{SW} = \frac{w_{Developer}}{v^R} \cdot \sum_i \sum_j BC_{ij} \tag{6.32}$$

$$WCC_{Devel}^{SW} = \frac{w_{Developer}}{v^D} \cdot \sum_i \sum_j WC_{ij} \tag{6.33}$$

i.e. in the best case the smallest assumable amount of function points is completely being reused, in the worst case the largest assumable amount of function points must be completely redeveloped.

With minimum, maximum, expected value and variance modeled the beta distribution pdf_{Devel}^{SW} of the software development cost can be derived according to the method outlined in Chapter 4.4.3. The distribution function allows the calculation of quartiles as a measure of estimation uncertainty.

[38] For a detailed derivation s. Appendix 10.2.

If the costs are to be estimated for more manufactured units than the development prototype additional software installation cost must be incorporated. To this end, the beta distribution $pdf_{Install}^{SW}$ of the installation cost per unit is constructed, employing worst case and best case estimates for the calculation of minimum and maximum cost:

$$WCC_{Install}^{SW} = 2.75 \cdot \frac{w_{Developer}}{d_{month}} \cdot (n_{units} - 1) \tag{6.34}$$

$$BCC_{Install}^{SW} = 0.5 \cdot \frac{w_{Developer}}{d_{month}} \cdot (n_{units} - 1) \tag{6.35}$$

Expected value and variance of the overall software costs are the sum of development and installation costs are

$$E(C^{SW}) = E\left(C_{\text{Install}}^{HW}\right) + E\left(C_{Devel}^{SW}\right) \tag{6.36}$$

$$\text{Var}(C^{SW}) = Var\left(C_{Install}^{SW}\right) + Var\left(C_{Devel}^{SW}\right) \tag{6.37}$$

The distributions pdf_{Devel}^{SW} and $pdf_{Install}^{SW}$ are subsequently folded to determine the overall distribution pdf^{SW} of the software cost.

6.3. System Designing Costs

There are cost components that cannot be attributed to specific components but only to the product as a whole, i.e. indirect product unit costs but direct product costs; because these costs concern the overall product as a system they are referred to as systemic costs in this work. The major constituents of these are the costs of the project management, overall system design and engineering process and the costs of system testing (Kossiakoff 2003, p. 92); the costs for software design and testing are already covered by the software productivity rates v^D and v^R and thus not considered in this section. Further project management costs other than those already included in the hardware and software estimates are not considered under systemic costs. The remaining systemic cost constituents are subsumed here under the term 'system designing cost'.

6.3.1. Data Collection

Estimating these costs is particularly difficult because lack of data for these costs arguably due to general issues of cost allocation and the companies' vested interest in non-disclosure of such information (Schehl 1994, p. 217). The complicatedness of estimating these systemic costs is aggravated even more as the process of overall

system design is usually not a closed, clearly segregated course of action but intricately interwoven with other parts of the development process.

In order to acquire at least indicative value of development costs other than hardware and software costs literature and database research on the topic of development cost allocation was conducted. No experts were interviewed as it seems questionable if they are willing or capable of disclosing such information given the constraints of market competitiveness. Secondary literature on costing was searched and the databases *statista* (statista 2012), *destatis* (Statistisches Bundesamt 2012b), *Eurostat* (COMM/ESTAT) and *Research Data Centres of the Federal Statistical Office* and the *statistical offices of the Länder* (NRW 2002) were scanned for information on research and development cost allocation.

One finding of the research was that detailed information is very sparse. If information on companies' research and development cost is provided it is either in absolutes for industry branches (e.g. Wiechers 2012), in relation to turnover or in relation to total costs (e.g. Meisl 1988; VDMA 2009). Absolutes and turnover-related numbers hold little information value for estimation purposes of this work and were thus not considered further. Sources of research and development costs in relation to total costs are displayed in Table 6.3-1.

Source	**Industry**	**R&D percentage of total cost**
(Meisl 1988, p. 97)	Defense	15%
(Ehrlenspiel 2007a, p. 397)	Mechanical engineering	8.3%
(VDMA 2009, p. 67)	Mechanical engineering	8.1%
(Schehl 1994, p. 219)	Industries with low R&D	1 to 3 %
	Industries with average	5 to 8.5 %
	Industries with high R&D	10 to 25%

Table 6.3-1 Mentions of research and development proportion of total costs

For the mechanical engineering industry a more detailed cost allocation was found that explains the average overhead cost structure in relation to the total costs. The according cost hierarchy is depicted in Figure 6.3-1 (adopted from Ehrlenspiel 2007a, p. 397, percentages derived from VDMA 2009, p. 67; the sum of all percentages is 100.1% due to rounding errors).

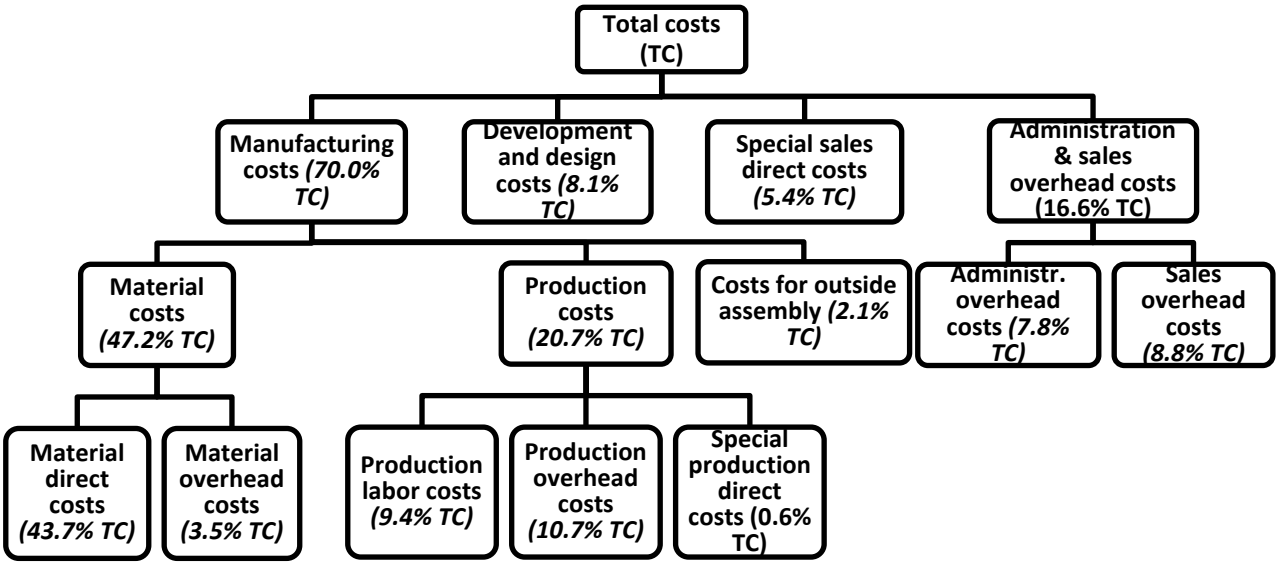

Figure 6.3-1 Cost structure in the machine engineering industry

6.3.2. System Designing Cost Model

Given the lack of data it is arguable if any attempt at estimating robot designing costs should be made at all. As long as the basis of the estimates is fully disclosed it is deemed more useful to the end of the objective of this work – namely decision-making support – to provide estimates with a high degree of uncertainty rather than omitting estimates. If the estimation model for the particular estimate is considered by the estimator to be implausible it can be foregone; omission of an estimate, however, can lead to a false impression of completeness.

In order to provide at least an orientational value for robot designing costs they are modeled as a markup α on cost incurred for hardware and software of the prototype:[39]

$$E(C^{RD}) = \alpha \cdot \left(E\left(c^{HW}_{Material}\right) + E(c^{HW}_{Install}) + E\left(C^{HW}_{Admin}\right) + E\left(C^{SW}_{Devel}\right)\right) \quad (6.38)$$

Because no data is available as a default value for α a model that draws on the knowledge of total cost ratios is applied. Average percentages of development costs and manufacturing costs in relation to total costs (*TC*) are available from secondary

[39]Notice that $C^{SW}_{Development}$ is used instead of C^{SW} because software installation costs are not separately incurred for the prototype; s. (6.12).

literature. If the robot development project is regarded as if it were a separate company it can be hypothesized that similar percentages apply accordingly. Furthermore, if the costs $E\left(c_{Material}^{HW}\right) + E\left(C_{Admin}^{HW}\right) + E\left(C_{Devel}^{SW}\right)$ estimated so far are regarded as preliminary manufacturing costs *MC* and the robot designing costs as the development and design costs *DDC* of the project according to Figure 6.3-1, the known ratios can be used as a calculation basis for the derivation of markup α:

$$E\left(c_{Material}^{HW}\right) + E(c_{Install}^{HW}) + E\left(C_{Admin}^{HW}\right) + E\left(C_{Devel}^{SW}\right) \cong MC \quad (6.39)$$

$$E(\mathrm{C^{RD}}) \cong DDC \quad (6.40)$$

If β and γ are the given ratios

$$\beta = \frac{MC}{TC} \quad (6.41)$$

$$\gamma = \frac{DDC}{TC} \quad (6.42)$$

then α is approximated by the quotient of γ and β:

$$E(\mathrm{C^{RD}}) \cong DDC = \frac{\gamma}{\beta} \cdot MC = \alpha \cdot MC \quad (6.43)$$

Employing values from Table 6.3-1 a corridor can be determined. Because service robotics is considered a research intensive high-technology industry the interval for industries with high research and development rates as indicated by Schehl appears to be a cogent choice. Thus the minimum value for γ is assumed to be 0.1 and the maximum value 0.25. As the percentage of manufacturing costs decreases with an increase of the development costs β must be adapted accordingly. Using the average values depicted in Figure 6.3-1 as a reference i.e. $\gamma_{ref} = 0.081$, $\beta_{ref} = 0.7$, β can be scaled in relation to γ as

$$\beta = \frac{(1-\gamma) \cdot \beta_{ref}}{\left(1-\gamma_{ref}\right)} \quad (6.44)$$

Thus, the corridor for α is calculated as

$$\alpha_{min} = \frac{0.1}{0.686} \approx 0.146 \quad (6.45)$$

$$\alpha_{max} = \frac{0.25}{0.571} \approx 0.438 \quad (6.46)$$

which means that systemic costs add between 14.6% and 43.8% to the hardware and software related costs according to this model.

Because no information was available on the exact distribution of development and design costs a beta distribution pdf^{RD} with a positive skew is assumed as these costs are mainly labor costs; the expected value is assumed to lie at one third of the corridor range

$$E(\alpha) = \alpha_{min} + \frac{\alpha_{max} - \alpha_{min}}{3} \approx 0.243 \tag{6.47}$$

The variance of α is determined by applying (4.20) as

$$Var(\alpha) = \left(\frac{\alpha_{max} - \alpha_{min}}{6}\right)^2 \approx 0.049^2 \tag{6.48}$$

With this auxiliary model the expected value of the robot designing costs is

$$\begin{aligned} E(C^{RD}) = {} & E(\alpha) \\ & \cdot \left(E\left(c^{HW}_{Material}\right) + E\left(c^{HW}_{Install}\right) + E\left(C^{HW}_{Admin}\right) + E\left(C^{SW}_{Devel}\right)\right) \end{aligned} \tag{6.49}$$

and the variance is derived as

$$\begin{aligned} Var(C^{RD}) &= Var\left(\alpha \cdot \left(c^{HW}_{Material} + c^{HW}_{Install} + C^{HW}_{Admin} + C^{SW}_{Devel}\right)\right) \\ &= Var(\alpha) \cdot \left(E\left(c^{HW}_{Material}\right) + E\left(c^{HW}_{Install}\right) + E\left(C^{HW}_{Admin}\right) \right. \\ &\quad \left. + E\left(C^{SW}_{Devel}\right)\right)^2 \\ &\quad + \left(Var\left(c^{HW}_{Material}\right) + Var\left(c^{HW}_{Install}\right) + Var\left(C^{HW}_{Admin}\right) \right. \\ &\quad \left. + Var(\alpha) \right. \\ &\quad \left. \cdot \left(Var\left(c^{HW}_{Material}\right) + Var\left(c^{HW}_{Install}\right) + Var\left(C^{HW}_{Admin}\right) + Var\left(C^{SW}_{Devel}\right)\right)\right) \end{aligned} \tag{6.50}$$

As noted earlier the presented model for robot designing costs is based on many unconfirmed assumptions; the effort required to validate these is beyond the scope of this work which is why this particular model is considered a tentative attempt to alleviate lack of a detailed data basis. The results, however, were included in the validation process explicated in Chapter 8.

6.4. Prototype Development Costs

The models for the separate cost blocks are combined into an aggregated cost model for the development of a service robot prototype. According to Figure 6.1-1 the manufacturing cost function then takes the form

$$\begin{aligned} c^{Prototype} &= c^{HW} + C^{SW} + C^{RD} \\ &= c^{HW}_{Install} + C^{HW}_{Admin} + c^{HW}_{Material} + C^{SW}_{Devel} + C^{RD} \end{aligned} \tag{6.51}$$

The software installation costs $C_{Install}^{SW}$ are not among the summands due to the reasons explained in Chapter 6.4.

The expected value and variance of the prototype costs is calculated as the sum of the expected values and variance, respectively for each cost subfunction.

Convolving the pertinent probability density functions yields the aggregate probability density function $pdf^{Prototype}$ which permits the localization of probability quantiles of costs for a robot prototype.

6.5. Derivation of Unit Costs

Although development of prototypes lies in the focus of the presented approach unit cost estimates are also modeled as they have the potential of providing additional insight to the estimator. For the calculation of unit costs i.e. costs per produced unit two different cost levels are computed in accordance with Figure 6.1-1. The first unit cost level *DUC* includes only direct costs at the unit level which means all development cost are exempt from this calculation. At this level the costs per unit are

$$DUC = c_{Install}^{HW} + c_{Material}^{HW} + c_{Install}^{SW} \tag{6.52}$$

As no economies of scale are considered in this approach the *DUC* are also the marginal costs i.e. costs incurred per increment in produced units.

Expected value and variance of DUC are calculated as the sum of the expected values and variances of the respectively pertinent cost blocks:

$$E(DUC) = E(c_{Install}^{HW}) + E\left(c_{Material}^{HW}\right) + E\left(c_{Install}^{SW}\right) \tag{6.53}$$

$$Var(DUC) = Var(c_{Install}^{HW}) + Var\left(c_{Material}^{HW}\right) + Var\left(c_{Install}^{SW}\right) \tag{6.54}$$

The second unit cost level *PUC* incorporates those costs that are directly related to the product but do not vary with the number of produced units; these costs are direct costs at the product level but indirect costs at the product unit level (s. Chapter 10.1.3.1). For the calculation of the *PUC* all previously calculated costs in this work are distributed among the number of produced units except the prototype unit; the costs of the prototype are burdened on the remaining $n_{units} - 1$ units.

$$PUC = \left(C_{Install}^{HW} + C_{Admin}^{HW} + C_{Material}^{HW} + C_{Install}^{SW} + C_{Devel}^{SW} + C^{RD}\right) \cdot \frac{1}{n_{units} - 1} \tag{6.55}$$

Accordingly, the expected value and variance are

$$E(PUC) = \left(E(C_{Install}^{HW}) + E\left(C_{Admin}^{HW}\right) + E\left(C_{Material}^{HW}\right) + E\left(C_{Install}^{SW}\right) + E\left(C_{Devel}^{SW}\right) + E(C^{RD})\right) \cdot \frac{1}{n_{units} - 1} \tag{6.56}$$

$$Var(PUC) = \left(Var(C_{Install}^{HW}) + Var\left(C_{Admin}^{HW}\right) + Var\left(C_{Material}^{HW}\right) + Var\left(C_{Install}^{SW}\right) + Var\left(C_{Devel}^{SW}\right) + Var(C^{RD})\right) \cdot \left(\frac{1}{n_{units} - 1}\right)^2 \tag{6.57}$$

The probability density functions for DUC and PUC are computed by convolving the corresponding distributions of the cost constituents.

Whereas the DUC properly reflect the cost causation per unit, the PUC is more holistic in the sense that all product-related costs are incorporated. Both values are amenable to further analysis e.g. cost-plus-pricing or comparison with target costs thus providing input for information-based decision making. It should be noted, however, that the unit costs derived with the described method cannot be seen as realistic estimates for larger series production where cost-saving mass production means are employed that are not considered here.

7. The Software Tool SEROCOST

As a part of this thesis, a software tool was developed in order to implement a method of computer aided application of the estimation methods described in the previous chapters. A description of this tool is presented in this chapter.

7.1. Aim

The declared objective of the presented work is to support service robot developers in their decision-making process. In order to fulfill the pertinent estimator requirements (s. Chapter 2.3) a software application has been developed in conjunction with the presented thesis. This tool called "SEROCOST" (short for "Service Robot Cost Estimation") was designed to serve several purposes:

- Provide access to the data and cost models forming the basis of the presented estimation approach
- Process provided data and user input according to the presented methods
- Present results in an aggregated and comprehensible way to the estimator

The application should warrant ease-of-use and applicability to reduce estimation complexity; SEROCOST was designed with the goal in mind to allow a quick estimate for service robot prototypes in less than one hour. Furthermore, traceability of results is important to ensure estimator acceptance.

7.2. Features

The application SEROCOST is an application with a graphical, windows-based user interface. It guides the estimator through a sequence of dialogs that are in accordance with the estimation process presented in Chapter 4 (s. Figure 4.1-2). Manifold possibilities to customize the estimation to the estimator's specific needs render the tool flexible yet maintaining ease of use.

The first selection window after starting a calculation from the main menu presents the user with lists of typical service robot skills as well as hardware and software component types. Here, the user can select from the listed items by simply mouse-clicking on the respective checkboxes (s. Figure 7.2-1). If desired the user can also request default recommendations for least requirements, enhancements and substitutes on each category which depend on the choices he has made so far.

The following dialog allows the parameterization of the previously selected hardware and software components (Figure 7.2-2). Each component is already parameterized with default values from the database but the user can calibrate the estimation to his concepts by changing hardware parameters and adding or removing software packages from the selection.

The third and final window of the calculation process displays the essential cost estimates in a clearly arranged fashion (Figure 7.2-3). The estimates presented here include all cost constituents – cost for hardware material, installation and administration, software development and installation and system designing costs – and the overall cost estimates for a prototype, for direct unit costs as well as product unit costs. Furthermore the basic parameters for wage rates, number of units to be built and productivity rates are displayed, all of which can be changed by the user in order to modify the estimation calculation. This window offers a multitude of further customization possibilities of the estimation, e.g. excluding any of the cost constituents or overriding singular estimation parts with own estimations.

At each step the results and choices can be saved to an Excel file and loaded from such an Excel file. These files are designed in such a way that they can be interpreted by users without the Software tool SEROCOST.

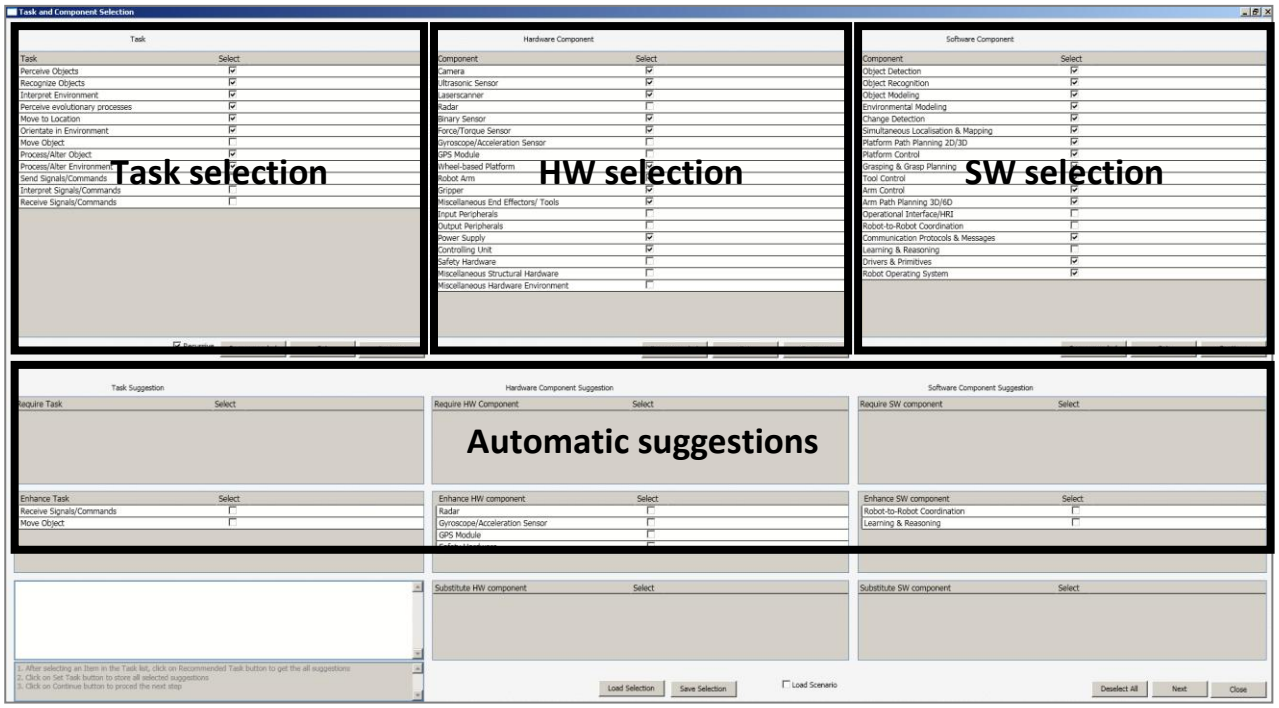

Figure 7.2-1 Step 1: Selecting tasks and component types

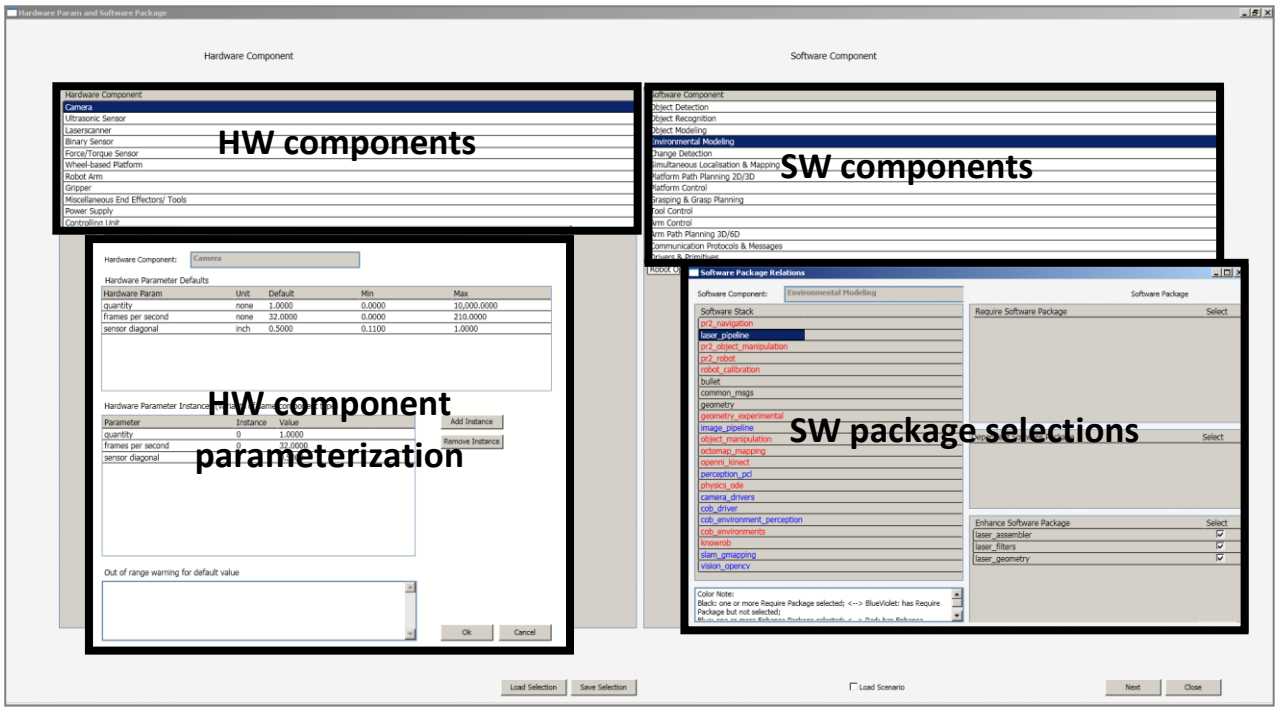

Figure 7.2-2 Step 2 Parameterization of components

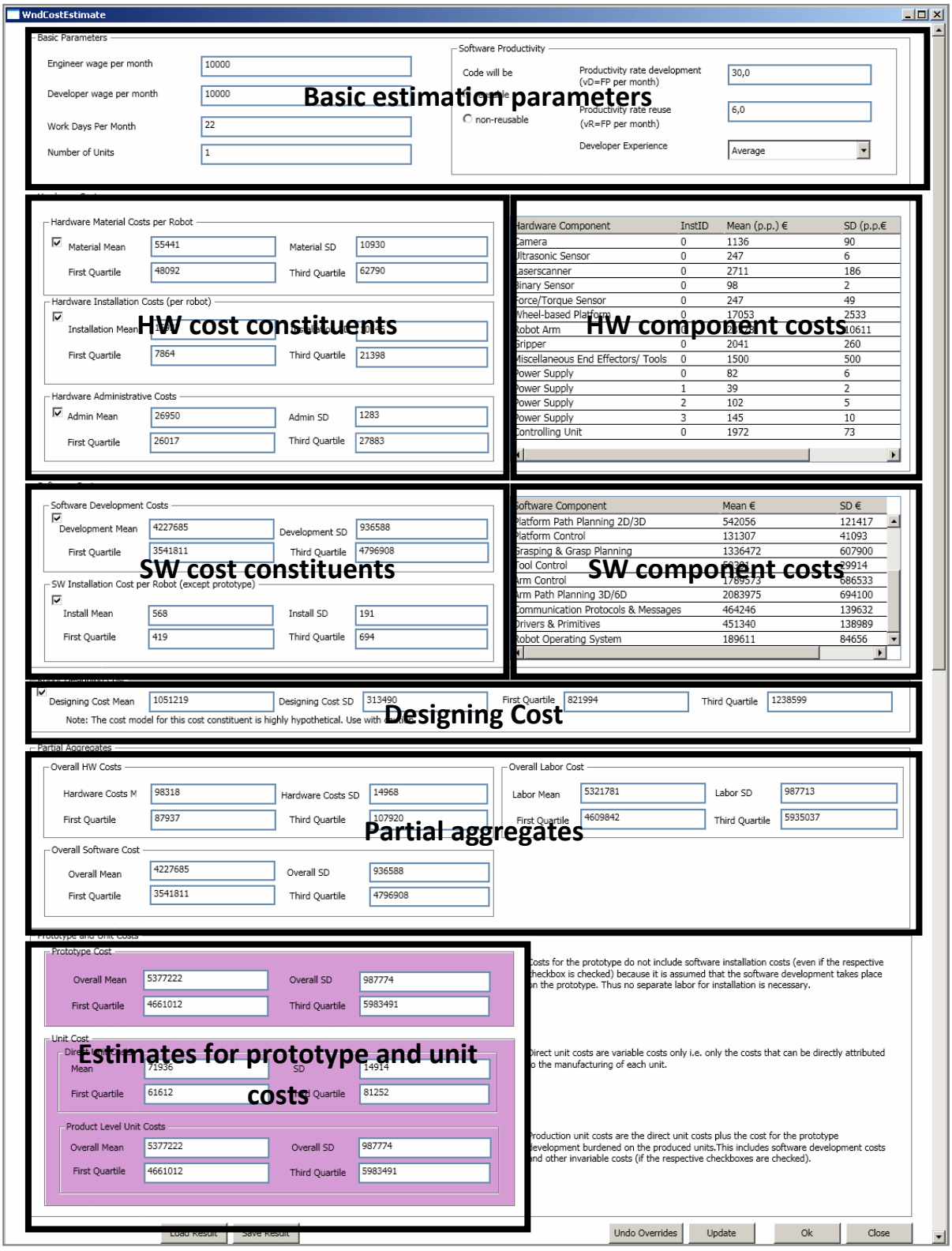

Figure 7.2-3 Step 3 Overview of calculated estimates

7.3. Basic Architecture

The SEROCOST application consists of three basic elements. The main application which was implemented in C# and XAML provides the graphical user interface as previously described and the control components of the estimation process.

A PostgreSQL database holds the data on skills, hardware and software component types, dependencies between them and further data like default values for hardware parameters and software packages. The database can be deployed locally or on a remote server.

The third component is the R statistics software which is a separate environment for statistical computing developed as an open source GNU project by Bell Laboratories (RDCT 2006). Interfacing with R permits the reliable on-the-fly calculation of distribution functions and statistical measures used in the presented approach.

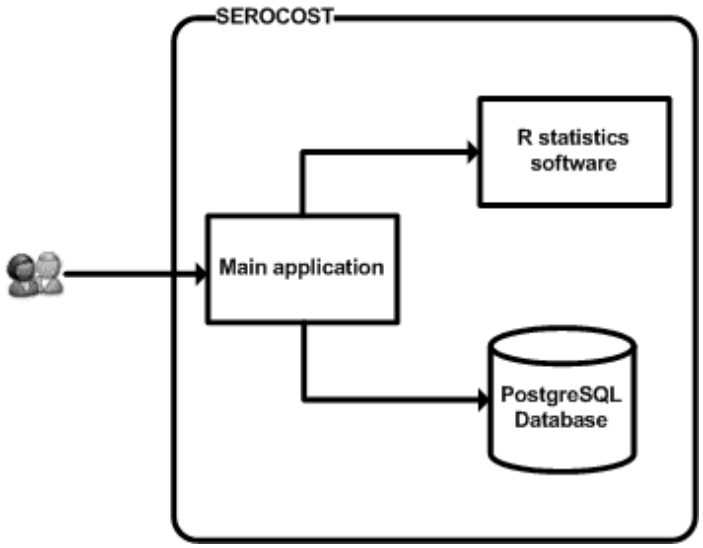

Figure 7.3-1 Basic architecture of the software tool SEROCOST

The described combination was chosen in order to reduce development time and apply state of the art techniques for statistical calculation and knowledge management.

8. Experimental Validation

Ideally, an estimate for a service robot would be conducted in an early product phase and then compared with the actual costs once the robot prototype is finished. Due to the typical development spanning years this validation method appears hardly practicable. Instead, two alternatives were applied for verifying the approach's usefulness.

8.1. Validation Methods

The first validation method consists in comparing the estimations of the presented approach with estimations of the EFFIROB study.[40] The EFFIROB scenarios are an appropriate reference for validation comparisons as they provide data on software and hardware cost estimates generated by service robot experts. Furthermore, as the service robot designs described in EFFIROB represent concept sketches and not actually built robots the cost estimates map well with the early phase estimation approach in this work. The comparison of estimates was conducted quantitatively and qualitatively: Absolute and relative differences between original and new estimate were calculated and interpreted; secondly, the estimation results were presented to and assessed by the individual expert who had calculated the original estimate. Because the presented approach allows the calculation of several cost aggregation levels, different aggregation forms were compared with the original estimate and the expert's opinion on their validity elicited.

The second validation method contrasts the estimated and actual costs of a completed service robot prototype, the autonomous lawn mower "Raser" with an estimate created using the presented approach and project background data. The Raser project was chosen because its specific objective was the development of a service robot prototype, the cost data available from this project are sufficiently detailed, and the robot concept fits well into the definition of service robot applied in the presented approach. The new estimate was compared with the original a priori cost estimate and the actual costs incurred. Additionally, the results were

[40]Each scenario details two variants of a concept. For the verification process, only variant "A" was used.

presented to one of the developers involved in the Raser project elicitating his assessment of the new estimate.

Both validation methods apply the deviation of the estimates based on the presented approach from original estimates or actual costs, respectively, as a performance indicator. In order to assess the significance of the discrepancies they are related to cost estimation error ranges that have been observed in the past. As indicated in Table 4.4-4 errors between -75% and +300% can be expected for the initial concept phase, -33% and +50%, respectively, are still common once all requirements have been established. Another study on forecasting errors reports that the average estimation inaccuracy compared to actual costs for rail construction projects is -51.4% and that 40% of the project deviated more than ±60% from the initial estimate, 84% revealed inaccuracies over ±20% (Flyvberg 2008, pp. 6–7). Thus, the discrepancies found in the validation process of the presented work are assessed in relation to this magnitude of plausibly expectable errors.

In the following explanations, the estimates based on the presented approach are referred to as *new estimates* whereas the values to which they are compared to are referenced as *original estimates* in order to distinguish terminologically between the different estimations.

8.2. Comparison with EFFIROB Estimates

In the EFFIROB study, the lifecycle costs for eleven conceivable service robots are analyzed. In order to derive lifecycle costs the costs for each service robot are extrapolated from design sketches based on the robot functionality required in each respective scenario. While the deduction of robot components from required skills is similar to the presented approach the costs for a prototype are not explicitly addressed in EFFIROB. For this reason some of the data had to be preprocessed in order to render a comparison possible; the detailed method is described in Chapter 8.2.1. To achieve a comprehensive view on the estimation and the several estimated cost constituents, several aggregation levels were calculated and contrasted with the analogon of the original scenario estimate.

8.2.1. Assessment Method

The validation by comparison with EFFIROB estimates was conducted following a procedure with several steps as depicted in Figure 8.2-1.

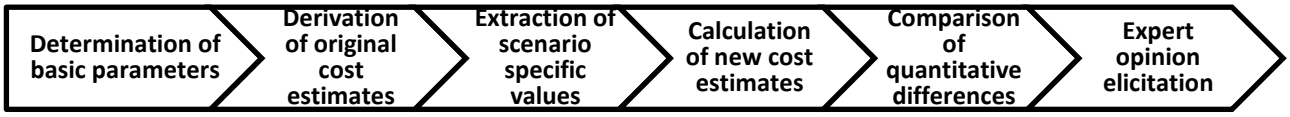

Figure 8.2-1 Validation process based on EFFIROB comparison

In order to render new and original estimates comparable they must be based on the same basic parameters. The parameters that are used throughout all EFFIROB scenarios and also required for the new estimation approach are the developers' wage rate, the number of robot units to be produced and the experience level of the software development team.

The wage rate is explicitly numeralized as 10,000 € per month including ancillary labor costs (Hägele 2010, p. 43); this wage rate is thus also applied in the new approach for both engineers and software developers. As no specification on workdays per month is given in the EFFIROB study, this value was set to 22 for the new estimations.

The number of units is not used as a parameter to determine robot prices in EFFIROB but is accounted for in the respective profitability analyses. Each scenario explicitly indicates the market potential of robot units (n_{market}) under consideration of the robot's estimated lifespan (ibid., p. 57). This value was used for the derivation of product unit costs i.e. burdening indirect costs like software development cost on the direct unit cost.

The experience level of the software development team is not stated explicitly in the EFFIROB study but can be extracted from the software cost model. The software costs are calculated using average productivity (Jones 2007, p. 144) which is the rate the presented approach applies for average team experience level. This value was adopted for the new estimations in order to ensure comparability with the original estimates (Hägele 2010, p. 43).

In the next step different cost aggregates of the original estimates were derived. The EFFIROB study explicitly indicates acquisition cost ($AC_{EFFIROB}$) for a service robot and software development costs ($C_{EFFIROB}^{SW}$) separately. The latter can be directly compared with the new estimates for software development. However, acquisition costs are regarded from the user's and not the manufacturer's perspective, they are indicated as the cost of hardware components including installation plus a 30% price markup. Deducting the markup yields the EFFIROB direct unit cost ($DUC_{EFFIROB}$) estimate which is also identical to the hardware cost estimate (labor and material) as software costs are not burdened on the units in the EFFIROB study.

To permit further comparisons, prototype cost ($c_{EFFIROB}^{Prototype}$) and product unit cost ($PUC_{EFFIROB}$) estimates were also derived from the given estimates. The EFFIROB prototype cost estimate is calculated as hardware costs (s. above) for one robot plus software development costs; product unit costs are calculated as software development costs plus hardware costs for a number of robot units indicated by the market potential plus one unit for the prototype, divided by the number of marketed robot units i.e. the costs for the prototype are burdened on the marketed units.

$$DUC_{EFFIROB} = \frac{AC_{EFFIROB}}{1.3} \tag{8.1}$$

$$c_{EFFIROB}^{Prototype} = C_{EFFIROB}^{SW} + DUC_{EFFIROB} \tag{8.2}$$

$$PUC_{EFFIROB} = \frac{DUC_{EFFIROB}(n_{market} + 1) + C_{EFFIROB}^{SW}}{n_{market}} \tag{8.3}$$

The next step concerns the extraction of scenario specific parameters for the calculation of new estimates. Required robot skills, hardware and software components and the pertinent parameter values were determined by analyzing the robot functions and component specifications described in the respective EFFIROB scenarios. The robot task descriptions were mapped to the generic robot skills and the component type selection suggested by the software tool adapted according to the design sketches. Parameters for hardware components were adopted where available otherwise plausible values were assumed. Because the specification of software components in the scenario descriptions is rather generic, the section of

required software components was similarly based on plausible assumptions fitting the given explanations. The details of each selection can be found in Appendix 10.5.

After determining the robot's cost relevant parameters, the different cost constituents and cost aggregates were calculated as explained in Chapter 6. The cost constituents calculated are

- Hardware material costs per robot unit
- Hardware installation costs per robot unit
- Hardware administrative costs
- Software development costs
- Software installation costs per robot unit except prototype
- System designing costs

The various cost aggregations calculated are

- Prototype costs including all cost constituents
- Product unit costs including all cost constituents
- Prototype costs including all cost constituents except system designing cost
- Product unit costs including all cost constituents except system designing cost
- Prototype costs including all cost constituents except system designing cost and hardware administrative costs
- Product unit costs including all cost constituents except system designing cost and hardware administrative costs
- Direct unit costs

Expected value, standard deviation as well as first and third quartile were computed for each cost estimate. The reason for calculating several aggregates is to find out which cost estimate meets the highest acceptance by expert opinion.

After the calculation of the new cost estimates the absolute and relative differences[41] to the corresponding original estimate were determined and interpreted. Large differences are an indicator for potential systemic estimation bias

[41]Relative differences were calculated with the original estimate representing the 100% reference.

whereas proximity to the original values is regarded as an indication of valid estimates; thresholds for difference classification are adopted from Table 4.4-4.

In a final step the documented calculation and the results were presented to the author of the original EFFIROB scenario description and his judgment elicited using a prepared questionnaire. The questionnaire surveys the opinion on the estimation results of the individual cost constituents, the various aggregation forms and the applicability of the approach; the questionnaire used can be found in Appendix 10.6. Based on the expert's opinion, the validity of the new estimate and the general usefulness of the approach were assessed.

8.2.1.1. Questionnaire Setup

The questionnaire contains ten questions, nine of which (questions Q1 to Q9) permit response on a Likert scale, one (question Q10) is an open question. Furthermore, each question also provides the possibility for further comments in a separate comment field. Because the available data in the Raser verification case differs from the EFFIROB scenarios, a modified version of the questionnaire was used for the Raser scenario.

Question Q1 addresses the estimate's relative quality for hardware (Q1.1) and software (Q1.2) costs by asking how realistic the most appropriate aggregate is assessed in comparison to the original cost estimate. In the Raser case, the main distinction is between material and labor, thus these are used as items to be assessed instead of hardware and software cost. Possible responses are 'much more realistic', 'more realistic', 'equally realistic', 'less realistic', 'much less realistic', and 'I don't know'.

Questions Q2 to Q4 serve to assess the estimates for the various cost constituents independently from the original estimates by direct elicitation of the expert's opinion e.g. "How do you assess the hardware cost estimation?" (question Q2). Question Q2 and Q3 address the constituents of each of these main cost blocks (Q2.1 to Q.2.2 and Q.3.1 to Q3.3, respectively), and Question Q4 surveys the opinion on system designing costs (Q4.1). Because in the Raser case, software installation costs and hardware administrative costs are not computed as a separate cost constituent, the modified questionnaire does not survey these items. Each item

offers five response possibilities: 'very realistic', 'reasonably realistic', 'somewhat unrealistic', 'very unrealistic', and 'I don't know'.

Question Q5 surveys the usefulness of indicating standard deviation and interquartile range. For each of these (Q5.1 and Q5.2), possible answers are 'very helpful', 'helpful to some degree', 'neither helpful nor confusing', 'confusing to some degree', 'very confusing' and 'I don't know'.

Question Q6 is aimed at finding out which cost aggregate the expert considers the most meaningful for cost estimate of prototype (Q6.1) and product level (Q6.2) units. For each of these two the possible responses are 'including all cost constituents', 'excluding system designing cost', excluding system designing costs, hardware administrative costs', 'direct unit costs', 'none of the above', and 'I don't know'. For the Raser case, only prototype costs are considered and thus the item product level units is not surveyed in the modified questionnaire.

Question Q7 to Q9 address the support value of the presented approach in combination with other estimation methods. Question Q7 surveys in how far the expert considers the additional support by the tool SEROCOST helpful (Q7.1), possible responses are 'very helpful', 'helpful to some degree', 'neither helpful nor confusing', 'confusing to some degree', 'very confusing', and 'I don't know'. Question Q8 surveys how the expert would combine original and SEROCOST estimate for hardware (Q8.1) and software (Q8.2) cost each. Possible responses are 'replace original values with SEROCOST values', 'find a compromise between both values', 'keep old values', and 'I don't know'. Question Q9 examines the expert's acceptance of tool and approach by asking if he or she would use it in the future if an early phase cost estimate for a service robot were required (Q9.1). Possible responses are 'definitely', 'probably', 'maybe', 'probably not', 'definitely not', and 'I don't know'.

The final question (question Q10) is an open question where opinions on strengths and weaknesses not covered in the previous questions can be stated.

8.2.1.2. Questionnaire Evaluation

The responses from the expert interviews were evaluated in various ways in order to assess if the objective of facilitating cost estimation for service robot prototypes has been met.

Positive, neutral (where applicable) and negative assessments were accumulated and compared for the questions with ordinal response scales Q1 to Q5 and Q7 to Q9. This evaluation yields an overview over the tendency of the assessments. Question Q6 has a nominal response scales and question Q10 is an open question; therefore these were exempt from this evaluation method. Because the questionnaire had been modified for the Raser estimation scenario the respective responses were evaluated separately.

Furthermore, ordinal responses were assigned integer values in order to calculate the median response and average values for each question, ranging from -2 for the most negative possible answer to +2 for the most positive possible answer, 0 for neutral responses where applicable. As question Q8 only offers the choice between the three response types of preferring the original estimate, compromising between new and original estimate, and preferring the new estimate the assigned values are -1, 0, and +1 , respectively. For each question the median of the responses was calculated to indicate the dominant judgment; additionally, the average value is also calculated to give a more detailed impression of the assessment tendency.

The responses to question Q6 indicate which aggregate the experts preferred. These responses are nominal i.e. no aggregate is considered better or worse than another one. The mentions for each aggregate are counted; the prevalence rates indicate the subjective suitability of the aggregates for cost estimation.

Comments and answers to question Q10 were segregated into the categories 'related to a specific cost constituent', related to the approach as a whole', and 'related to the software tool SEROCOST' and marked as either positive, negative and neutral statements; similar remarks were aggregated and the number of mentions counted. This qualitative evaluation adds additional information on the expert assessment of the approach.

8.2.2. EFFIROB Scenarios

Five scenarios were picked from the EFFIROB study as a heterogeneous collection of use cases representing conceivable service robots with largely varying purposes and designs. The scenarios selected are "Provisioning of Care Utensils", "Ground-crop

Harvesting", "Floor Cleaning", "Container Transport in Hospitals" and "Assistance with Interior Finishing Works".

8.2.2.1. Provisioning of Care Utensils

The scenario for care utensil provisioning describes a concept for a service robot that can replace conventional medical carts in homes and reduce care staff workload by supplying staff members with required care utensils. It is able to restock autonomously from a central store and to travel to specified rooms where care utensils are needed while keeping track of the consumption of the delivered items. It features a mobile platform with room for item storage and dispensing and a robot arm with a gripper so it fits well into the notion of service robots as conceived in the presented work.

In the EFFIROB study, the direct cost of this robot are indicated as 117,200 € and the software development cost as 6,886,000 €. The derived prototype costs are calculated to amount to 7,003,200 €, product unit cost lie at 409,000 €.

The new estimates calculated for the main cost constituents are displayed in Table 8.2-1. The system designing costs depend on the values of the other cost constituents; the indicated value is calculated for the aggregation form of including all cost constituents.

Figure 8.2-2 Design sketch for care utensil service robot

The removal mechanism for item dispensing, the linear axis on which the robot arm is mounted and the WLAN router could not be estimated by the component type specific models because they did not fit any of the component types other than miscellaneous tools or miscellaneous structural hardware. For these items, the estimates from the original are adopted as they were.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Material (per unit)	121,389.58 €	11,183.26 €	113,870.33 €	128,908.84 €
Hardware installation (per unit)	21,900.00 €	13,950.00 €	10,813.14 €	29,421.70 €
Administration	36,575.00 €	1,741.67 €	35,308.97 €	37,841.03 €
Software development	5,814,611.29 €	972,242.91 €	5,117,914.34 €	6,428,873.08 €
Software installation (per unit, not for prototype)	568.18 €	190.91 €	418.62 €	693.92 €
System designing	1,456,657.64	378,552.25 €	1,182,376.29	1,688,607.26

Table 8.2-1 New estimates for cost constituents of the care utensil scenario

The different cost constituents were subsequently combined into various cost forms according to the formulas described in Chapter 6. These cost aggregates are indicated in Table 8.2-2. The software costs are invariant for the aggregates and thus already listed in Table 8.2-1.

As software development cost and direct unit cost are invariant to the cost aggregate form their comparison is straightforward. The new estimate of the direct unit cost lies 22.75% above the direct unit cost derived in the EFFIROB study, software development cost lie 15.56% below the original value. These differences are well within the limits of estimation errors for an early phase estimate and thus these estimates can be considered close to each other.

Prototype cost, product unit cost and hardware cost need to be regarded separately for each cost aggregation. In the most comprehensive aggregate, the new prototype cost estimate lies 6.4% higher than the original one and thus can be considered very close. Product unit costs are estimated to be 11.08% larger than in the original which is also a small difference. Hardware costs however are estimated to be 53.47% higher than in the EFFIROB study. This is a difference that is still plausible as differences in early design phases can assume more than 100% but it also indicates that the different estimation methods lead to significantly differing values. A

possible explanation for this discrepancy is the inclusion of administrative cost for hardware components in the presented approach, a cost constituent which was not considered in the EFFIROB study.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Aggregate variant: All cost constituents				
Prototype cost	7,451,133.51 €	1,043,494.80 €	6,709,661.54 €	8,115,678.09 €
Product unit	454,321.66 €	47,294.78 €	420,971.39 €	484,606.70 €
Hardware cost	179,864.58 €	17,963.88 €	167,191.01 €	191,121.83 €
Aggregate variant: Excluding system designing cost				
Prototype cost	5,994,475.87 €	972,408.85 €	5,297,505.71 €	6,608,545.06 €
Product unit	393,627.59 €	44,586.69 €	362,000.32 €	422,018.08 €
Hardware cost	179,864.58 €	17,963.88 €	167,191.01 €	191,121.83 €
Aggregate variant: Excluding system designing cost, hardware admin. cost				
Prototype cost	5,957,900.87 €	972,407.29 €	5,259,576.98 €	6,570,616.47 €
Product unit	392,103.64 €	44,586.64 €	360,450.08 €	420,457.86 €
Hardware cost	143,289.58 €	17,879.25 €	130,752.43 €	153,936.05 €
Aggregate invariant				
Direct unit cost	143,857.77 €	17,880.27 €	131,309.44 €	154,547.79 €

Table 8.2-2 Cost aggregate estimations for care utensil scenario

The cost aggregate disregarding system designing cost presents a similar difference profile; the new prototype cost estimate lies 14.4% below the original, product unit costs are very close to the original with a difference of only 3.76%. The hardware cost difference remains equal to the previous aggregate. Excluding the hardware administrative cost brings the hardware costs much closer to the original, the difference shrinking to 22.26%; prototype cost and product unit cost estimates stay close to the original estimates with differences of -14.93% and -4.13% respectively. Overall, the least comprehensive aggregate numerically leaves the impression to be the most compatible estimate to the original EFFIROB values.

The expert's general impression of the estimated value was positive. Although she pointed out that she would have chosen a slightly different hardware selection hardware costs were judged to be more realistic and software costs to be equally

realistic in comparison to the original values.[42] However, hardware administrative costs were considered as tending too high and thus somewhat unrealistic; the system designing costs were regarded as very unrealistic. The usefulness of the approach was assessed as high, the expert stated that she would probably replace original estimates with the new estimates excluding the designing costs and use the estimation tool in the future estimation situations.

8.2.2.2. Ground-crop Harvesting

The EFFIROB scenario for ground-crop harvesting describes a robot that is able to harvest and grade ground-crops e.g. lettuce. The primary components are six robot arms with three-finger grippers and appropriate sensors mounted on a large horizontal linear axis. It is designed to be dragged behind a tractor so the robot is mobile in the sense that is used while being moved but is not able to do so on its own accord. Thus, this service robot is not based on an autonomous mobile platform typically found in many service robot concepts which makes it an appropriate case for analyzing if the presented approach is amenable to more unusual designs, too.

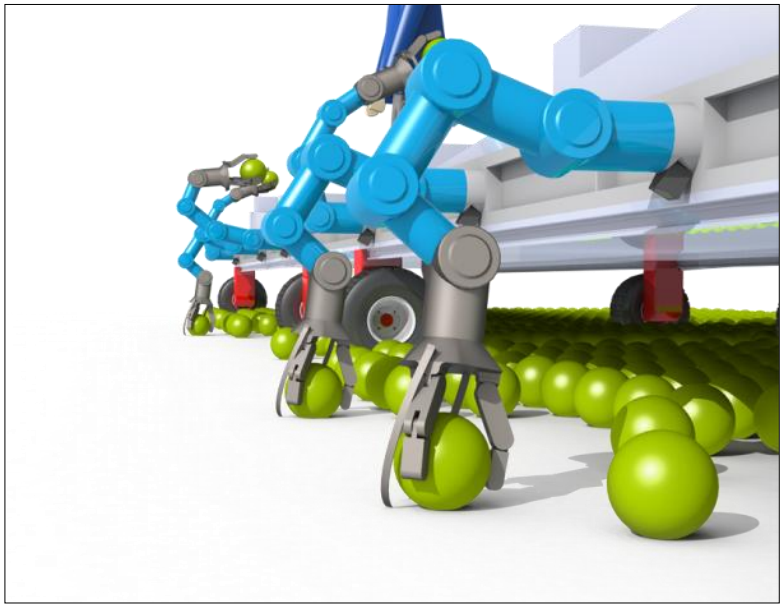

Figure 8.2-3 Design sketch of ground-crop service robot

[42]The estimate was computed with one controlling unit, the expert indicated that two might be more realistic which would have increased materials costs by 1,972 €.

Direct costs in the original estimate amount to 541,615 € and software development cost to 903,100 €. Prototype costs are estimated to be 4,871,392 € and product unit cost 767,155 €.

The new estimates of main cost constituents for the ground-crop harvesting service robot are displayed in Table 8.2-3; the different aggregates of these are listed in Table 8.2-4.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Material (per unit)	564,761.94 €	85,362.59 €	507,366.92 €	622,156.95 €
Hardware installation (per unit)	33,845.45 €	21,559.09 €	16,711.22 €	45,469.90 €
Administration	23,100.00 €	1,100.00 €	22,300.40 €	23,899.60 €
Software development	3,245,956.24 €	921,900.92 €	2,555,077.78 €	3,770,080.83 €
Software installation (per unit, not for prototype)	568.18 €	190.91 €	418.62 €	693.92 €
System designing	939,842.26 €	296,909.74 €	721,305.47 €	1,114,313.03 €

Table 8.2-3 New estimates for cost constituents of ground-crop harvest scenario

The cost estimates for the linear axis and the three-hand gripper were adopted from the original because they did not fit into the component type cost models. The gripper could not be estimated using the regressive cost model because the model was derived from data for jaw-like grippers only.

Comparing the estimates showed that the direct unit costs are close to each other with the new estimate being 10.63% higher; the original estimate thus lies well within the interquartile range of the new estimate. The software development costs, however, differ significantly: The new estimate exceeds the original by 259.42% which is already very close to the upper threshold of 300% for typical estimation errors in early design stages.

Further analysis revealed that a large proportion of the new estimate for software development stems from arm and gripper planning and control. This large discrepancy was discussed with the creator of the original estimate. The author of the original scenario found that the new software cost estimate was too high because in his opinion arm and gripper control and planning are not as complex and thus cost-intensive as the new estimate suggests. He also suggested that the

increase of costs in the estimate could stem from the large expansion of the ROS software that had taken place in the meantime.[43]

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Aggregate variant: All cost constituents				
Prototype cost	4,807,505.90 €	972,527.37 €	4,092,939.99 €	5,371,484.86 €
Product unit	759,425.77 €	96,536.38 €	694,195.15 €	823,853.29 €
Hardware cost	621,707.39 €	88,049.85 €	562,412.80 €	680,798.88 €
Aggregate variant: Excluding system designing cost				
Prototype cost	3,867,663.63 €	926,176.41 €	3,206,353.93 €	4,427,735.59 €
Product unit	728,097.69 €	96,028.76 €	664,250.43 €	793,191.72 €
Hardware cost	621,707.39 €	88,049.85 €	562,412.80 €	680,798.88 €
Aggregate variant: Excluding system designing cost, hardware admin. cost				
Prototype cost	3,844,563.63 €	926,095.50 €	3,151,456.17 €	4,370,516.31 €
Product unit	727,327.69 €	96,027.71 €	662,439.40 €	791,375.79 €
Hardware cost	598,607.39 €	88,042.98 €	539,350.67 €	657,713.38 €
Aggregate invariant				
Direct unit cost	599,175.57 €	88,043.19 €	539,876.17 €	658,228.09 €

Table 8.2-4 Cost aggregate estimations for ground-crop harvest scenario

By heavily reducing the number of selected packages for the concerned software component types the software development cost estimate could be shrunk from 3,245,956 € to 1,608,411 €, this reduction also propagates to system designing costs and prototype costs. The revised calculations were also presented to the same expert: The new software development estimate was judged to be more realistic, the system designing cost at 541,919 € as still too high; the expert stated that 300,000 € could be a plausible value.

Due to the initially large difference the initial prototype cost estimates differ largely, too; the initial new estimate lies 232.76% higher at the highest aggregation level, 167.71% if system designing costs are excluded and 166.11% higher if hardware administrative costs are also disregarded. Even though the discrepancy for software

[43]From May 2009 to November 2010 alone, the number of available packages has more than quadrupled, s. (Boren 2011, p. 20).

development is significant the product unit costs at the three aggregation levels differ 28.77%, 23.45% and 23.32% respectively which is a moderate deviation. Material costs also only differ by 14.79%.

Even though the initial software cost estimate according to the new approach was assessed as less realistic than the original estimate the overall judgment was positive. Hardware costs were considered as much more realistic in general: material costs as very realistic and installation costs as reasonably realistic but administrative costs were considered somewhat unrealistic. The latter were found as tending to be too high and the interquartile range as too narrow. Software installation costs were regarded as very realistic but system designing costs as too high and thus somewhat unrealistic.

The availability of standard deviation and interquartile range was pointed out to be very helpful especially when contrasted with the point estimates in the EFFIROB study. Furthermore, the expert stressed that even though the approach and the accompanying software tool cannot replace expert "gut feeling" they were considered a valuable support for early cost estimations.

8.2.2.3. Floor Cleaning

The floor cleaning service robot presented in the EFFIROB study is designed for autonomous cleaning service within office buildings. To this end it features a variety of sensors (laser scanners, ultrasonic arrays, cameras) and also a robot arm which allows the unlocking and opening of doors. The robot is meant to provide its service in a typical office environment so it must be able to robustly navigate in spite of possibly repositioned objects like tables or waste bins. The concept also includes a separable floor cleaning unit which is operated by the robot arm.

The original estimate for direct unit costs and hardware costs are indicated as 71,900 €, software development costs amount to 14,436,400 €. These numbers yield prototype costs of 14,498,300 € and product unit costs of 118,972 €. As the design was found unprofitable at these costs the EFFIROB study states that zero units could be marketed; however, a theoretical market saturation potential of 308 robots is pointed out. This number was used for the derivation of product unit costs.

Figure 8.2-4 Design sketch of floor-cleaning service robot

Most of the components from the given design are amenable to the presented cost models; the only exception is the cleaning unit which was classified as a miscellaneous tool. For this tool, the original cost estimate was adopted. The resulting estimates for the main cost blocks are displayed in Table 8.2-5. The different aggregations are presented in Table 8.2-6.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Material (per unit)	55,441.02 €	10,930.45 €	48,091.74 €	62,790.30 €
Hardware installation (per unit)	15,927.27 €	10,145.45 €	7,864.11 €	21,397.60 €
Administration	26,950.00 €	1,283.33 €	26,017.13 €	27,882.87 €
Software development	4,227,684.71	936,587.63 €	3,541,810.77	4,796,908.35
Software installation (per unit, not for prototype)	568.18 €	190.91 €	418.62 €	693.92 €
System designing	1,051,267.25	313,490.50 €	821,994.46 €	1,238,599.36

Table 8.2-5 New estimates for cost constituents of floor cleaning scenario

The new direct unit cost estimate is almost identical to the original with an excess of only 0.05%. Similarly, hardware costs excluding hardware administrative costs only lie 0.74% below; including the administrative costs (first and second aggregate) yields a surplus of 36.74%, an value still well within the expectable margins.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Aggregate variant: All cost constituents				
Prototype cost	5,377,221.73 €	987,773.62 €	4,661,011.77 €	5,983,490.86 €
Product unit cost	89,394.99 €	15,302.64 €	78,840.34 €	99,208.78 €
Hardware cost	71,455.51 €	14,913.25 €	87,936.89 €	107,919.62 €
Aggregate variant: Excluding system designing cost				
Prototype cost	4,326,003.00 €	936,707.23 €	3,639,869.41 €	4,895,042.64 €
Product unit cost	85,981.94 €	15,268.75 €	75,437.65 €	95,735.61 €
Hardware cost	71,455.51 €	14,913.25 €	45,505.04 €	60,197.54 €
Aggregate variant: Excluding system designing cost, hardware admin. cost				
Prototype cost	4,299,053.00 €	936,706.35 €	3,611,423.68 €	4,866,596.65 €
Product unit cost	85,894.44 €	15,268.75 €	75,346.68 €	95,642.56 €
Hardware cost	71,368.29 €	14,913.25 €	61,049.87 €	80,655.38 €
Aggregate invariant				
Direct unit cost	71,936.47 €	14,914.47 €	61,611.66 €	81,252.42 €

Table 8.2-6 Cost aggregate estimations for floor cleaning scenario

In contrast, the new estimate for software development costs grossly undercuts the original by 70.69% which is close to the lower error margin of -75%. This large difference could possibly be caused by the fact that the software cost estimation method applied in the EFFIROB study does not discern installed and developed packages. This could have the effect that a large proportion of software components had been estimated to be developed whereas the new approach excluded these from development cost. In order to find out the discrepancy was explicitly discussed with the creator of the original estimate. The expert agreed that the original cost estimate for software was probably too high and that the differentiation between components to be developed and those only to be installed appears more realistic.

Due to the large discrepancy in software costs, the new cost estimates lie far below the original: -62.91% when all cost constituents are considered, -70.16% excluding system designing costs and -70.35% further excluding hardware administrative costs.

The overall judgment was that the hardware cost estimate was equally realistic and the software cost estimate more realistic than the original values. Hardware administrative costs were not assessed because the expert did not consider the cost

definition as sufficiently transparent. Software installation costs were regarded as very unrealistic unless installation would be automatic; the expert's opinion was that installation would take much longer and thus cost more. System designing costs were considered too high and thus somewhat unrealistic. The expert indicated that he could not assess the additional information given by interquartile range and standard deviation because he considered himself too inexperienced with statistical measures. He regarded the tool as helpful and pointed out that if the time effort spent to calculate the estimates amounted to circa an hour he would "definitely" apply the presented approach in order to arrive at an early order-of-magnitude cost estimate.

8.2.2.4. Container Transport in Hospitals

The container transport scenario presents the idea of a service robot that is able to autonomously transport containers or beds in a hospital environment. Its central feature is a mobile platform of a particularly flat design which permits it to position itself below the transport object and lift it with is built-in lifter from the ground.

The originally estimated direct unit costs and hardware costs for the container transporting robot are 43,800 € and software development costs add up to 3,336,900 €. Accordingly, prototype costs amount to 3,380,700 € and product unit cost 50,843 €.

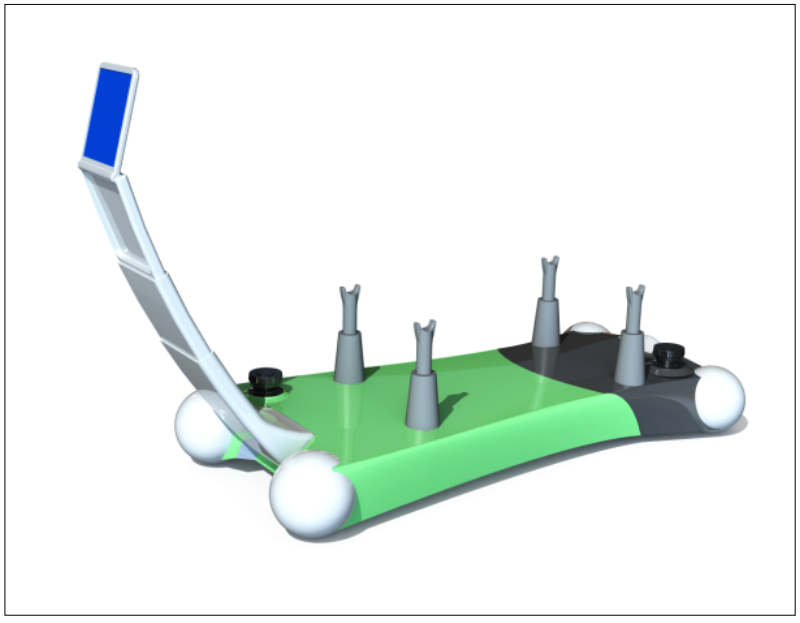

Figure 8.2-5 Design sketch of container transporting service robot

The new estimates for the main cost constituents are depicted in Table 8.2-7, the aggregations in Table 8.2-8. Components not estimated separately are the miscellaneous structural hardware components WLAN router and RFID reader as well as the miscellaneous end effector component i.e. the lifting mechanism.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Material (per unit)	52,539.38 €	4,318.18 €	49,635.98 €	55,442.78 €
Hardware installation (per unit)	14,600.00 €	9,300.00 €	7,208.76 €	19,614.47 €
Administration	23,100.00 €	1,100.00 €	22,300.40 €	23,899.60 €
Software Development	1,881,798.45	302,103.78 €	1,668,786.23	2,078,207.95
Software installation (per unit, not for prototype)	568.18 €	190.91 €	418.62 €	693.92 €
System designing	479,205.19 €	121,801.23 €	391,604.63 €	555,483.92 €

Table 8.2-7 New estimates for cost constituents for hospital container transport scenario

The results of the comparison leave an unclear picture: While the new estimate for software development is 43.61% below the original the new hardware cost estimates lie significantly above them with 106.03% surplus for the two more comprehensive aggregates and still 53.29% excluding hardware administrative costs. Further analysis revealed that the cost estimation for the platform component were the principal source of this difference: While the EFFIROB study states the cost for the platform to be 30,000 € including installation costs, the model estimates it to be circa 40,000 € excluding installation costs at a parameterization of 300 kg weight, 275 kg payload and 1.8 m^2 footprint size. As this component accounts for almost three quarters of the original cost estimate the deviation results in a significant relative difference. As these values are close to the upper limit of the model parameters for platform weight and payload (350 kg and 300 kg, respectively) border effects could also have influenced this specific cost estimate. Furthermore, the inclusion of components not considered in the original estimate e.g. power supply possibly biases the new estimates towards higher values. Because the hardware component costs dominate the direct unit costs, these also exceed the original by 54.58%.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Aggregate variant: All cost constituents				
Prototype cost	2,451,243.02 €	325,896.58 €	2,222,209.56 €	2,663,094.93 €
Product unit cost	72,814.32 €	10,299.13 €	65,317.43 €	78,998.68 €
Hardware cost	67,187.41 €	10,312.45 €	82,670.22 €	96,496.72 €
Aggregate variant: Excluding system designing cost				
Prototype cost	1,972,037.83 €	302,279.74 €	1,758,484.21 €	2,167,945.05 €
Product unit cost	71,815.98 €	10,296.01 €	64,304.54 €	77,961.68 €
Hardware cost	67,187.41 €	10,312.45 €	82,670.22 €	96,496.72 €
Aggregate variant: Excluding system designing cost, hardware admin. cost				
Prototype cost	1,948,937.83 €	302,277.74 €	173,573.86 €	2,145,034.72 €
Product unit cost	71,767.85 €	10,296.01 €	64,256.25 €	77,912.45 €
Hardware cost	67,139.38 €	10,253.62 €	59,722.92 €	72,860.32 €
Aggregate invariant				
Direct unit cost	67,707.56 €	10,255.39 €	60,271.50 €	73,496.70 €

Table 8.2-8 Cost aggregate estimations for hospital container transport scenario

However, the new prototype cost estimate undercuts the original by 27.49% at the most comprehensive aggregation level and by 42.35% excluding system designing and hardware administrative costs. This is due to the prevalence of the software cost. In contrast, the product unit costs at the three aggregation levels are 41.16% to 43.21% higher than in the EFFIROB estimate because at the indicated level for market saturation with 480 units the burdened software development costs plays a less significant role than the hardware costs.

The expert's opinion on the results was ambiguous: Whereas most component cost estimates were considered appropriate, the cost estimate for the mobile platform and thus the total hardware cost estimate were assessed as too high and thus less realistic than the original. A possible explanation suggested by the expert was that most platforms in the considered price range already have integrated sensors and provide ready-to-use software so that mere material costs should be lower. The expert also pointed out that the indicated hardware costs were reasonable for prototypes but that he expects them to decrease for manufacturing of larger unit numbers. Software costs in genereal, however, were judged as more realistic. The system designing costs were considered somewhat unrealistic, the expert guessed that 200,000 € instead of the estimated 479,205 € would be more reasonable.

Nevertheless, the general approach and the software tool were considered a very helpful support for decision-making in early development phases as it provides the possibility to adjust estimations created by different methods. Particularly positive aspects pointed out were the provision of standard deviations and interquartile ranges as additional information on estimation uncertainty, the formalism of the approach and the availability of component cost models. Aside from the poor performance for the platform cost estimation, another mentioned disadvantage was the approach's dependency on up-to-date cost models.

8.2.2.5. Assistance with Interior Finishing Works

The scenario for interior finishing works outlines a service robot that is designed to assist with typical work activities associated with dry-wall mounting like installing profile rails, drilling holes and sanding down walls. It mainly consists of a mobile platform and a multi-purpose robot arm which can operate a variety of tools.

The original direct cost estimate amounts to 146,500 € and the software development cost to 4,299,800 €. Prototype cost and product unit cost estimates are deduced from these numbers to be 4,446,300 € and 252,364 €, respectively.

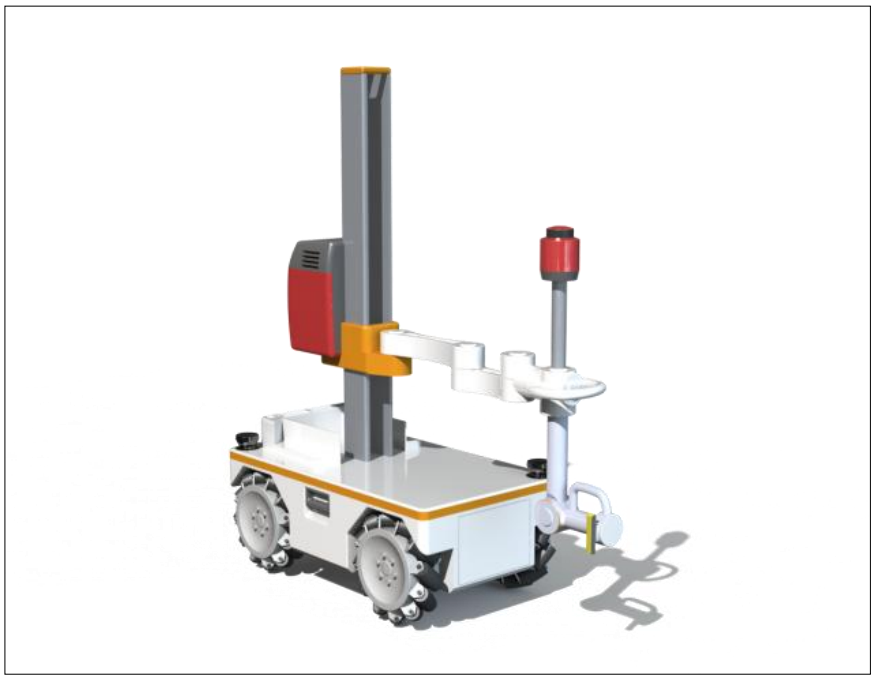

Figure 8.2-6 Design sketch of service robot for interior finishing works

The new estimates for cost constituents and aggregates are displayed in Table 8.2-9 and Table 8.2-10. One noteworthy phenomenon for this scenario is that the given robot design includes more miscellaneous hardware components than any of the other examined scenarios, i.e. eight components could not be estimated using the precalculated models. These components include the miscellaneous end effectors or tools power drill, power cutter and the combination of 2-DOF and a 1-DOF module as well as the miscellaneous structural hardware linear axis, bayonet tool connection, laser projector, 2D rotary laser and microphone. For these components, the original estimate was used. The distance measuring sensor skin mentioned in the original was approximated by 20 binary sensors.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Material (per unit)	84,559.81 €	11,633.82 €	76,737.61 €	92,382.01 €
Hardware installation (per unit)	29,863.64 €	19,022.73 €	14,745.20 €	40,120.50 €
Administration	34,650.00 €	1,650.00 €	33,450.60 €	35,849.40 €
Software Development	3,942,451.36	932,031.97 €	3,255,123.00	4,498,294.42
Software installation (per unit, not for prototype)	568.18 €	190.91 €	418.62 €	693.92 €
System Designing	994,240.53 €	305,105.40 €	770,532.07 €	1,174,866.98

Table 8.2-9 New estimates for cost constituents for interior works scenario

All of the new cost estimates lie well within the predefined estimation error thresholds. The estimate for the direct unit costs was calculated to be 21.51% lower than in the original. Software development costs lie 8.31% below the original estimate. The prototype cost exceed the original estimate by 14.38% for the most comprehensive cost aggregation level, undercut it by 7.98% on the intermediate aggregation level and by 8.76% on the lowest level. All these differences can be regarded as low given that design details are only vaguely known. The product unit cost difference moves from -6.45% (most comprehensive aggregate) to -16.16%; hardware cost vary from 1.76% excess to -21.90%. This comparison reveals that the new approach can lead to estimation values very similar to the ones in EFFIROB. It should be noted that the amount of predetermined estimates for the miscellaneous components evidently reduces the room for deviations; however, the cost of these components only accounts for roughly a third of the material cost.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Aggregate variant: All cost constituents				
Prototype cost	5,085,765.34 €	980,955.06 €	4,369,268.14 €	5,676,340.53 €
Product unit cost	236,081.28 €	32,656.23 €	212,858.21 €	256,467.55 €
Hardware cost	149,073.45 €	22,359.17 €	133,013.24 €	162,623.21 €
Aggregate variant: Excluding system designing cost				
Prototype cost	4,091,524.81 €	932,300.18 €	3,402,846.80 €	4,646,102.12 €
Product unit cost	212,408.89 €	31,837.99 €	189,606.20 €	232,020.62 €
Hardware cost	149,073.45 €	22,359.17 €	133,013.24 €	162,623.21 €
Aggregate variant: Excluding system designing cost, hardware admin. cost				
Prototype cost	4,056,874.81 €	932,298.67 €	3,367,923.91 €	4,611,178.97 €
Product unit cost	211,583.89 €	31,837.96 €	188,783.58 €	231,196.98 €
Hardware cost	114,423.45 €	22,298.20 €	98,561.67 €	127,196.33 €
Aggregate invariant				
Direct unit cost	114,991.63 €	22,299.02 €	99,117.96 €	127,828.56 €

Table 8.2-10 Cost aggregate estimations for interior works scenario

The expert's judgment of the estimation is positive concerning most cost constituents costs except material costs and consequently hardware costs on the whole. Whereas other cost estimates were considered reasonably realistic, material costs were considered to be too low for prototypical development but as reasonable for larger numbers of produced units; the expert pointed out the importance of negotiating power when determining hardware component prices. The approaches overall usefulness was regarded as helpful to some degree.

8.3. Comparison with the Raser project

The second validation method compares the estimates of the presented approach with early estimates for a real service robot project and the actual costs incurred. Before explaining this method two general problems concerning it shall be pointed out.

Firstly, knowledge on the final design of the robot has the potential of causing a bias towards the actual implementation thus artificially reducing uncertainty in a way which does not exist in real a priori estimation situations. Secondly, costs of input factors for R&D activities often lack direct reference to a specific project; if they do they tend to be broken down in only coarse categories like personnel and non-

personnel expenses (Gerpott 1999, p. 79). Due to this coarsely granular cost documentation only material and labor costs as well as derived aggregates are considered for this method, the clear distinction between hardware and software costs is not feasible.

The Raser project was chosen for validation because the documentation of the project including invoices, budget planning and technical information was considered as particularly detailed and the objective of the project – the development of two functional prototypes for a small lawn-mowing service robot – fit well with the aim of the estimation approach.

Figure 8.3-1 The functional Raser prototype

8.3.1. Assessment Method

The assessment method for comparing the Raser project cost data with new estimates is similar to the previous validation method with a few modifications.

First, the basic parameters for the estimates were established. The monthly wage rate documented amounts to 15,600 € per month at 20 workdays per month. This value is therefore used for the new estimates in order to guarantee comparability. As this relatively high rate includes not only ancillary labor costs but also burdened costs for administration and equipment the cost constituent hardware

administrative costs is left out in this analysis to avoid multiple accounting for the same costs.

The number of units was set to one because this project was aimed at creating a prototype only and no data was available on the market potential. As the actual project produced two variants that basically differed in complexity of the control software, the original material costs for one unit were assumed to be half of the overall material cost; software development costs are not affected by the number of units. The production of only one prototype also entails that no product unit cost or direct unit cost estimate was calculated.

The team's software development experience was assumed to be above average given that the team consisted of service robot experts at the department for robot systems at the Fraunhofer Institute for Manufacturing Engineering and Automation with 40 years of experience in the field (Fraunhofer IPA 2011). Thus estimates were calculated for the team levels "good" and "excellent". The code was assumed to be specific for a lawn-mowing robot and thus the productivity rate for non-reusable development was applied.

The comparison with the Raser project not only permits comparison with actual costs but also with the original a priori cost estimation. Therefore, the next step of this validation method consisted in ascertaining the original estimate and the actually incurred costs. The estimates for labor and material costs were adopted from the project proposal. The actual costs were investigated from invoice, monthly progress reports and workforce allocation plans. The underlying assumption in this context is that the cost information is complete and no actual cost blocks remained undocumented.

In the next step, the desired skills and thus required hardware and software components had to be determined in preparation of the computation of the new estimate. To this end project proposal, documented brainstorming sessions, and technical sketches were analyzed so that the components and their parameterizations could be extrapolated. Where no exact information was available e.g. for the maximum operation distance of ultrasonic sensors plausible assumptions had to be made. This can be viewed as a realistic situation because in early design phases these design details are commonly not known either.

After the determination of components, material and labor costs were computed, labor costs being system designing costs, hardware installation costs and software development costs. Two different aggregate levels of prototype costs were calculated, one including system designing costs and one without them. This was done in order to assess which estimate level more adequately approximates the real costs and was deemed necessary given the large model uncertainty for system designing cost (s. Chapter 6.3.2). For each aggregate level the labor costs were calculated for the two team experience levels "good" and "excellent" as outlined above.

Calculating the differences between estimates and actual costs permits the evaluation of the estimation; the smaller the difference the better the estimate. Both original and new estimates are compared to the actual costs; furthermore both estimates are compared directly to each other. Finally, the new estimations and the comparisons were presented to one of the experts responsible for the Raser prototype development and his opinion elicited in order to achieve qualitative validation.

8.3.2. The Raser lawn-mowing service robot

The costs for the Raser robot project were extracted from the proposals for the construction and optimization of a functional prototype. Because two almost identical units were built the documented material costs are divided by two to arrive at the hardware costs per unit. This yielded estimated material costs of 13,500 € per robot; labor costs were estimated as 258,960 €. The actual costs amounted to 9,670 € per robot material and 391,089 € for labor. The estimated prototype costs which were extrapolated as the sum of material cost per robot and labor costs thus were 272,460 €, actual prototype costs amounted to 391,089 €. This means that the actual costs were 28.37% lower than originally estimated whereas labor costs exceeded the estimate by 51.02% and prototype costs were 47.09% higher than estimated.

The various aggregates of the new estimates are displayed in Table 8.3-1, details on the estimation parameters are listed in Appendix 10.5.

Cost type	Expected value	Standard deviation	First quartile	Third quartile
Team software development experience level: good				
	Aggregate variant: Excluding hardware administrative costs			
Prototype cost	697,746.04 €	89,501.21 €	634,990.86 €	753,767.07 €
Labor cost	685,943.14 €	89,478.31 €	622,682.15 €	743,083.35 €
	Aggregate variant: Excluding system designing cost.			
Prototype cost	561,340.34 €	82,721.51 €	502,980.16 €	612,495.48 €
Labor cost	549,537.44 €	82,696.73 €	490,511.77 €	601,767.07 €
Team software development experience level: excellent				
	Aggregate variant: Excluding hardware administrative costs			
Prototype cost	583,690.31 €	74,488.64 €	531,368.88 €	629,081.57 €
Labor cost	571,887.41 €	74,461.12 €	518,892.57 €	618,761.25 €
	Aggregate variant: Excluding system designing cost.			
Prototype cost	469,581.91 €	68,809.66 €	420,905.04 €	510,746.11 €
Labor cost	457,779.01 €	68,779.86 €	408,263.85 €	500,269.26 €
Aggregate and experience level invariant				
Material cost	11,802.90 €	2,024.78 €	10,441.51 €	13,164.29 €

Table 8.3-1 Cost estimates for the Raser service robot

These estimates were contrasted with the original estimation values and the actual costs. An overview of the results is given in Table 8.3-2. The shaded areas indicate where the new estimates are closer to the actual costs than the original estimate. The differences indicate that material costs are overestimated by 22.06% which can be considered a good estimate at an early estimation phase; the additional consideration of component cost increases could possibly reduce the difference even further. Assuming that the team's experience level in software development is high i.e. 'excellent' labor and prototype cost estimates lie circa 17.1% higher than the actual costs which is also considered a solid estimate. Including the system designing cost estimate in the prototype and labor costs drives the estimation error significantly up which can be interpreted as an indication of them being systematically estimated too high. Given the high uncertainty for the estimation model of system designing cost this phenomenon seems to demand the development of an improved cost model for this cost constituent.

The comparison results also show that apart from the component selection the assessment of the team's development experience has a significant impact on the

estimation results. As no stringent metrics exist for the measurement of development experience this factor leaves room for subjective judgment.

	Team Exp. Level: Good		Team Exp. Level: Excellent	
	Diff. to Raser	**Diff. to Raser**	**Diff. to Raser**	**Diff. to Raser actual**
Prototype cost (w/o admin.	156.09%	74.11%	114.23%	45.65%
Material costs	-12.57%	22.06%	-12.57%	22.06%
Labor costs (w/o admin.	164.88%	75.39%	120.84%	46.23%
Prototype cost (w/o admin. cost, designing cost)	106.03%	40.07%	72.35%	17.17%
Material costs	-12.57%	22.06%	12.57%	22.06%
Labor costs (w/o admin. cost, designing cost)	112.21%	40.51%	76.78%	17.05%

Table 8.3-2 Relative differences between new estimates and original values

The expert's opinion was predominantly positive; the only estimate considered somewhat unrealistic were hardware installation costs; the expert pointed out that the real costs had probably been higher because for some components custom parts had to be developed, e.g. a casing for the compass. Software development costs and material costs were judged to be very realistic and the latter to be equally realistic as the original estimate. Notably, the system designing cost estimate was not considered unrealistic: The expert explained that he considered the calculated numbers as reasonable for diligent robot designing but conceded that budget restrictions tend to cut down the actual time and thus costs spent for the according activities. Another noteworthy finding was that the expert found it impossible to judge the usefulness of information on quartiles and standard deviations as he considered himself inexperienced in the field of statistical evaluation.

The tool and approach were assessed as useful in combination with other estimation methods like expert judgment because they offer the possibility of comparing separately derived estimates. This statement can be regarded as confirmation of the approach's validity and shows that the main objective has been achieved.

8.4. Evaluation Results and Interpretation

Figure 8.4-1 displays the responses to the questions with ordinal response scales for the EFFIROB estimation scenarios, separated into the categories 'positive', 'negative' and 'neutral'. Table 8.4-1 reveals the median and average response values according to the evaluation method described above.

Responses to Q1 reveal that the overall cost estimates derived with the proposed approach can be considered equally realistic to the original estimates with a weak tendency towards a more positive assessment. The assessments for hardware material and installation costs (Q2) are dominantly considered realistic; however, the assessment of hardware administrative costs is ambiguous. Software development and installation costs (Q3) are largely regarded as realistic. The system designing cost estimate is considered unrealistic by the majority of experts; this item is the only one with a dominant negative assessment value. The availability of additional information by indicating interquartile range and standard deviation (Q5) is positive throughout barring one expert abstention. The helpfulness of the approach in combination with other estimation methods (Q7) was unanimously affirmed.

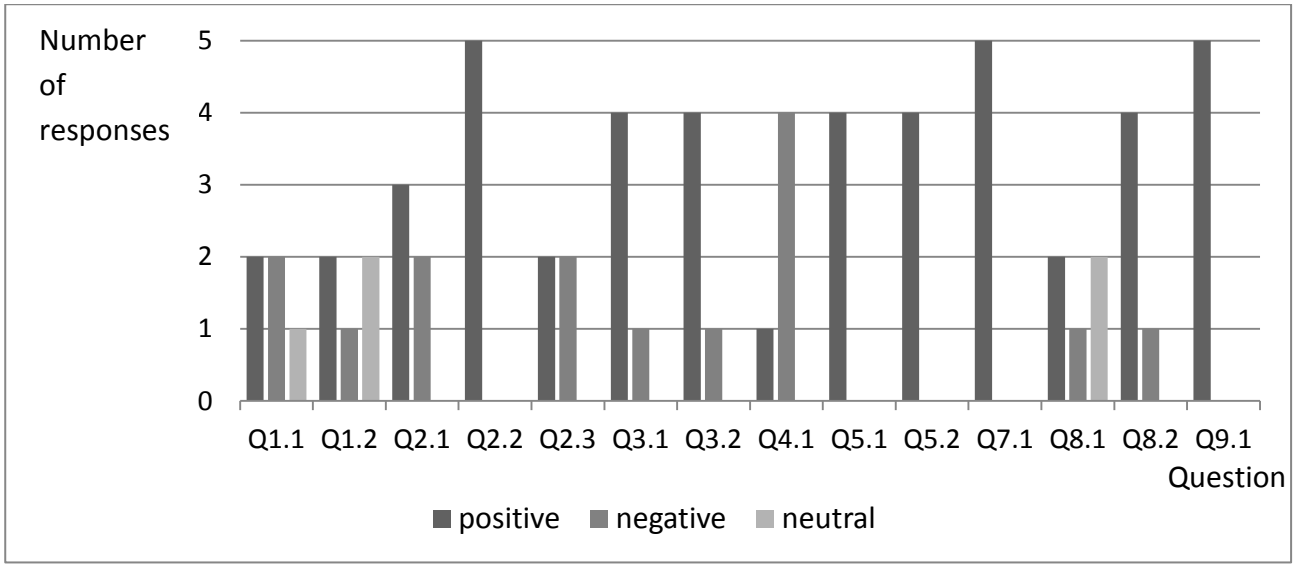

Figure 8.4-1 Cumulated responses for the EFFIROB estimation scenarios

Question	Q1.1	Q1.2	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q4.1	Q5.1	Q5.2	Q7.1	Q8.1	Q8.2	Q9.1
Median	0	0	1	1	0	1	1	-1	1.5	1.5	1	0	1	2
Average	0.2	0.2	0.4	1.2	0	0.8	0.6	-0.8	1.5	1.5	1.4	0.2	0.6	1.6

Table 8.4-1 Median and average values of responses for the EFFIROB estimation scenarios

The responses to Q8 indicate that software cost estimates resulting from the presented approach tended to be preferred over the original EFFIROB estimate whereas the hardware cost estimates would be used to find a compromise between the new and the original value. All experts confirmed that they would probably or definitely use the presented approach for quick cost estimation of service robot prototype development (Q9).

Figure 8.4-2 shows the integer values assigned to the responses for the Raser scenario as described in Chapter 8.2.1.2. The zero value for Q1.1 indicates that the material cost estimates from SEROCOST and EFFIROB are considered as equally realistic; the zero values for Q8 signify that the expert deems a compromise between these two estimates preferable. The responses are generally consistent with the pattern of the responses from the EFFIROB scenarios; for Q5 no conclusions can be drawn as the expert abstained from assessing the usefulness of interquartile range and standard deviation. An exception is the assessment of system designing costs which were judged to be realistic for the Raser case whereas the majority of the EFFIROB expert considered the respective estimate for their case unrealistic.

Figure 8.4-3 shows the prevalence rates of aggregates from Q6 including the response for the Raser scenario. Even though some cost constituents were assessed less positively than others, the most frequently preferred aggregate for prototype cost is the one including all constituents. This can be interpreted as a confirmation that developers recognize the need to incorporate more than only direct costs in a cost estimate even though most articles on low-cost service robots only describe hardware material costs. A different perspective is offered for the product unit costs: Here, some experts prefer the all-encompassing cost aggregate whereas others favor direct unit costs only. The fact that none of the experts regarded all mentioned aggregates inappropriate can be taken as an indicator that the chosen aggregates reflect the information demands towards cost estimation correctly.

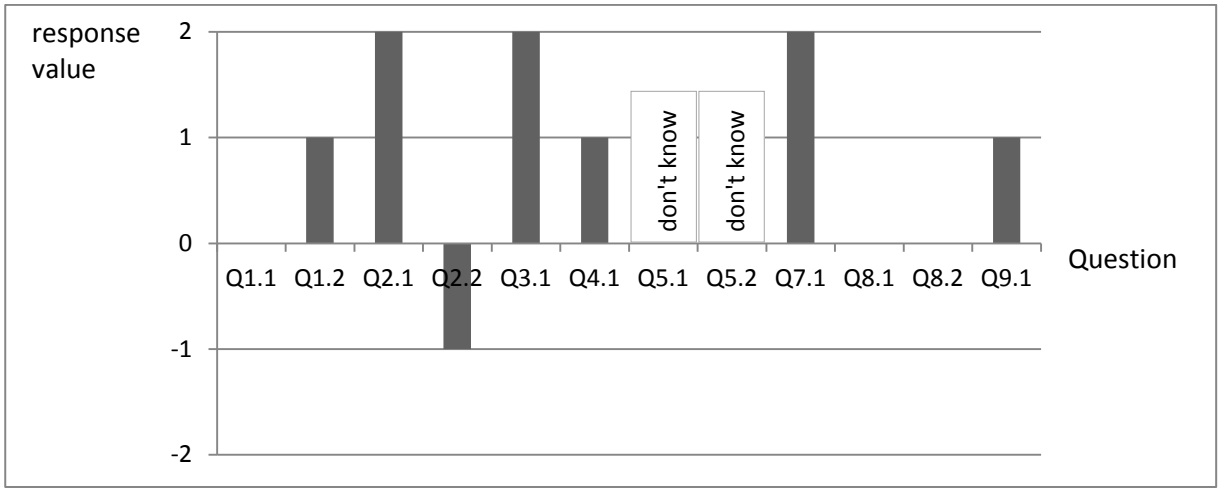

Figure 8.4-2 Response values for the Raser scenario

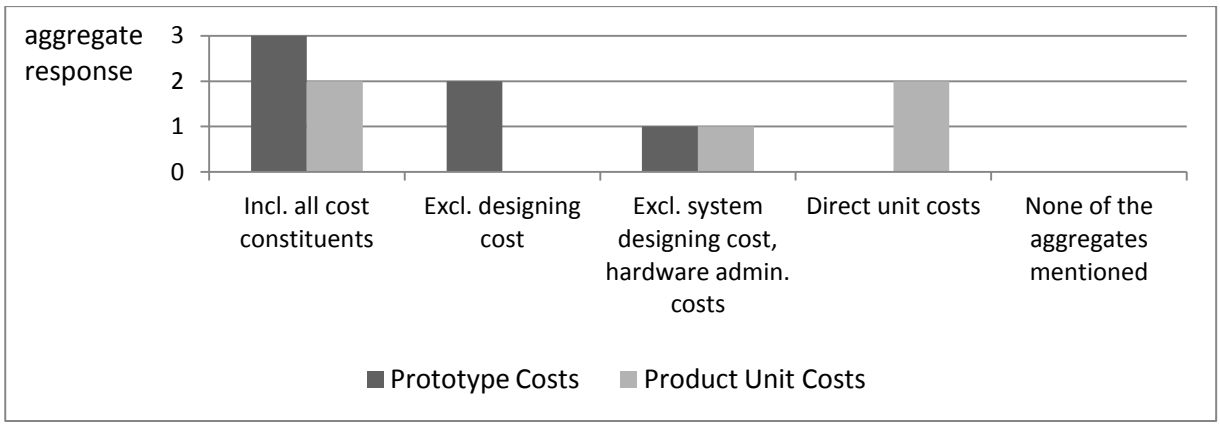

Figure 8.4-3 Preferred cost aggregates

An overview over the explicit comments stated during the expert interviews is given in Table 8.4-2; positive statements are marked with a plus sign, negative ones with a minus and neutral ones with a circular symbol.

Additional comments regarding the overall approach tend to underpin the positive assessment of the presented methodology. The most frequent positive statements stress the additional orientation and the possibility to compare estimates which are provided by the new approach. This allows the estimator to confirm their initial ad-hoc estimates or adjust them where appropriate. The availability of estimation ranges instead of point estimates was also appreciated as a means to express estimation uncertainty. Further praise was given to the strong formalization resulting in a high degree of traceability of estimates.

One comment categorized as neutral is aimed at the information value of interquartile range and standard deviation. Although the expert agreed on their usefulness as indicators for estimation uncertainty he pointed out that one of these would suffice.

A criticism concerning the overall approach was that the accuracy of the estimates strongly depends on models and estimation data base being up-to-date. This is a true and valid critique; however, it applies to all estimation approaches be it expert judgment or parametric modeling and thus can be considered a problem inherent to cost estimation in general.

Another group of comments addresses specific cost constituents or a single aspect surveyed in the questionnaire. Positive remarks were stated concerning the provision of hardware cost models, the suggesting of software components and the finer resolution of separate software parts into packages instead of stacks.[44] These perceived advantages were considered to give the estimator a higher degree of control of the estimation process and facilitate the estimation of the respective cost constituents which is in line with the objectives of this thesis.

Two neutral comments concerning specific cost constituents were given. Regarding hardware installation costs, one expert noted that install costs will differ for prototypes and series products. This is a correct observation; however, the product unit costs are only calculated for a production situation similar to the prototype construction i.e. manual assembly is assumed, the effects of cost reduction through production process optimization are not considered in the presented approach (s. Chapter 2.2.1 and 2.2.3).

Another neutral statement stressed the difference between the estimation of costs as a function of required effort and the determination of actual budgets for designing a robot system; the expert pointed out that a given budget can lie below the required amount for diligent prototype design thus discrepancies are prone to occur. This is in agreement with literature on cost estimation and planning (McConnell 2006, pp. 259–270).

[44] A ROS stack consists of one or more packages.

Related to	Comment	Occur-rence
Overall approach	✚ Additional orientation, possibility of comparison	3
	✚ Range estimates instead of point estimates	2
	✚ Formalization	1
	○ Redundancy between interquartile range and standard deviation	1
	━ Dependency on timeliness of data and models	1
Specific cost component	✚ Provided hardware material cost models (Q2.1)	1
	✚ Software component suggestions (Q.3.1)	1
	✚ Higher granularity for software estimation compared to EFFIROB (Q3.1)	1
	○ Differentiation between hardware install cost for prototype and product unit costs (Q2.2)	1
	○ Discrepancy between design effort necessity and actual budgeting (Q4.1)	2
	━ Term 'hardware administrative costs' misleading (Q2.3)	1
	━ Hardware administrative cost estimate too high (Q2.3)	1
	━ Software installation cost estimate too low (Q3.2)	4
	━ System designing cost estimate too high (Q4.1)	
Software tool SEROCOST	○ Tool supports but cannot replace human expertise	1
	○ Importance of tool usability	2

Table 8.4-2 Overview of expert comments on the estimation approach

Specific criticism was directed at the used cost terminology and the estimation of certain cost constituents. Two experts found the term 'hardware administrative cost' to be misleading or difficult to interpret. As this term is explicated in the presented approach as consisting of purchasing miscellaneous activities required for the procurement of hardware components it could be renamed to 'hardware procurement cost'; however, this new term could be interpreted as including material costs. Thus and considering that the rest of the experts did not criticize the term it was left unchanged.

The cost estimate for hardware administrative costs was explicitly commented on as being too high by one expert, another expert found the software installation costs to be too low; four experts pointed out that they considered the estimate for system designing being much too high. These comments are in line with the given responses (s. Figure 8.4-1 and Figure 8.4-2). The fact that four experts saw the need to explicitly comment on the system designing cost can be regarded as a strong indicator that further research is required into this specific cost constituent which is plausible given the high model uncertainty indicated in Chapter 6.3.2.

Explicit comments on the software tool address its scope of usage and its usability. One expert underlined that even though the tool is helpful in supporting cost estimation it cannot replace human intuition. This statement is valid because estimation is not deterministic and strongly depends on situative contingencies it can be argued that a certain degree of subjective expert judgment will influence cost estimates of all estimation approaches.

Two experts also stressed that the tool's user-friendliness is paramount to the acceptance of the approach; the GUI was generally considered as an appropriate way to provide ease of use. In order to comply with this requirement the application SEROCOST has been tested continuously with test cases. The demonstration of its functionality has satisfied the experts to which the tool was demonstrated.

9. Conclusion

9.1. Contributions

This thesis proposes a systematic, formalized and traceable method of early phase cost estimation for service robot prototypes thus increasing transparency and support of the pertinent decision-making processes in the development. The integration of technical and economic aspects was achieved by a combination of adequate methods originating in research areas of system engineering, project management and cost accounting.

The technical structure of the planned robot forms the basis of the cost estimation. The dilemma caused by the lack of definite design plans or system architectures typical in early development phases has been approached by formalizing the deduction of necessary hardware and software using knowledge on structural dependencies between desired functionalities and thus required components. Generic categorizations have been developed for skills, hardware component types and software component types and interconnected by structure matrices. Typical dependencies have been established within an expert workshop.

The economic perspective i.e. the derivation of cost estimates has been implemented by identifying central cost constituents and developing dedicated cost models for each one of them. These cost models are based on current cost theory and statistical research and represent the key innovation of the presented approach.

Hardware material costs have been modeled employing non-parametric regression analysis on hardware component data collected by market research. For each hardware category, an individual regression model has been derived from the collected data and validated using statistical assessment methods.

The model for software development costs combines code sizing and analogy methods. To this end, the code analysis of one of the most popular robot software frameworks has been conducted; the resulting code sizes mapped to function points have been stored in a database forming the base of the software development cost model. Further parameters influencing productivity, software reuse and team experience have been included in the model.

Models have been developed for the labor costs resulting from hardware installation, hardware administration, software installation and overall system designing by employing expert opinions and statistical industry data. The validity of these models has been analyzed in a variety of use cases. These models contribute to extending the cost perspective on service robots as many current cost considerations are restricted to material costs.

An innovation of the implemented cost estimation is the computation of estimates based on probability density functions which permits the calculation of estimate ranges which can serve as a measure of estimation uncertainty. This information increases the interpretability of estimation results and has been assessed by experts to be highly useful.

To facilitate and speed up the application of the presented approach a software tool has been implemented that allows both quick first-guess calculation as well as calibration to more detailed scenario specific contexts. It also allows the extrapolation of unit costs beyond the prototype costs and the calculation of various cost aggregates. This tool is an additional innovation; to the knowledge of the author no other software dedicated to cost estimation for service robots exists as of 2012.

9.2. Discussion

The presented approach shows that realistic early phase cost estimation is reasonably possible even when little information on the planned service robot is available. The cost models have been based on statistical data and thus reduce subjectivity in the estimation process. A certain degree of individual assessment remains in the selecting items from the categories for robot skills, hardware and software.

Expert interviews revealed that the provision of categories for service robot skills, hardware component types and software component types are useful in identifying the technical structure as a basis for cost estimation because they reduce the chance of omitting relevant parts and in general render the structure-finding process more systematic. The collected dependencies proved to be a reliable indicator of required components although occasional custom changes to the

calculated selections are situationally required due to the large variety of generally conceivable service robot applications. As none of the categories claims to be complete in the sense that they cover all possible skills or component types, respectively, there is room for further expansion of these categories. However, too large categories could have counter-productive effects as larger numbers of items become increasingly difficult to handle in interdependency considerations.

The regression models for the hardware material cost models were assessed as realistic by experts although some models revealed high sensitivity to certain parameters that could lead to unexpected results. The reason for local inaccuracies of these models lies in the relatively small size of samples from which the models were computed. An exhaustive market research on each hardware component type is beyond the scope of this work but might provide more accurate models.

The software development cost estimation met positive reception by the experts. Most experts found that they would replace their prior estimates within the EFFIROB scenarios with the estimation results from the presented approach. Although some of the scenario-specific estimations were considered less realistic than prior individual estimates based on expert judgment all experts agreed that the estimates are plausible, traceable and, given the high uncertainty of early phase estimation, sufficiently accurate for first economic assessments of development plans. A drawback of the approach is that it requires periodical updating of the data on the software analogy (ROS) because service robot software has been and is expected to continue to be subject to significant changes.

Hardware and software installation costs models have been constructed separately in order to allow the clearer distinction between variable, unit specific costs and one-time cost. Although these are of relatively simple design they were unanimously considered as reasonably realistic for prototypes. One expert pointed out that he would expect the hardware installation costs to decrease significantly once the number of built units exceeds five or more robots; capturing this effect requires the analysis of economies of scale which is beyond the scope of the presented work as it focuses on the development of service robot prototypes.

This thesis also introduced cost models for hardware administrative costs and system designing costs as labor cost constituents within the development of service

robot prototypes. These cost models met the strongest skepticism of the experts. Although all experts agreed that the inclusion of these cost constituents is useful the concrete estimation values where generally considered as too high. Whereas the hardware administrative costs were judged to be at least within reasonable ranges, system designing costs were considered by most experts as too excessive by a factor of two to four. A potential cause for this inaccuracy of these cost models lies in the lack of concrete data and difficulty of cost attribution for these cost blocks. Further research is required to provide more accurate cost models.

The formalized method cannot and is not supposed to replace expert estimation due to a variety of circumstances. The categorizations of skills, hardware and software components are not all-encompassing; thus, many service robot designs will display features that are not covered by one of the presented categories and consequently cannot be estimated by a pertinent cost model. Previously mentioned local inaccuracies of the cost models also advise expert scrutiny of the results. Furthermore, experts often hold implicit knowledge on manufacturing processes, prime costs, special requirements and other relevant factors impacting on development costs that are not integrated in the cost models. However, the approach offers the possibility to compare the expert's assessment to a systematically derived estimate. Also, the expert can adjust various parameters of the presented approach to map his specific experience and better calibrate the results to the individual estimation situation.

All experts agreed that the approach in combination with the software tool pose a useful means for quick cost estimation in early service robot prototype development phases. The availability of estimation ranges and the short time required to arrive at an order-of-magnitude estimate were pointed out as particularly supportive.

In conclusion, the presented work has shown that the systematic and formalized cost estimation of service robot prototypes in early development stages is feasible by applying a combination of structure information and individually constructed cost models. The calculated estimation ranges provide additional value concerning uncertainty assessment. The estimates based on the presented approach tend to be slightly higher than individual expert estimate which is in line with studies showing that costs of many projects have been underestimated in the past. Literature on cost

estimation recommends the combination of various estimation methods for reliable results (Williams 1994, p. 4). Thus, this work is a sensible addition to the development of service robot prototypes.

9.3. Outlook

The presented work forms the basis for the systematic consideration of cost relevant aspects in the development of service robots. Obvious extensions are the broadening of the categories for skills, hardware and software components and the refinement of cost models. Including more items in said categories could render the approach amenable to a larger variety of service robots. The refinement of cost models could also have this effect if the data basis and thus the borders for the considered parameters of the models were expanded.

Furthermore, the approach could be extended by considering additional parameters like complexity or quality, either on component or system level. Inversely, the data on structural dependencies employed in the presented approach might also be used to derive a measure for a specific robot's complexity e.g. the number of components used (e.g. Barclay 2000; Wuang 2010; Tani 2009). Adding a temporal dimension might also be worthwhile considering e.g. in the form of schedule estimation.

Accounting for economies of scale is another possible extension of the approach. This could also imply the departure from the assumption of manual assembly (s. Chapter 2.2.1) and open the approach to cost estimation and analysis of unit costs for larger production numbers.

10. Appendices

10.1. Cost Types

The perspectives used and their applications within the scope of this work are explained in this section (for the following delineations of costs, s. Coenenberg 2009, pp. 57–95; Porter 2000, pp. 63–96).

10.1.1. Classification by Factor Consumption

The classification by consumption of production factors presented here is strictly from an industrial management point of view. There are different classifications for production factors, most notably those in the field of macroeconomics which are neither used nor elaborated in this work. This classification is considered to be the basic framework of most cost accounting approaches (Coenenberg 2009, pp. 62–63).

10.1.1.1. Material Costs

Material costs can be subdivided into operating material costs, i.e. those costs forgone in the production of goods or services, e.g. lubricants, small maintenance articles, and basic material costs, i.e. costs incurred for the materials that make up the actual product. As operating material costs usually constitute only a minor cost factor they are not considered in the presented cost estimation. Material costs within the scope of this approach are the costs for physical components of the robot.

10.1.1.2. Labor Costs

Labor costs originate in the deployment of workers and employees in business operation and typically take form in wages and salaries but also include further costs caused by the use of manpower, fringe costs like employee benefit costs.

In the scope of the presented approach labor costs constitute one of the largest overall cost components. Development costs, notably software development costs are mostly labor costs. Labor costs not considered directly in this approach are the indirect labor costs incurred by administration of human resources, e.g. the cost for

finding adequate developers (s. Chapter 2.2.2). However, these can be incorporated in the cost estimation by assigning a markup on the labor wage rate.

10.1.1.3. Capital Equipment Costs

Capital equipment costs are those costs that originate from rendering available operational machinery, buildings and other investment goods necessary for productive operation. The most frequent costs of this type are acquisition costs and capital costs e.g. interests.

In the presented work capital equipment costs are not considered as the assumption of manual assembly exclude the necessity of product-specific equipment acquisitions (s. Chapter 2.2.1 and 2.2.2).

10.1.2. Classification by Organizational Function

The classification by organizational functions distinguishes costs according to their locus within the organization, i.e. assigning costs to so-called cost centers (Coenenberg 2009, pp. 103–119; Porter 2000, p.63-76, pp.98-107). The most common categories for cost centers according to their organizational function are cost centers for material, production, marketing and sales, administration and research and development as well as miscellaneous secondary cost centers that provide general services to the other cost centers (Coenenberg 2009, pp. 106–107).

A further common distinction is that of primary and support activities based on a concept by Michael Porter, the value chain (Porter 1996; Porter 2000, pp. 63–96). The structure of the value chain is depicted in Figure 10.1-1 (adopted from Porter 2000, p. 66). In order to enhance comprehensibility and comparability the cost types by organizational structure underlying the presented approach are matched against the value chain categorization.

10.1.2.1. Research and Development Costs

Research and development (R&D) is the first phase in a product's lifecycle from the manufacturer's point of view. As the term suggest these are not directly tied to production but a necessary precondition. Commonly, research is understood to be the striving for fundamental knowledge whereas development entailing the product's planning and designing tends to emphasize product implementation. In

relation to Porter's value chain, R&D costs are primarily technology costs (ibid., p. 73).

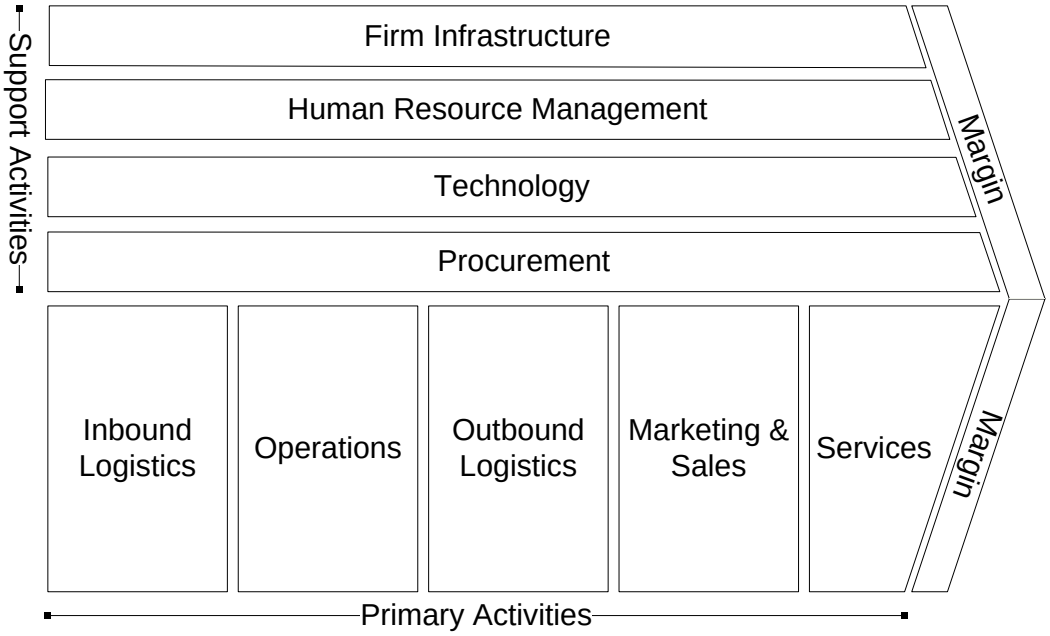

Figure 10.1-1 Porter Value Chain

In the scope of the presented work research cost are not considered in accordance with the assumption that no fundamental research is undertaken (s. Chapter 2.2.4). Evidently, development costs are in the focus of the presented approach and thus constitute one of most important parts of the cost estimate. Following the classification by factors, they are composed of labor costs and material costs, capital investment costs are not considered (s. Chapter 2.2.1 and 2.2.2).

10.1.2.2. Acquisition and Procurement Costs

Acquisition and procurement entail the costs for the input material as well as the supply costs, occasionally also referred to as warehouse costs or logistics costs. They are related to the purchase and supplying of production factors needed for the production of goods or delivery of service. These are costs for materials and for labor costs spent on necessary activities of rendering the production factors available, e.g. costs for stocking and transporting. The hardware costs prevail in the

direct material costs, as software material costs for production are assumed to be negligible.[45]

The costs for these material-related activities are mirrored by inbound logistics and procurement in Porter's value chain; the costs of production input materials in a narrower sense are not explicitly covered in the value chain as Porter's concept focuses on activities (ibid., pp. 70–73).

In this work, acquisition and procurement costs are subsumed under production costs and to a minor degree under R&D costs.

10.1.2.3. Production Costs

Production costs in a narrower sense are costs incurred by turning the input i.e. production factors into actual products. These costs include processing and installing of parts and components. Production costs are mapped to operation costs in the value chain (ibid., p. 70).

The term 'production costs' is also commonly used in a broader sense, encompassing material costs and manufacturing costs, i.e. the costs for matter and activities necessary for the production of goods or services. In the scope of this work the term 'production costs' is applied in a more general sense including the material costs.

10.1.2.4. Marketing and Sales Costs

Marketing and sales costs are incurred by rendering the produced goods available on the market and selling them. Typical examples are costs for advertizing and creating and maintaining channels of distribution. The value chain has its homonymous correspondent but outbound logistics may also be subsumed under sales cost (ibid., p. 71).

Marketing and sales costs are not directly related to development and production but strongly depend on the company's policies – one can spend enormous amounts

[45]The purchase of software is disregarded due to the assumption that software either stems from own development or open-source reuse (s. Chapter 2.2.5).

of money for marketing a simple product or conversely very little on a very expensive product – thus they are not considered in the presented work.

10.1.2.5. Administration Costs

Administrative costs include a large variety of costs that are tied to managing organizational processes. Though not producing sellable goods per se administration orchestrates the operations of the company so goods and services come into existence in the first place. This indirect relation to the costs of produced units is usually captured by burden rates, i.e. markups on the direct unit production costs. According to the value chain administration costs are caused by firm infrastructure, human resource management and procurement (ibid., pp. 74–75).

Due to their indirect nature, administrative costs are only marginally considered in this approach which is aimed at capturing costs tied directly to the development and production of prototypical service robots. Thus only administrative costs related to the components of the service robot concept are incorporated in the cost estimation (s. Chapter 0).

10.1.3. Classification by Allocation

Classifying cost by allocation is the distinction between costs that can be explicitly assigned to a specific cost unit and costs which are impossible or difficult to allocate to cost objects like produced units or services delivered (Coenenberg 2009, pp. 63–64).

10.1.3.1. Direct Costs

Direct costs are costs which are unambiguously caused by a specific cost object. A lucid example is that of material costs incurred for the production of a product unit i.e. a piece of merchandize. Evidently, the basic material that went into its construction is allocated in the product unit and thus causes direct costs.

Cost objects or cost units are usually the units of manufactured end-products of the company. The problem with produced units as sole cost object is that many types of costs, e.g. development costs and administration costs cannot be related to single units. Thus, cost reflecting causation can be ordered in a hierarchical fashion with direct costs at its bottom and more indirect cost sources on higher levels.

Direct costing, also known as marginal costing only assigns costs to cost units which are inherently related to its production. In the scope of this work a two-tier cost allocation model is applied in order to map costs in accordance with their cause. The considered cost objects are produced units of service robots on the lower tier and the service robot as a product type on the higher tier. The cost estimation on the lower tier includes material and labor costs per unit, the cost estimation for the product type encompasses development costs.

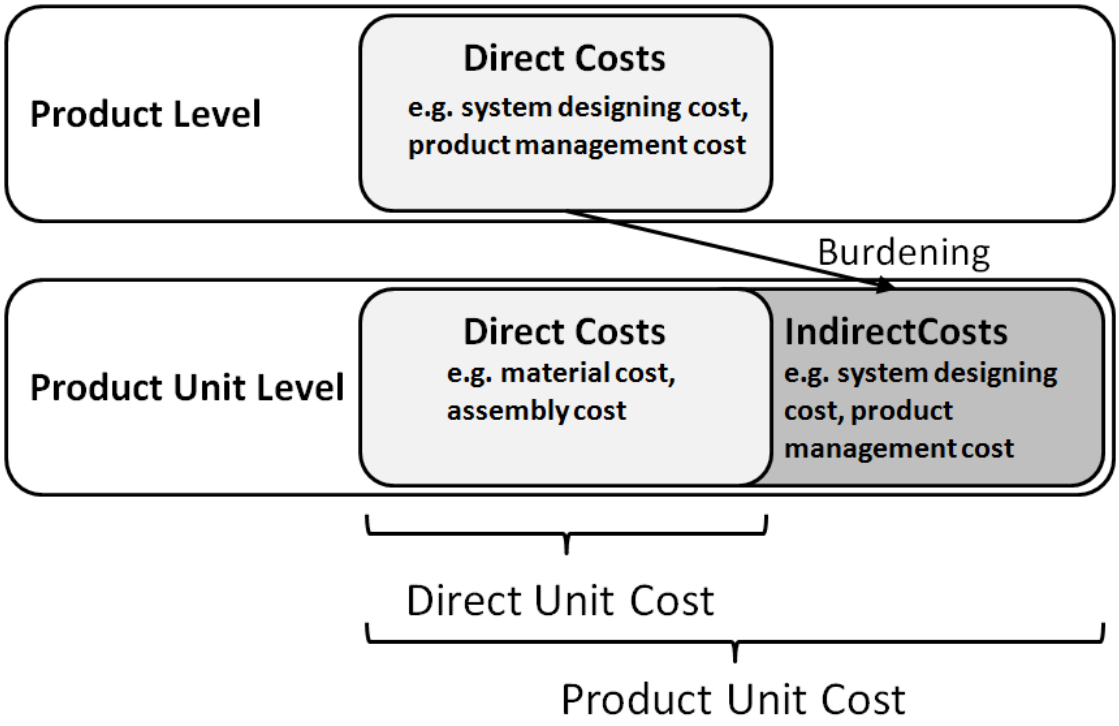

Figure 10.1-2 Two-tier product cost object

Product is used as a term for the aggregate of technical features of a specific product design and produced units are materialized entities of that design. This relation is depicted in Figure 10.1-2. The concept allows the allocation of development costs to the product but not (directly) to the produced units, in other words development costs for a product are direct costs on the product level but indirect costs on the level of produced units of this product; indirect costs from higher levels e.g. company administration are not considered here. This implies that the distinction between direct and indirect costs is relative to the cost object considered.

10.1.3.2. Indirect Costs

Indirect costs are costs not attributable to a cost unit and are incurred by business decisions affecting several cost units simultaneously. Typical indirect costs are administration costs and large capital investments.

The issue with indirect costs is that they do not have a revenue counterpart whereas direct costs can be contrasted with respective direct revenues. Nevertheless, they must be covered for an enterprise to be profitable. For this reason, several principles of cost imputation can be applied to distribute indirect costs among cost objects, remarkably the averaging principle, i.e. divide the indirect costs by the number of cost objects and assign evenly or in a weighted manner, and the principle of financial viability, i.e. cost objects generating higher revenues are assigned greater portions of indirect costs. This violation of the principle of causation is accepted because not all interdependencies between business-related processes can be disentangled (ibid., pp. 59–60).

The costing method of including indirect costs is called *absorption costing* or *full costing* and is a widely used accounting approach that burdens all costs incurred on the produced goods, i.e. the produced units 'absorb' all overhead costs within the considered time period via cost allocation rates.

Production costs are calculated as

direct material costs

+ indirect material costs

+ direct production costs

+ indirect production costs

+ special direct production costs.[46]

[46]There are different accounting standards, e.g. the US-American Generally Accepted Accounting Principles (US-GAAP), the International Accounting Standards 2 (IAS 2), the German Commercial Code (HGB), all of which differ in certain aspects defining production costs. Therefore, the

One major problem with this approach is the adequate distribution of large indirect costs incurred once before production e.g. development costs or plant investments. The dilemma is that these should be burdened on all accumulated produced units but their number is not known until production has been stopped. If the number of produced units is estimated an unrealistic estimate results in distorted unit cost and can thus bias the assessment of the product's marketability. Further problems arise if a company manufactures more than one product in which case burden rates must be defined for each product which is subject to the company's policy and legal accounting factors.[47]

Standard costing is almost synonymous to full costing but focuses on the planning aspect, i.e. *expected* indirect costs are burdened. As the presented work is assuming the a priori estimation of costs for prototypical service robots there is no difference between these terms in its scope.

The absorption costing approach is not applied in its explicit form because many cost blocks that have no relation to the development and production of service robots other than that they are incurred by the same company depend on too many firm-specific factors as to be considered in this work (s. Chapter 10.1.2). However, a partial absorption of indirect cost at product unit level is adopted as described in the previous section. Also, if the monthly cost rates for the work forces include not only wages and ancillary payroll related costs but also indirect overhead cost e.g. for administration personnel or asset depreciation, absorption of costs beyond the robot project at hand is implicit.

10.1.4. Classification by Variability

The classification of costs by their variability distinguishes between costs according to their behavior upon variation of their cost origin (ibid., pp. 65–68). Although there are many potential cost origins all of which should ideally be considered

---

description of production costs given here is a simplification for illustrative purposes. Detailing each of the mentioned accounting standards is beyond the scope of this work.

[47]These issues are not elaborated here because the presented approach only considers the development of a single service robot.

simultaneously, for practical reasons the variables are usually reduced to capacity utilization or occupation level and produced output. As the consideration of capacity utilization or occupation levels is beyond the scope of this work (s. Chapter 2.2.1, 2.2.2), only the number of produced unit as cost actuating variable is taken into account in the following explanations.

Similar to the distinction classification by allocation the distinction between variable and fixed costs is relative to the cost object regarded and furthermore relative to the time period regarded – from a long term perspective, all costs are variable.

10.1.4.1. Variable Costs

Variable costs are those costs that change along with the variation of the respective reference parameter. A prominent example in the scope of the presented work is given by the material costs which are incurred per produced unit.

Variable cost functions can be linear, degressive i.e. less than linear cost increase in relation to its origin, progressive i.e. greater than linear cost increase, or even regressive i.e. the total sum of variable costs of the reference objects decreases. As the production cost function is unknown at the time of estimation, linear variable cost functions based on produced output are assumed.

10.1.4.2. Fixed Costs

In opposition to variable costs fixed costs do not vary in regard to their cost origin within a certain time span thus they are categorical indirect costs. Typical examples are plant and equipment costs. A special case of fixed costs are stepped fixed or semi-fixed costs which do not change until the cost origin parameter crosses a threshold. At the threshold the cost increase creates a discontinuous step. Semi-fixed costs are not taken into account in this work because capital equipment investment based on output thresholds are beyond its scope (s. Chapter 10.1.1.3).

10.2. Software Package Development Effort Estimation: Value and Variance

The calculation of the expected value of the software development cost $E\left(C_{Development}^{SW}\right)$ and its variance $Var\left(C_{Development}^{SW}\right)$ is detailed in this section.

Assuming $Cov(s_i, r_i) = 0$, the estimated time and variance required for each package *i* of the components selection are computed as

$$
\begin{aligned}
E\left(t_{SW_{\mathrm{i}}}\right) &= E\left(\frac{s_i \cdot r_i}{v^R} + \frac{s_i \cdot (1 - r_i)}{v^D}\right) \\
&= E\left(\frac{s_i \cdot r_i}{v^R} + \frac{s_i \cdot (1 - r_i)}{v^D}\right) \\
&= E\left(s_i \cdot r_i \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right) + s_i \cdot \frac{1}{v^D}\right) \\
&= E(s_i \cdot r_i)\left(\frac{1}{v^R} - \frac{1}{v^D}\right) + E(s_i) \cdot \frac{1}{v^D} \\
&= \left(\frac{1}{v^R} - \frac{1}{v^D}\right) \cdot \sum_i E(s_i) \cdot E(r_i) + \frac{1}{v^D} \cdot \sum_i E(s_i)
\end{aligned}
\tag{10.1}
$$

and

$$Var\left(t_{SW_i}\right) = Var\left(\frac{s_i \cdot r_i}{v^R} + \frac{s_i \cdot (1-r_i)}{v^D}\right) = Var\left(s_i \cdot \left(\frac{r_i}{v^R} + \frac{1-r_i}{v^D}\right)\right) \tag{10.2}$$

$$= E(s_i)^2 \cdot Var\left(\frac{r_i}{v^R} + \frac{1-r_i}{v^D}\right) + \left(E\left(\frac{r_i}{v^R} + \frac{1-r_i}{v^D}\right)\right)^2 \cdot Var(s_i) + Var(s_i) \cdot Var\left(\frac{r_i}{v^R} + \frac{1-r_i}{v^D}\right)$$

$$= E(s_i)^2 \cdot Var(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2 + \left(E\left(\frac{r_i}{v^R} + \frac{1-r_i}{v^D}\right)\right)^2 \cdot Var(s_i) + Var(s_i) \cdot Var(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2$$

$$= Var(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2 \cdot \left(E(s_i)^2 + Var(s_i)\right) + Var(s_i) \cdot \left(\frac{1}{v^R} \cdot E(r_i) + \frac{1}{v^D} - \frac{1}{v^D} \cdot E(r_i)\right)^2$$

$$= \mathrm{Var}(s_i) \cdot \left(2\mathrm{Var}(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2 + \mathrm{E}(r_i)^2 \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2 + \frac{2}{v^R v^D} \cdot \mathrm{E}(r_i) + \left(\frac{1}{v^D}\right)^2\right) + \mathrm{Var}(r_i) \cdot \left(\frac{1}{v^R} - \frac{1}{v^D}\right)^2 \cdot \mathrm{E}(s_i)^2$$

10.3. Hardware Components

The following list contains indicates all parameters that were used for the cost models of the hardware components and the according mean, median, minimum and maximum values. Parameters in italics had been considered potential candidates for model parameters but have been excluded from the final cost models.

Component type	Parameter	Unit	Mean	Median	Min.	Max.
camera	frames per second	[none]	47,37	32	0	210
	sensor diagonal	[']	0,5	0,5	0,11	1
	pixel array size	[Pixel]				
ultrasonic sensor	blind range	[mm]	112,98	60	0	600
	maximal operating distance	[mm]	1336,643	500	100	6000
	sampling frequency	[Hz]				
laser scanner	scanning angle	[°]	36,55	0	0	360
	blind range	[mm]	90	100	0	300
	maximal operating distance	[mm]	52710	8000	30	300000
	protective field range	[m]				
binary sensor	maximal operating distance	[mm]	97,78	15	0	500
	blind range	[mm]	2,74	0	0	50
radar	volume	[mm³]	194802,2	8750	1898	742500
	broadcasting power	[dBm]				
force torque sensor	degrees of freedom	[none]	1,96	1	1	6
	measurable moment:	[Nm]	27,92	0	0	1000
	measurable force: Fz	[N]				
	volume	[mm³]				
gyroscope/ acceleration sensor	number of axes	[none]	1,9	2	1	6
	power usage	[mA]	7,45	4,2	0,14	25
	linear measuring range	[m/s²]				
	rotatory measuring	[°/s]				
	bandwidth	[kHz]				
GPS	-	-				
mobile platform	weight	[kg]	73,91	51	2,04	350
	footprint size	[m²]	0,45	0,35	0,08	1,61
	maximum payload capacity	[kg]	65,16	40	0	300
	maximum velocity	[km/h]				
robot arm	maximum payload capacity	[kg]	3,12	2	0,3	10
	weight	[kg]	14,45	12,8	2	28,9

	reach	[mm]	759,95	776,2	433	1300
	degrees of freedom	[none]				
gripper	jaw stroke	[mm]	36	37	6	80
	weight	[kg]				
	closing force/moment	[N]				
input peripherals	number of axes	[none]	2,12	3	0	6
	number of	[none]	29,91	5	2	100
output peripherals	pixel array size	[pixel²]	11365,88	8192	1	76800
	screen diagonal	[']				
	weight	[kg]				
controlling unit	processor type	[type]	9,95	9	1	17
	volume	[mm³]	13400000	5900000	11000	42900000
	flash memory capacity	[Gb]	9,23	0	0	32
	RAM capacity	[Mb]				
power supply (DC/DC converter)	output voltage	[V]	7,76	5	1,5	48
	output power	[W]	33,82	15	0,25	451
	input voltage	[V]				
power supply (battery)	capacity	[Ah]	17,91	12	0,65	100
	output voltage	[V]	10,54	12	3	15
	volume	[mm³]				
safety installation	weight	[kg]	21,49	6,3	0,15	116,7
	footprint size	[m²]	0,7	0,32	0	3,5

Table 10.3-1 Parameters of hardware component cost models

10.4. ROS Software Packages

The following packages of the Electric Emys release (April 2012) for the Robot Operating System have been used as an analogy for code sizing purposes:

Stack	Package
arm_navigation	arm_kinematics_constraint_aware
arm_navigation	arm_navigation_msgs
arm_navigation	collision_map
arm_navigation	collision_space
arm_navigation	constraint_aware_spline_smoother
arm_navigation	geometric_shapes
arm_navigation	joint_normalization_filters
arm_navigation	kinematics_base
arm_navigation	kinematics_msgs
arm_navigation	mapping_rviz_plugin
arm_navigation	motion_planning_rviz_plugin
arm_navigation	move_arm
arm_navigation	ompl
arm_navigation	ompl_ros_interface
arm_navigation	planning_environment
arm_navigation	planning_models
arm_navigation	robot_self_filter
arm_navigation	sbpl
arm_navigation	spline_smoother
arm_navigation	trajectory_filter_server
arm_navigation_experimental	arm_navigation_experimental_tools
arm_navigation_experimental	chomp_motion_planner
arm_navigation_experimental	collider
arm_navigation_experimental	collision_checking
arm_navigation_experimental	collision_free_arm_trajectory_controller
arm_navigation_experimental	collision_proximity
arm_navigation_experimental	collision_proximity_planner
arm_navigation_experimental	collision_space_ccd
arm_navigation_experimental	distance_field
arm_navigation_experimental	head_monitor_msgs
arm_navigation_experimental	interpolated_ik_motion_planner
arm_navigation_experimental	move_arm_head_monitor
arm_navigation_experimental	move_arm_warehouse
audio_common	audio_capture
audio_common	audio_common_msgs
audio_common	audio_play
audio_common	sound_play
bond_core	bond
bond_core	bondcpp
bond_core	bondpy

bond_core	smclib
brown_remotelab	rosbridge
bullet	bullet
camera_drivers	camera1394
camera_drivers	prosilica_camera
camera_drivers	prosilica_gige_sdk
camera_drivers	wge100_camera
camera_drivers	wge100_camera_firmware
client_rosjava_jni	rosjava_jni
client_rosjava_jni	tfjava
cob_command_tools	cob_dashboard
cob_command_tools	cob_script_server
cob_command_tools	cob_teleop
cob_common	brics_actuator
cob_common	cob_default_config
cob_common	cob_description
cob_common	cob_goco
cob_common	cob_srvs
cob_common	cob_utilities
cob_common	cob_vision_utils
cob_common	desire_description
cob_driver	cob_arm
cob_driver	cob_base
cob_driver	cob_base_drive_chain
cob_driver	cob_battery
cob_driver	cob_camera_sensors
cob_driver	cob_canopen_motor
cob_driver	cob_forcetorque
cob_driver	cob_generic_can
cob_driver	cob_head_axis
cob_driver	cob_hokuyo
cob_driver	cob_joint_state_aggregator
cob_driver	cob_joy
cob_driver	cob_light
cob_driver	cob_manipulator
cob_driver	cob_oodl_scanner
cob_driver	cob_powercube_chain
cob_driver	cob_pseudo_joint_state_publisher
cob_driver	cob_relayboard
cob_driver	cob_sdh
cob_driver	cob_sick_s300
cob_driver	cob_sound
cob_driver	cob_torso
cob_driver	cob_trajectory_controller
cob_driver	cob_tray
cob_driver	cob_tray_sensors
cob_driver	cob_undercarriage_ctrl

cob_environment_perception	cob_3d_mapping_msgs
cob_environment_perception	cob_3d_mapping_pipeline_fake
cob_environments	cob_default_env_config
cob_extern	brics_oodl_scanner_libs
cob_extern	libcvd
cob_extern	libhokuyo_urg
cob_extern	libm5api
cob_extern	libmesasr
cob_extern	libntcan
cob_extern	libpcan
cob_extern	libphidgets
cob_extern	libtoon
cob_people_perception	cob_people_detection
cob_people_perception	cob_people_detection_msgs
cob_simulation	cob_controller_configuration_gazebo
cob_simulation	cob_gazebo
cob_simulation	cob_gazebo_worlds
cob_simulation	cob_simulated_tactile_sensors
common	actionlib
common	bfl
common	tinyxml
common_msgs	actionlib_msgs
common_msgs	diagnostic_msgs
common_msgs	geometry_msgs
common_msgs	nav_msgs
common_msgs	sensor_msgs
common_msgs	stereo_msgs
common_msgs	trajectory_msgs
common_msgs	visualization_msgs
common_tutorials	actionlib_tutorials
common_tutorials	pluginlib_tutorials
common_tutorials	turtle_actionlib
control	control_msgs
diagnostics	diagnostic_aggregator
diagnostics	diagnostic_analysis
diagnostics	diagnostic_updater
diagnostics_monitors	robot_monitor
diagnostics_monitors	runtime_monitor
documentation	rosdoc
driver_common	driver_base
driver_common	dynamic_reconfigure
driver_common	timestamp_tools
executive_smach	smach
executive_smach	smach_msgs
executive_smach	smach_ros
executive_smach_visualization	smach_viewer
filters	filters

geometry	angles
geometry	eigen_conversions
geometry	tf
geometry	tf_conversions
geometry_experimental	tf2
geometry_experimental	tf2_bullet
geometry_experimental	tf2_geometry_msgs
geometry_experimental	tf2_kdl
geometry_experimental	tf2_msgs
geometry_experimental	tf2_ros
geometry_experimental	tf2_tools
geometry_tutorials	turtle_tf
geometry_visualization	tf2_visualization
ias_common	annotation_srvs
ias_common	cogman_msgs
ias_common	navp_action
ias_common	triangle_mesh_msgs
ias_common	vision_msgs
ias_common	vision_srvs
image_common	camera_calibration_parsers
image_common	camera_info_manager
image_common	image_transport
image_common	polled_camera
image_pipeline	camera_calibration
image_pipeline	image_proc
image_pipeline	image_rotate
image_pipeline	image_view
image_pipeline	stereo_image_proc
image_transport_plugins	compressed_image_transport
image_transport_plugins	theora_image_transport
imu_drivers	microstrain_3dmgx2_imu
joystick_drivers	cwiid
joystick_drivers	joy
joystick_drivers	ps3joy
joystick_drivers	spacenav
joystick_drivers	spacenav_node
joystick_drivers	wiimote
knowrob	bosch_semantic_map
knowrob	comp_cop
knowrob	comp_germandeli
knowrob	comp_orgprinciples
knowrob	comp_spatial
knowrob	comp_temporal
knowrob	ias_knowledge_base
knowrob	ias_prolog_addons
knowrob	ias_semantic_map
knowrob	jpl

knowrob	json_prolog
knowrob	knowrob_actions
knowrob	knowrob_common
knowrob	knowrob_objects
knowrob	mod_probcog
knowrob	mod_srdl
knowrob	mod_vis
knowrob	rosprolog
knowrob	semweb
knowrob	srldb
knowrob	tf_prolog
knowrob	thea
laser_drivers	hokuyo_node
laser_drivers	sicktoolbox
laser_drivers	sicktoolbox_wrapper
laser_pipeline	laser_assembler
laser_pipeline	laser_filters
laser_pipeline	laser_geometry
navigation	amcl
navigation	base_local_planner
navigation	carrot_planner
navigation	clear_costmap_recovery
navigation	costmap_2d
navigation	dwa_local_planner
navigation	fake_localization
navigation	map_server
navigation	move_base
navigation	move_base_msgs
navigation	move_slow_and_clear
navigation	nav_core
navigation	navfn
navigation	robot_pose_ekf
navigation	rotate_recovery
navigation	voxel_grid
nodelet_core	nodelet
nodelet_core	nodelet_topic_tools
object_manipulation	bayesian_grasp_planner
object_manipulation	compressed_pointcloud_transport
object_manipulation	current_state_validator
object_manipulation	household_objects_database
object_manipulation	household_objects_database_msgs
object_manipulation	interactive_marker_helpers
object_manipulation	object_manipulation_msgs
object_manipulation	object_manipulator
object_manipulation	point_cloud_server
object_manipulation	probabilistic_grasp_planner
object_manipulation	rviz_interaction_tools

object_manipulation	static_transform_broadcaster
octomap_mapping	octomap
octomap_mapping	octomap_ros
octomap_mapping	octomap_server
openni_kinect	depth_image_proc
openni_kinect	openni_camera
openni_kinect	openni_launch
openni_kinect	openni_tracker
orocos_kinematics_dynamics	orocos_kdl
orocos_kinematics_dynamics	python_orocos_kdl
perception_pcl	cminpack
perception_pcl	flann
perception_pcl	pcl
perception_pcl	pcl_ros
perception_pcl_addons	pcl_tutorials
perception_pcl_addons	pcl_visualization
perception_pcl_addons	terminal_tools
physics_ode	opende
physics_ode	parallel_quickstep
pluginlib	pluginlib
point_cloud_perception	point_cloud_converter
pr2_apps	pr2_app_manager
pr2_apps	pr2_mannequin_mode
pr2_apps	pr2_position_scripts
pr2_apps	pr2_teleop
pr2_apps	pr2_teleop_general
pr2_apps	pr2_tuckarm
pr2_arm_navigation	pr2_3dnav
pr2_arm_navigation	pr2_arm_navigation_actions
pr2_arm_navigation	pr2_arm_navigation_config
pr2_arm_navigation	pr2_arm_navigation_filtering
pr2_arm_navigation	pr2_arm_navigation_kinematics
pr2_arm_navigation	pr2_arm_navigation_perception
pr2_arm_navigation	pr2_arm_navigation_planning
pr2_arm_navigation	pr2_arm_navigation_tutorials
pr2_calibration	calibration_msgs
pr2_calibration	dense_laser_assembler
pr2_calibration	image_cb_detector
pr2_calibration	interval_intersection
pr2_calibration	joint_states_settler
pr2_calibration	laser_cb_detector
pr2_calibration	laser_joint_processor
pr2_calibration	laser_joint_projector
pr2_calibration	monocam_settler
pr2_calibration	pr2_calibration_estimation
pr2_calibration	pr2_calibration_executive
pr2_calibration	pr2_calibration_launch

pr2_calibration	pr2_calibration_propagation
pr2_calibration	pr2_dense_laser_snapshotter
pr2_calibration	pr2_se_calibration_launch
pr2_calibration	settlerlib
pr2_common	pr2_dashboard_aggregator
pr2_common	pr2_description
pr2_common	pr2_msgs
pr2_common_actions	joint_trajectory_action_tools
pr2_common_actions	joint_trajectory_generator
pr2_common_actions	pr2_arm_move_ik
pr2_common_actions	pr2_common_action_msgs
pr2_common_actions	pr2_tilt_laser_interface
pr2_common_actions	pr2_tuck_arms_action
pr2_controllers	control_toolbox
pr2_controllers	ethercat_trigger_controllers
pr2_controllers	joint_trajectory_action
pr2_controllers	pr2_calibration_controllers
pr2_controllers	pr2_controllers_msgs
pr2_controllers	pr2_gripper_action
pr2_controllers	pr2_head_action
pr2_controllers	pr2_mechanism_controllers
pr2_controllers	robot_mechanism_controllers
pr2_controllers	single_joint_position_action
pr2_ethercat_drivers	eml
pr2_ethercat_drivers	ethercat_hardware
pr2_ethercat_drivers	fingertip_pressure
pr2_gui	pr2_dashboard
pr2_kinematics	pr2_arm_kinematics
pr2_kinematics	pr2_arm_kinematics_constraint_aware
pr2_mechanism	pr2_controller_interface
pr2_mechanism	pr2_controller_manager
pr2_mechanism	pr2_hardware_interface
pr2_mechanism	pr2_mechanism_diagnostics
pr2_mechanism	pr2_mechanism_model
pr2_mechanism	pr2_mechanism_msgs
pr2_mechanism	realtime_tools
pr2_navigation	laser_tilt_controller_filter
pr2_navigation	pr2_move_base
pr2_navigation	pr2_navigation_config
pr2_navigation	pr2_navigation_global
pr2_navigation	pr2_navigation_local
pr2_navigation	pr2_navigation_perception
pr2_navigation	pr2_navigation_self_filter
pr2_navigation	pr2_navigation_slam
pr2_navigation	pr2_navigation_teleop
pr2_navigation	semantic_point_annotator
pr2_object_manipulation	active_realtime_segmentation

pr2_object_manipulation	fast_plane_detection
pr2_object_manipulation	object_recognition_gui
pr2_object_manipulation	object_segmentation_gui
pr2_object_manipulation	pick_and_place_demo_app
pr2_object_manipulation	pr2_create_object_model
pr2_object_manipulation	pr2_grasp_adjust
pr2_object_manipulation	pr2_gripper_grasp_controller
pr2_object_manipulation	pr2_gripper_grasp_planner_cluster
pr2_object_manipulation	pr2_gripper_reactive_approach
pr2_object_manipulation	pr2_gripper_sensor_action
pr2_object_manipulation	pr2_gripper_sensor_controller
pr2_object_manipulation	pr2_gripper_sensor_msgs
pr2_object_manipulation	pr2_handy_tools
pr2_object_manipulation	pr2_interactive_gripper_pose_action
pr2_object_manipulation	pr2_interactive_manipulation
pr2_object_manipulation	pr2_interactive_object_detection
pr2_object_manipulation	pr2_manipulation_controllers
pr2_object_manipulation	pr2_marker_control
pr2_object_manipulation	pr2_object_manipulation_launch
pr2_object_manipulation	pr2_object_manipulation_msgs
pr2_object_manipulation	pr2_pick_and_place_demos
pr2_object_manipulation	pr2_tabletop_manipulation_launch
pr2_object_manipulation	pr2_wrappers
pr2_object_manipulation	rgbd_assembler
pr2_object_manipulation	robot_self_filter_color
pr2_object_manipulation	segmented_clutter_grasp_planner
pr2_object_manipulation	tabletop_collision_map_processing
pr2_object_manipulation	tabletop_object_detector
pr2_object_manipulation	tabletop_vfh_cluster_detector
pr2_object_manipulation	vfh_recognition
pr2_object_manipulation	vfh_recognizer_db
pr2_object_manipulation	vfh_recognizer_fs
pr2_power_drivers	ocean_battery_driver
pr2_power_drivers	power_monitor
pr2_power_drivers	pr2_power_board
pr2_robot	imu_monitor
pr2_robot	pr2_bringup
pr2_robot	pr2_camera_synchronizer
pr2_robot	pr2_computer_monitor
pr2_robot	pr2_controller_configuration
pr2_robot	pr2_etherCAT
pr2_robot	pr2_run_stop_auto_restart
pr2_simulator	pr2_controller_configuration_gazebo
pr2_simulator	pr2_examples_gazebo
pr2_simulator	pr2_gazebo
pr2_simulator	pr2_gazebo_plugins
robot_calibration	camera_offsetter

robot_model	collada_parser
robot_model	collada_urdf
robot_model	colladadom
robot_model	convex_decomposition
robot_model	ivcon
robot_model	kdl_parser
robot_model	resource_retriever
robot_model	robot_state_publisher
robot_model	simmechanics_to_urdf
robot_model	urdf
robot_model	urdf_interface
robot_model	urdf_parser
robot_model_tutorials	urdf_tutorial
ros	mk
ros	rosboost_cfg
ros	rosbuild
ros	rosclean
ros	roscreate
ros	rosdep
ros	rosemacs
ros	roslib
ros	rosmake
ros	rospack
ros	rosunit
ros_comm	cpp_common
ros_comm	message_filters
ros_comm	rosbag
ros_comm	rosconsole
ros_comm	roscpp
ros_comm	roscpp_serialization
ros_comm	roscpp_traits
ros_comm	rosgraph
ros_comm	rosgraph_msgs
ros_comm	roslaunch
ros_comm	roslisp
ros_comm	rosmaster
ros_comm	rosmsg
ros_comm	rosnode
ros_comm	rosout
ros_comm	rosparam
ros_comm	rospy
ros_comm	rosservice
ros_comm	rostime
ros_comm	rostopic
ros_comm	roswtf
ros_comm	std_msgs
ros_comm	std_srvs

ros_comm	topic_tools
ros_comm	xmlrpcpp
ros_realtime	allocators
ros_realtime	lockfree
ros_realtime	rosatomic
ros_realtime	rosrt
ros_tutorials	roscpp_tutorials
ros_tutorials	rospy_tutorials
ros_tutorials	turtlesim
rx	rxbag
rx	rxdeps
rx	rxgraph
rx	rxtools
rx	wxPython_swig_interface
rx	wxswig
rx	xdot
schunk_modular_robotics	schunk_description
schunk_modular_robotics	schunk_powercube_chain
schunk_modular_robotics	schunk_sdh
simulator_gazebo	gazebo
simulator_gazebo	gazebo_msgs
simulator_gazebo	gazebo_plugins
simulator_gazebo	gazebo_tools
simulator_gazebo	gazebo_worlds
slam_gmapping	gmapping
sql_database	database_interface
sql_database	student_database
stage	stage
vision_opencv	cv_bridge
vision_opencv	cv_markers
vision_opencv	image_geometry
visualization	interactive_markers
visualization	rviz
visualization	rxbag_plugins
visualization	wxpropgrid
visualization_common	ogre
visualization_common	ogre_tools
visualization_tutorials	interactive_marker_tutorials
visualization_tutorials	visualization_marker_tutorials
warehousewg	mongo_ros
warehousewg	mongodb
warehousewg	pymongo
web_interface	ckill
web_interface	image_stream
web_interface	launchman
web_interface	pyclearsilver
web_interface	ros_apache2

web_interface	rosjson
web_interface	rosweb
web_interface	web_msgs
web_interface	webui
wifi_drivers	wifi_ddwrt
xacro	xacro

Table 10.4-1 ROS stacks and packages used for code sizing

The following files were excluded from function point derivation because they provide secondary functionality only e.g. package management or content administration:

File type
manifest.xml
stack.xml
shell scripts (.sh or similar)
makefiles (.make, .cmake)
init.py
rosdep.yaml
documentation (.dox)
configuration files (.config)
HTML files
ASP.Net files

Table 10.4-2 Excluded files

10.5. Verification Scenarios Parameterizations

The following tables give details on the various components selected for the computation of estimates using the SEROCOST tool and the indicated parameterizations. Only changes from the default suggestions are listed for the software selection; items marked with the prefix '-' have been removed from the scenario-specific selection, otherwise they have been added.

GH = Ground harvesting robot

CU = Provisioning of care utensils

CT = Container transport in hospitals

FC = Floor cleaning robot

IF = Assistance with interior finishing works

Ra = Raser lawn mowing robot

Basic parameter	**GH**	**CU**	**CT**	**FC**	**IF**	**Ra**
Wage rate (€)	10,000	10,000	10,000	10,000	10,000	15,600
Workdays per month	22	22	22	22	22	20
Code reusability	yes	yes	yes	yes	yes	no

Table 10.5-1 Basic parameters

Skills	**GH**	**CU**	**CT**	**FC**	**IF**	**Ra**
Perceive Objects	x	x	x	x	x	x
Recognize Objects	x	x	x	x		
Interpret Environment		x	x	x	x	x
Perceive evolutionary processes		x		x	x	
Move to Location		x	x	x	x	x
Orientate in Environment	x	x	x	x	x	x
Move Object	x	x	x		x	
Process/Alter Object	x			x	x	
Process/Alter Environment				x		x
Send Signals/Commands		x	x			
Interpret Signals/Commands					x	
Receive Signals/Commands	x	x	x		x	x

Table 10.5-2 Scenario-specific skills

Hardware components	GH	CU	CT	FC	IF	Ra
Camera	x	x	x	x		
Ultrasonic Sensor				x		x
Laserscanner	x	x	x	x	x	
Radar						
Binary Sensor				x	x	x
Force/Torque-Sensor	x	x		x	x	
Gyroscope/Acceleration						x
GPS System						
Wheel-based Platform		x	x	x	x	x
Robot Arm	x	x		x	x	
Gripper		x		x		
Miscellaneous End	x	x	x	x	x	x
Input Peripherals	x	x	x			x
Output Peripherals	x	x	x			x
Power Supply (batteries)	x	x	x	x	x	x
Power Supply (DC/DC	x	x	x	x	x	x
Controlling Computer	x	x	x	x	x	x
Safety Hardware		x				x
Miscellaneous Structural Hardware	x	x	x		x	x
Miscellaneous Hardware						x

Table 10.5-3 Scenario-specific hardware components

Software	GH	CU	CT	FC	IF	Ra
Object Detection	x	x	x	x	x	x
Object Recognition	x	x	x	x		
Object Modeling	x	x		x		
Environmental Modeling		x		x	x	
Change Detection		x		x	x	
Simultaneous Localisation		x	x	x	x	
Platform Path Planning		x	x	x	x	
Platform Control		x	x	x	x	x
Grasping & Grasp Planning	x	x		x		
Tool Control	x	x	x	x	x	x
Arm Control	x	x		x	x	
Arm Path Planning 3D/6D	x	x		x	x	
Operational Interface (HRI)	x	x	x		x	x
Robot-to-Robot						
Communication Protocols	x	x	x	x	x	x
Learning & Reasoning		x				
Drivers & Primitives	x	x	x	x	x	x
(Robot) Operating System	x	x	x	x	x	x

Table 10.5-4 Scenario-specific software components

Ground crop harvester

Component type	quant.	parameters	comments
Camera	6	frames per second:32	low-res
		sensor diagonal(inch):0.2	
Laserscanner	6	scanning angle(°):60	
		blind range(mm):100	
		maximal operating distance(mm):2000	
Force/Torque-Sensor	6	degrees of freedom:1	
		measurable moment Mz(Nm):50	
Robot Arm	6	maximum payload capacity(kg): 4	
		weight(kg):20	
		reach(mm):776	
Miscellaneous End Effectors/ Tools	6	expected cost(€):50	crop cutter
		estimated cost stand. dev.(€):20	
Input Peripherals	1	number of axes:1	touchscreen
		number of buttons/keys:50	
Output Peripherals (screens)	1	pixel array size(inch):76800	touchscreen
Power Supply (batteries)	0	capacity(Ah):12	
		output voltage(V):12	
Power Supply (DC/DC converter)	6	output voltage(V):24	
		output power(W):200	
Controlling Computer	1	processor type:9	
		volume(mm^3):590000	
		flash memory capacity(GB):0	
Miscellaneous Structural Hardware	6/6	expected cost(€):20000/48000	linear axle; gripper
		estimated cost stand. dev.(€):5000/8000	

Table 10.5-5 Ground harvester hardware parameters

ROS stack	package
camera_drivers	prosilica_camera
-robot_model	-collada_parser
	-collada_urdf
	-colladadom
	-convex_decomposition
	-ivcon
	-kdl_parser
	-resource_retriever
	-robot_state_publisher
	-simmechanics_to_urdf
	-urdf
	-urdf_interface
	-urdf_parser
-navigation	-robot_pose_ekf
pr2_controllers	robot_mechanism_controllers
pr2_kinematics	pr2_arm_kinematics
cob_command_tools	cob_dashboard
-common_msgs	-nav_msgs
-pluginlib	-pluginlib
-executive_smach	-smach
	-smach_msgs

Table 10.5-6 Ground harvester software selection

Provisioning of Care Utensils

Component type	quant.	parameters	comments
Camera	6/2/1	frames per second:32	
		sensor diagonal(inch):0.2/1/0.5	
Laserscanner	1	scanning angle(°):180	
		blind range(mm):100	
		maximal operating distance(mm):8000	
Force/Torque-Sensor	1	degrees of freedom:6	
		measurable moment Mz(Nm):100	
Wheel-based Platform	1	weight(kg):150	
		footprint size(m²):0.64	
		maximum payload capacity(kg):75	
Robot Arm	1	maximum payload capacity(kg):4	
		weight(kg):20	
		reach(mm):776	
Gripper	1	jaw stroke(mm):80	
Miscellaneous End Effectors/ Tools	1	expected cost(€): 5000	removal mechanism
		estimated cost stand. dev.(€):1000	
Input Peripherals	1	number of axes:1	touchscreen (input)
		number of buttons/keys:50	
Output Peripherals	1	pixel array size(inch):76800	touchscreen (output)
Power Supply (batteries)	8	capacity(Ah):12	
		output voltage(V):12	
Power Supply (DC/DC converter)	1/1/1	output voltage(V):5/12/24	
		output power(W):15/100/200	
Controlling Computer	1	processor type:9	
		volume(mm³):5900000	
		flash memory capacity(GB):0	
Safety Hardware	2	weight(kg):0.15	emergency stop
		contact surface size([m²):0.0025	

Miscellaneous Structural Hardware	1/1	expected cost():5000/500	linear axle, wlan
		estimated cost stand. dev.(€):1000/200	

Table 10.5-7 Provisioning of care utensils hardware parameters

ROS stack	**Package**
laser_pipeline	laser_assembler
	laser_filters
	laser_geometry
image_pipeline	camera_calibration
	image_proc
	stereo_image_proc
vision_opencv	cv_bridge
	cv_markers
	image_geometry
camera_drivers	prosilica_camera
slam_gmapping	gmapping
cob_driver	cob_canopen_motor
	cob_forcetorque
	cob_undercarriage_ctrl
executive_smach	Smach
	smach_ros
pr2_controllers	robot_mechanism_controllers
	pr2_gripper_action
pr2_object_manipulation	pr2_grasp_adjust
	pr2_gripper_grasp_planner_cluster
	pr2_gripper_reactive_approach
	pr2_gripper_sensor_action
	pr2_gripper_sensor_controller
	pr2_gripper_sensor_msgs
	pr2_interactive_gripper_pose_action
	pr2_manipulation_controllers
	pr2_object_manipulation_launch
	pr2_object_manipulation_msgs
	pr2_tabletop_manipulation_launch
	tabletop_collision_map_processing
	tabletop_object_detector
	pr2_gripper_grasp_controller
	pr2_interactive_manipulation

pr2_arm_navigation	pr2_arm_navigation_perception
wifi_drivers	wifi_ddwrt
cob_command_tools	cob_dashboard
	cob_teleop
knowrob	ias_knowledge_base
	knowrob_actions
	knowrob_common
	knowrob_objects
	mod_probcog
	mod_srdl

Table 10.5-8 Provisioning of care utensils software selection

Container transport in hospitals

Component type	quant.	parameters	comments
Camera	2	frames per second:32	
		sensor diagonal(inch):0.5	
Laserscanner	2	scanning angle(°):270	
		blind range(mm):100	
		maximal operating distance(mm):8000	
Wheel-based Platform	1	weight(kg):300	
		footprint size(m^2):1.8	
		maximum payload capacity(kg):275	
Miscellaneous End Effectors/ Tools	1	expected cost(€):750	lifter
		estimated cost stand. dev.(€):150	
Input Peripherals	1	number of axes:1	touchscreen (input)
		number of buttons/keys:50	
Output Peripherals	1	pixel array	touchscreen (output)
Power Supply (batteries)	8	capacity(Ah):12	
		output voltage(V):12	
Power Supply (DC/DC converter)	1/2	output voltage(V):5/24	
		output power(W):15/200	
Controlling Computer	1	processor type:9	
		volume(mm^3):590000	
		flash memory capacity(GB):0	
Miscellaneous Structural Hardware	1/1	expected cost(€):250/500	wlan, rfid/barcode reader
		estimated cost stand. dev.(€):50/100	

Table 10.5-9 Container transport in hospitals hardware parameterization

ROS stack	package
cob_driver	cob_camera_sensors
	cob_forcetorque
	cob_base
	cob_base_drive_chain
laser_pipeline	laser_assembler
	laser_filters
	laser_geometry
image_pipeline	camera_calibration
	image_proc
	stereo_image_proc
camera_drivers	prosilica_camera
perception_pcl	cminpack
	flann
	pcl
	pcl_ros
slam_gmapping	gmapping
pr2_controllers	robot_mechanism_controllers
wifi_drivers	wifi_ddwrt
cob_command_tools	cob_dashboard
	cob_teleop
control	control_msgs
web_interface	web_msgs
	rosweb
	webui

Table 10.5-10 Container transport in hospitals software selection

Floor cleaning

Component type	quant.	parameters	comments
Camera	1	frames per second:32	
		sensor diagonal(inch):0.5	
Ultrasonic Sensor	6	blind range(mm):60	
		maximal operating distance(mm):500	
Laserscanner	2	scanning angle(°):270	
		blind range(mm):100	
		maximal operating distance(mm):8000	
Binary Sensor	2	maximal operating distance(mm):100	step detection to avoid falling down stairs
		blind range(mm):0	
Force/Torque-Sensor	1	degrees of freedom:1	
		measurable moment Mz(Nm):5	
Wheel-based Platform	1	weight(kg):51	
		footprint size(m^2):0.2	
		maximum payload capacity(kg):40	
Robot Arm	1	maximum payload capacity(kg):4	
		weight(kg):20	
		reach(mm):776	
Gripper	1	jaw stroke(mm):37	
Miscellaneous End Effectors/ Tools	1	expected cost(€):1500	cleaning unit
		estimated cost stand. dev.(€):500	
Power Supply (batteries)	4	capacity(Ah):12	
		output voltage(V):12	
Power Supply (DC/DC converter)	1/1/1	output voltage(V):5/12/24	
		output power(W):15/100/200	
Controlling Computer	1	processor type:9	
		volume(mm^3):590000	
		flash memory capacity(GB):0	

Table 10.5-11 Floor cleaning hardware parameterization

ROS stack	package
cob_driver	cob_camera_sensors
	cob_base
	cob_base_drive_chain
	cob_forcetorque
camera_drivers	prosilica_camera
perception_pcl	cminpack
	flann
	pcl
	pcl_ros
vision_opencv	cv_bridge
	cv_markers
	image_geometry
image_pipeline	image_proc
	camera_calibration
laser_pipeline	laser_assembler
	laser_filters
	laser_geometry
object_manipulation	household_objects_database
cob_environment_perception	cob_3d_mapping_msgs
slam_gmapping	gmapping
executive_smach	smach
	smach_ros
pr2_controllers	robot_mechanism_controllers
	pr2_gripper_action

Table 10.5-12 Floor cleaning software selection

Assistance with interior finishing works

Component type	quant.	parameters	comments
Laserscanner	2	scanning angle(°):270	
		blind range(mm):100	
		maximal operating distance(mm):8000	
Binary Sensor	20	maximal operating distance(mm): 50	optical distance sensors for safety
		blind range(mm):0	
Force/Torque-Sensor	1	degrees of freedom:4	
		measurable moment Mz(Nm):150	
Wheel-based Platform	1	weight(kg):50	
		footprint size(m²):0.75	
		maximum payload capacity(kg):40	
Robot Arm	1	maximum payload capacity(kg):4	
		weight(kg):20	
		reach(mm):776	
Miscellaneous End Effectors/ Tools	1/1/1	expected cost(€):10000/4000/2000	2DOF module + 1DOF module, power drill,
		estimated cost stand. dev. (€):2000/1000/500	
Power Supply (batteries)	8	capacity(Ah):12	
		output voltage(V):12	
Power Supply (DC/DC	1/1/1	output voltage(V):5/12/24	
		output power(W):15/100/200	
Controlling Computer	1	processor type:9	
		volume(mm³):5900000	
		flash memory capacity(GB):0	
Miscellaneous Structural Hardware	1/1/1/1/1	expected cost(€):10000/1500/3000/1000/200	linear axle, tool bayonet connection, laser projector, 2D
		estimated cost stand. dev.(€):2000/300/500/200/50	

Table 10.5-13 Assistance with interior finishing works hardware parameterization

ROS stack	package
laser_pipeline	laser_assembler
	laser_filters
	laser_geometry
vision_opencv	cv_bridge
	cv_markers
	image_geometry
perception_pcl	cminpack
	flann
	pcl
	pcl_ros
slam_gmapping	gmapping
object_manipulation	object_manipulation_msgs
	object_manipulator
	point_cloud_server
audio_common	audio_capture
	audio_common_msgs
	audio_play
	sound_play
pr2_navigation	pr2_navigation_teleop
cob_environment_perception	cob_3d_mapping_msgs
pr2_controllers	robot_mechanism_controllers

Table 10.5-14 Assistance with interior finishing workssoftware selection

Raser lawn mowing robot

Component type	quant.	parameters	comments
Ultrasonic Sensor	4	blind range(mm):60	
		maximal operating distance(mm):500	
Binary Sensor	2/2	maximal operating distance(mm):15	
		blind range(mm):0	
Gyroscope/Acceleration Sensor	1	number of axes:6	
		power usage(mA):4.2	
Wheel-based Platform	1	weight(kg):15	
		footprint size(m^2):0.2	
		maximum payload capacity(kg):2	
Miscellaneous End Effectors/ Tools	1	expected cost(€):0	mowing unit (provided externally)
		estimated cost stand. dev.(€):0	
Input Peripherals	1	number of axes:3	
		number of buttons/keys:5	
Output Peripherals	1	pixel array size(inch):1600	2-line dotmatrix
Power Supply (batteries)	0	capacity(Ah):12	(provided externally)
		output voltage(V):12	
Power Supply (DC/DC converter)	1	output voltage(V):5	
		output power(A):15	
Controlling Computer	1	processor type:17	
		volume(mm^3):16000	
		flash memory capacity(GB): 0	
Safety Hardware	1	weight(kg):0.3	bumper
		contact surface size([m^2):0.02	
Miscellaneous Structural Hardware	1/1	expected cost(€):2000/200	grass sensor, compass
		estimated cost stand. dev.(€):500/50	
Miscellaneous Hardware Environment	1/1	expected cost(€):500+500	RFID tags, charging device
		estimated cost stand. dev.(€): 250/100	

Table 10.5-15 Raser lawn mowing hardware parameterization

ROS stack	package
pr2_controllers	robot_mechanism_controllers
joystick_drivers	joy
wifi_drivers	wifi_ddwrt
-common_msgs	-trajectory_msgs
	-actionlib_msgs
-filters	-filters
-nodelet_core	-nodelet_topic_tools
-pluginlib	-pluginlib

Table 10.5-16 Raser lawn mowing software selection

10.6. Scenario Questionnaires

The following questionnaire was used for EFFIROB scenarios in the verification process.

The SEROCOST estimate is … than the EFFIROB estimate.

	much more realistic	more realistic	equally realistic	less realistic	much less realistic	don't know
hardware	○	○	○	○	○	○
software costs	○	○	○	○	○	○

Comment:

How do you assess the hardware cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
material costs	○	○	○	○	○
installation	○	○	○	○	○
administr.	○	○	○	○	○

Comment:

How do you assess the software cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
development costs	○	○	○	○	○
installation	○	○	○	○	○

Comment:

How do you assess the designing cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
designing cost	○	○	○	○	○

Comment:

How do you assess the additional availability of standard deviation and

	very helpful	helpful to some degree	neither helpful nor confusing	confusing to some degree	very confusing	don't know
Standard deviation	○	○	○	○	○	○
IQR	○	○	○	○	○	○

Comment:

Under the aspects of clarity and meaning, which aggregation would you prefer in cost estimates for a service robot?

	for prototype	for product
Including all cost constituents	○	○
Excluding designing cost	○	○
Excluding designing cost, admin. costs	○	○
Direct unit costs	○	○
None of the above	○	○
I don't know.	○	○

Comment:

Combined with other estimation methods, how do you assess the additional support with the SEROCOST approach and tool?

very helpful	helpful to some degree	neither helpful nor confusing	confusing to some degree	very confusing	don't know
○	○	○	○	○	○

Comment:

Given the SEROCOST estimate, how would you adapt the original EFFIROB

	Hardware costs	Software costs
Replace original values with SEROCOST values	○	○
Find a compromise between both	○	○
Keep old values	○	○
I don't know.	○	○

Comment:

If you had to do an early design phase cost estimate for a service robot, would you use the SEROCOST tool and approach?

	Definitely	Probably	Maybe	Probably not	Definitely not	don't know
	○	○	○	○	○	○

Comment: given usability is sufficient

Open question: What do you consider strengths or weaknesses of the SEROCOST approach not covered by the questions above?

The following, slightly modified version of the questionnaire above was used for the Raser scenario verification.

The SEROCOST estimate is … than the Raser estimate.

	much more realistic	more realistic	equally realistic	less realistic	much less realistic	don't know
material costs	○	○	○	○	○	○
labor costs	○	○	○	○	○	○

Comment:

How do you assess the hardware cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
material costs	○	○	○	○	○
installation costs	○	○	○	○	○
administr. costs	○	○	○	○	○

Comment:

How do you assess the software cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
development costs	○	○	○	○	○
installation costs	○	○	○	○	○

Comment:

How do you assess the designing cost estimation?

	very realistic	reasonably realistic	somewhat unrealistic	very unrealistic	don't know
designing cost	○	○	○	○	○

Comment:

How do you assess the additional availability of standard deviation and IQR?

	very helpful	helpful to some degree	neither helpful nor confusing	confusing to some degree	very confusing	don't know
Standard deviation	○	○	○	○	○	○
IQR	○	○	○	○	○	○

Comment:

Under the aspects of clarity and meaning, which aggregation would you prefer in cost estimates for a service robot?

	for prototype cost
Including all cost constituents	○
Excluding designing cost	○
Excluding designing cost, admin. costs	○
Direct unit costs	○
None of the above	○
I don't know.	○

Comment:

Combined with other estimation methods, how do you assess the additional support with the SEROCOST approach and tool?

very helpful	helpful to some degree	neither helpful nor confusing	confusing to some degree	very confusing	don't know
○	○	○	○	○	○

Comment:

Given the SEROCOST estimate, how would you have adapted the original

	Material costs	Labor costs
Replace original values with SEROCOST values	○	○
Find a compromise between both	○	○
Keep old values	○	○
I don't know.	○	○

Comment:

If you had to do an early design phase cost estimate for a service robot, would you use the SEROCOST tool and approach (assuming maximum time effort required for a tool based estimate < 1h)?

Definitely	Probably	Maybe	Probably	Definitely	don't know
○	○	○	○	○	○

Comment:

Open question: What do you consider strengths or weaknesses of the SEROCOST approach not covered by the questions above?

The following questionnaire was used for retrieving expert feedback on the categorization of skills as well as hardware and software component types.

Functions

1. Below you see a categorization of generic skills as requirements a service robot (often) needs to fulfill.
1a. Please verify the usefulness of each category and each function (Does it make sense? Is it important?).
1b. If you think an important category or function is missing please add it. Please make sure that it cannot be subsumed under one of the already listed.
1c. The skills should be as decoupled as possible. If you think that two functions have overlapping meaning please indicate so and, if possible, suggest a better distinction.

Category	**Skill**	**Comments**
Perception	Object Detection	
	Object Recognition	
	Environmental	includes kinematics
Navigation & Locomotion	Self Localization	
	Platform Path	includes collision avoidance
	Platform Control	
Manipulation	Manipulator Path	includes collision avoidance
	Manipulator Control	includes visual servoing, force control
	Grasping & Grasp	
Robot Interface	Human-Robot Interaction	includes speech and gesture recognition, operation and teleoperation
	Programming & Teaching	
	Robot-Robot	
Additional Function	…	

Component Types Hardware

2. Below you see a categorization of generic hardware component types often installed in a service robot.
2a. Please verify the usefulness of each category and each component type (Does it make sense? Is it important?).
2b. If you think an important category or component type is missing please add it. Please make sure that it cannot be subsumed under one of the already listed.

Category	Component	Comments
Perception Hardware	Camera	
	Ultrasonic Sensor	
	Laser scanner	
	Radar	
	Binary Sensor	Point sensor, area sensor (including sensor skin), includes angle transmitters
	Force/Torque-Sensor	
	Gyro-Sensor	
	GPS System	
Navigation & Locomotion	Ground Platform	excludes walkers
	Aerial Platform	UAV Platforms
Manipulation Hardware	Robot Arm	
	Gripper	
	Miscellaneous End Effectors/ Tools	e.g. welding equipment
HRI Hardware (if not listed among sensors)	Input Peripherals	e.g. simple: keyboard, buttons, microphone; advanced: Haptic Input Devices
	Output Peripherals	e.g. simple: loudspeakers, screen; advanced: Augmented Reality Visors

Infrastructure Hardware	Power Supply	
	Controlling Computer	
	Safety Hardware	hardware dedicated to safety issues, i.e. emergency stop circuit
	Miscellaneous Structural Hardware	e.g. frame, wiring harness
Additional	…	

Component Types Software

3. Below you see a categorization of generic software component types often installed in a service robot.

3a. Please verify the usefulness of each category and each component type (Does it make sense? Is it important?).

3b. If you think an important category or component type is missing please add it. Please make sure that it cannot be subsumed under one of the already listed.

Category	Component	Comments
Perception Software	Object Recognition	
	Sensor Fusion	
	Environmental Modeling	
Navigation & Locomotion Software	Self Localization and Mapping	
	Path Planning 2D	
Manipulation Software	Path Planning 3D/6D	
	Visual Servoing	
	Grasping & Grasp Planning	
	Application Specific Control	
HRI Software	Speech Recognition	
	Gesture Recognition	
	Operation & Teleoperation	
	Graphical User Interface	
Systemic Software	(Robot) Operating System	
	Internal Component Interfaces	
	Learning & Reasoning	
Additional Software	…	

Publication bibliography

4cost 2012	4cost (2012): aces - module for parametric cost estimation. Available online at http://www.4cost.de/en/software/parametric-cost-estimating, checked on 2012-09-18.
Acquah 2010	Acquah, Henry de-Graft (2010): Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of an asymmetric price relationship. In *Journal of Development and Agricultural Economics* 2 (1), pp. 1–6.
Albrecht 1979	Albrecht, Allan J. (1979): Measuring Application Development Productivity. In : IBM Application Development, proceedings. IBM Application Development Symposium. GUIDE Int.; SHARE Inc., IBM Corp. Monterey, CA, USA, pp. 83–92.
Albrecht 1983	Albrecht, A.J; Gaffney, J.E (1983): Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 9 (6), pp. 639–648.
Austin 2009	Austin, David; Cole, Luke (2009): Dave's Robotic Operating System. Available online at http://dros.org/, updated on 2009-10-27, checked on 2012-09-19.
Backhaus 1996	Backhaus, Klaus (1996): Multivariate Analysemethoden. Eine anwendungsorientierte Einführung. 8th ed. Berlin, Germany: Springer.
Banker 1991	Banker, R.D; Kauffman, R.J; Kumar, R. (1991): Output measurement metrics in an object-oriented computer aided software engineering (CASE) environment: critique, evaluation and proposal. In: System Sciences, proceedings. 24th Annual Hawaii International Conference on System Sciences. Hawaii, USA, January 8-11. Los Alamitos, CA, USA: IEEE Computer Society, pp. 18–27.

Barclay 2000 — Barclay, I.; Dann, Z. (2000): New-product-development performance evaluation: a product-complexity-based methodology. In *IEE Proc., Sci. Meas. Technol.* 147 (2), p. 41.

Bischoff 2009 — Bischoff, Rainer; Guhl, Tim: Robotic Visions. To 2020 and beyond. The Strategic Research Agenda for Robotics in Europe, 07/009 (2009): European Robotics Technology Platform.

BMBF 2012 — BMBF (2012): BMBF › Forschung › Bekanntmachung. Available online at http://www.bmbf.de/foerderungen/18386.php, updated on 2012-03-14, checked on 2012-09-17.

Boehm 1981 — Boehm, Barry W. (1981): Software Engineering Economics. Englewood Cliffs, NJ, USA: Prentice-Hall.

Boehm 2000 — Boehm, Barry; Abts, Chris; Chulani, Sunita (2000): Software development cost estimation approaches - A survey. In *Annals of Software Engineering* 10 (1/4), pp. 177–205.

Boren 2011 — Boren, Jonathan; Cousins, Steve (2011): Exponential Growth of ROS [ROS Topics]. In *IEEE Robot. Automat. Mag.* 18 (1), pp. 19–20.

Botthof 2011 — Botthof, Alfons; Domröse, Wolfgang; Groß, Wolfram (2011): Technologische und wirtschaftliche Perspektiven Deutschlands durch die Konvergenz der elektronischen Medien. Studie der VDI/VDE Innovation + Technik GmbH. Studienband. VDI/VDE Innovation + Technik GmbH. Berlin, Germany. Available online at http://www.vdivde-it.de/publikationen/studien/technologische-und-wirtschaftliche-perspektiven-deutschlands-durch-die-konvergenz-der-elektronischen-medien-studienband, checked on 2012-09-17.

Braun 2007 — Braun, Stefanie C.; Lindemann, Udo (2007): A Multilayer Approach for early Cost Estimation of mechatronical Products. In : ICED, proceedings. 16th International Conference on Engineering Design. Paris, France, August 28-31. Scotland: The Design Society, pp. 1–10. Available

	online at http://www.pe.mw.tum.de/forschung/publikationen/publikationen/pdfs/BraunLindemann2007a.pdf, checked on 2011-12-28.
Bronštejn 1995	Bronštejn, Ilja N. (1995): Taschenbuch der Mathematik. 2., überarb. und erw. Aufl., erw. Lizenzausg. der bis 1977 erschienenen russ. Orig.-Ausg. Thun, Switzerland: Deutsch.
Brooks 2005	Brooks, A.; Kaupp, T.; Makarenko, A.; Williams, S.; Oreback, A. (2005): Towards component-based robotics. In : IROS, proceedings. IEEE/RSJ International Conference on Intelligent Robots and Systems. Alberta, Canada, August 2-6. Piscataway, NJ, USA: IEEE, pp. 163–168.
Browning 2001	Browning, T.R (2001): Applying the design structure matrix to system decomposition and integration problems: a review and new directions. In *IEEE Trans. Eng. Managemt. (IEEE Transactions on Engineering Management)* 48 (3), pp. 292–306.
Browning 2002	Browning, T.R; Eppinger, S.D (2002): Modeling impacts of process architecture on cost and schedule risk in product development. In *IEEE Trans. Eng. Managemt. (IEEE Transactions on Engineering Management)* 49 (4), pp. 428–442.
Brugali 2007	Brugali, Davide; Brooks, Alex; Cowley, Anthony; Côté, Carle; Domínguez-Brito, Antonio C.; Létourneau, Dominic et al. (2007): Trends in Component-Based Robotics. In Davide Brugali (Ed.): Software engineering for experimental robotics, vol. 30. Berlin, Germany: Springer (30), pp. 135–142.
Brugali 2009	Brugali, Davide; Scandurra, Patrizia (2009): Component-based robotic engineering (Part I) [Tutorial]. In *IEEE Robot. Automat. Mag.* 16 (4), pp. 84–96.
CARMEN-Team 2009	CARMEN-Team (2009): CARMEN. Available online at http://carmen.sourceforge.net/home.html, updated on 2009-12-02, checked on 2012-09-19.

Casella 2002 — Casella, George; Berger, Roger L. (2002): Statistical inference. 2nd ed. Pacific Grove, CA, USA: Thomson Learning.

CSE 2000 — Center for Software Engineering, USC (2000): COCOMO II. Model Definition Manual. Version 2.1. Available online at ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/modelman.pdf, checked on 2012-09-17.

Chen 2009 — Chen, Qi-Wei; Li, Guo-Yin; Zhuang, Qing-Hui (2009): The Analysis of Project Schedule Uncertainty: Based on Monte Carlo Simulation. In : MASS, proceedings. International Conference on Management and Service Science. Wuhan, China, September 20-22. Piscataway, NJ, USA: IEEE, pp. 2217–2220.

Chidamber 1994 — Chidamber, S.R; Kemerer, C.F (1994): A metrics suite for object oriented design. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 20 (6), pp. 476–493.

Clausing 1994 — Clausing, Don (1994): Total quality development. A step-by-step guide to world class concurrent engineering. New York, NY, USA: ASME Press.

Coenenberg 2009 — Coenenberg, Adolf Gerhard; Fischer, Thomas M.; Günther, Thomas (2009): Kostenrechnung und Kostenanalyse. 7th ed. Stuttgart, Germany: Schäffer-Poeschel.

COMM/ESTAT — COMM/ESTAT: Eurostat Home. eurostat. Available online at http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home, checked on 2012-09-20.

CBET 1990 — Committee on Budget Estimating Techniques, Building Research Board National Research Council (1990): Improving the accuracy of early cost estimates for federal construction projects. Washington, D.C., USA: National Academy Press. Available online at http://www.nap.edu/catalog.php?record_id=1693, checked on 2012-10-24.

CSB 2012	Construx Software Builders (2012): Construx Estimate - Construx. Available online at http://www.construx.com/Page.aspx?nid=68, checked on 2012-09-18.
Cook 1982	Cook, Dennis; Weisberg, Sanford (1982): Criticism and Influence Analysis in Regression. In *Sociological Methodology* 13, pp. 313–361.
CSSE 2002	CSSE (2002): Parametric Cost Estimating Handbook - Estimation Methodologies. Cost Estimation And Assessments Office, Johnson Space Center, National Aeronautics and Space Administration. Available online at http://cost.jsc.nasa.gov/pcehhtml/pceh.htm, updated on 2004-03-24, checked on 2012-09-17.
CSSE 2011	CSSE (2011): CSSE Website. COCOMO(TM) II. Center for Software Engineering, University of Southern California. Available online at http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html, updated on 2011-03-22, checked on 2012-09-18.
Dalgaard 2010	Dalgaard, Lars (2010): Rational System-level Design Methodology for Autonomous Robotic Systems. Ph. D. thesis. University of Southern Denmark, Odense, Denmark. The Maersk-McKinney Moller Institute. Available online at http://www.teknologisk.dk/_root/media/47265_thesis_larsdalgaard.pdf.
Danial 2012	Danial, Al (2012): CLOC -- Count Lines of Code. Northrop-Grumman Corporation. Available online at http://cloc.sourceforge.net/, updated on 2012-04-10, checked on 2012-09-20.
Dewdney 1998	Dewdney, A. K. (1998): Yes, we have no neutrons. An eye-opening tour through the twists and turns of bad science. New York, NY, USA: Wiley.
Diankov 2012	Diankov, Rosen (2012): OpenRAVE | Home. Available online at http://openrave.org/, checked on 2012-09-19.

Dörner 2003	Dörner, Dietrich (2003): Die Logik des Misslingens. Strategisches Denken in komplexen Situationen. Erw. Neuausg. Reinbek bei Hamburg, Germany: Rowohlt.
Ehrlenspiel 2007a	Ehrlenspiel, Klaus; Kiewert, Alfons; Lindemann, Udo; Hundal, Mahendra S. (2007): Cost-efficient design. Berlin, Germany: Springer; ASME Press.
Ehrlenspiel 2007b	Ehrlenspiel, Klaus; Kiewert, Alfons; Lindemann, Udo (2007): Kostengünstig entwickeln und konstruieren. Kostenmanagement bei der integrierten Produktentwicklung ; mit 143 Tabellen. 6th ed. Berlin, Germany: Springer.
Eversheim 1985	Eversheim, W.; Rothenbücher, J. (1985): Kalkulation von Vorrichtungen in der Konzeptphase. Theory and Practice of Engineering Design in International Comparison. In V. Hubka (Ed.): ICED, proceedings. Theory and practice of engineering design in international comparison, vol. 1. International Conference on Engineering Design. Hamburg, Germany, August 26-28. Zurich, Switzerland: Edition Heurista, pp. 427–436.
Fahrmeir 2009	Fahrmeir, Ludwig; Kneib, Thomas; Lang, Stefan (2009): Regression. Modelle, Methoden und Anwendungen. 2nd ed. Berlin, Germany: Springer.
FDT 2009	Fawkes Development Team (2009): Fawkes | Robot Software Framework. Edited by Fawkes Development Team. Available online at http://www.fawkesrobotics.org/, checked on 2012-09-19.
Fink 2005	Fink, Miriam (2005): Metrikeinsatz in Software-Projekten. Zugel. Diplomarbeit. Universität Stuttgart, Stuttgart, Germany. Fakultät Elektrotechnik, Informatik, Informationstechnik.
Fitzpatrick 2012	Fitzpatrick, Paul; Natale, Lorenzo; Metta, Giorgio (2012): YARP: Welcome to YARP. Available online at http://eris.liralab.it/yarpdoc/index.html, updated on 2012-09-19, checked on 2012-09-19.

Flick 2007	Flick, Uwe; Kardorff, Ernst von; Steinke, Ines (2007): Qualitative Forschung. Ein Handbuch. 5th ed. Reinbek bei Hamburg, Germany: Rowohlt.
Flyvberg 2008	Flyvberg, Bent (2008): Curbing Optimism Bias and Strategic Misrepresentation in Planning:Reference Class Forecasting in Practice. In *European Planning Studies* 16 (1), pp. 3–21.
Fox 2000	Fox, John (2000): Nonparametric simple regression. Smoothing scatterplots. Thousand Oaks, CA, USA: Sage.
Fox 2005	Fox, John (2005): Introduction to Nonparametric Regression. McMaster University. Hamilton, Ontario, Canada, 2005. Available online at http://civil.colorado.edu/~balajir/CVEN6833/lectures/nonparametric-regression-slides.pdf.
Frakes 1996	Frakes, William; Terry, Carol (1996): Software Resue: Metrics and Models. In *ACM Computing Surveys* 28 (2), pp. 415–435.
Fraunhofer IPA 2011	Fraunhofer IPA (2011): Robot Systems. Available online at http://www.ipa.fraunhofer.de/Robot_Systems.17.0.html?&L=2, checked on 2012-09-20.
Galway	Subjective Probability Distribution Elicitation in Cost Risk Analysis. Galway, Lionel A. (2007): Subjective Probability Distribution Elicitation in Cost Risk Analysis. A Review. RAND Corporation. Santa Monica, CA, USA. Available online at http://www.rand.org/pubs/technical_reports/TR410.html.
Gerpott 1999	Gerpott, Torsten J. (1999): Strategisches Technologie- und Innovationsmanagement. Eine konzentrierte Einführung. Stuttgart, Germany: Schäffer-Poeschel.
Glinz 2004	Glinz, Martin (2004): 5. Software-Aufwandschätzung. Software Engineering I. Universität Zürich. Zurich, Switzerland, 2004. Available online at https://files.ifi.uzh.ch/rerg/arvo/ftp/se_I/kapitel_05.pdf, checked on 2011-12-27.

Golub 1979 — Golub, Gene H.; Heath, Micheal; Wahba, Grace (1979): Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter. In *Technometrics* 21 (2), pp. 215–223.

Gordon 1961 — Gordon, William J. J. (1961): Synectics. The Development of Creative Capacity. New York, NY, USA: Harper Row.

Gostai 2012 — Gostai (2012): UrbiForge Main/Home Page. Available online at http://www.urbiforge.org/, updated on 2012-11-15, checked on 2013-01-25.

Gurney 1997 — Gurney, Kevin (1997): An introduction to neural networks. London, UK: UCL Press.

Hägele 2010 — Hägele, Martin; Blümlein, Nikolaus; Kleine, Oliver (2010): Wirtschaftlichkeitsanalysen neuartiger Servicerobotik (EFFIROB). Anwendungen und ihre Bedeutung für die Robotik-Entwicklung. Eine Analyse der Fraunhofer-Institute IPA und ISI im Auftrag des BMBF. Fraunhofer-Gesellschaft. Munich, Germany. Available online at http://www.ipa.fraunhofer.de/index.php?id=1643, checked on 2012-09-17.

Hägele 2011 — Hägele, Martin (2011): World robotics 2011. Service robots. Frankfurt am Main, Germany: VDMA.

Han 2011 — Han, Long; Wu, Xinyu; Liu, Guangyuan; Chen, Chunjie; Ou, Yongsheng; Xu, Yangsheng (2011): An efficient and low-cost robot grasping system in household environments. In : WCICA, proceedings. 9th World Congress on Intelligent Control and Automation. Taipeh, Taiwan, June 21-25. Piscataway, NJ, USA: IEEE, pp. 593–598.

Hender 2001 — Hender, J.M; Rodgers, T.L; Dean, D.L; Nunamaker, J.F (2001): Improving group creativity: brainstorming versus non-brainstorming techniques in a GSS environment. In : System Sciences, proceedings. 34th Annual Hawaii International Conference on System Sciences. Maui, HI, USA, January 3-6. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1–10.

Hertz 1991	Hertz, John; Krogh, Anders; Palmer, Richard G. (1991): Introduction to the theory of neural computation. Redwood City, CA, USA: Addison-Wesley.
Hughes 1996	Hughes, R. T. (1996): Expert Judgment as an Estimation Method. In *Information and Software Technology* (38), pp. 67–75.
ISO 8373	ISO 8373, 2012-03-01: Robots and robotic devices - Vocabulary.
Jaybridge Robotics 2012	Jaybridge Robotics (2012): Jaybridge Robotics - Products - RI-JAUS. Available online at http://www.jaybridge.com/products/ri-jaus/, checked on 2012-09-19.
JPL 2008	Jet Propulsion Laboratory (2008): CLARAty. Available online at https://claraty.jpl.nasa.gov/man/overview/index.php, updated on 2008-09-10, checked on 2012-09-19.
Jones 1995	Jones, Capers (1995): Backfiring: converting lines of code to function points. In *Computer* 28 (11), pp. 87–88.
Jones 2007	Jones, Capers (2007): Estimating software costs. Bringing realism to estimating. 2nd ed. New York, NY, USA: McGraw-Hill. Available online at http://www.loc.gov/catdir/enhancements/fy0712/2007011911-d.html.
Jorgensen 2007	Jorgensen, Magne; Shepperd, Martin (2007): A Systematic Review of Software Development Cost Estimation Studies. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 33 (1), pp. 33–53.
Kalb 1990	Kalb, George E. (1990): Counting Lines of Code, Confusions, Conclusions, and Recommendations. Briefing to the 3rd Annual REVIC User's Group Conference, 1990-10-01. Available online at http://sunset.usc.edu/research/CODECOUNT/documents/3rd_REVIC.pdf, checked on 2012-09-18.
Kang 2005	Kang, Kyo Chul; Kim, Moonzoo; Lee, Jaejoon; Kim, Byungkil; Hong, Youngjin; Lee, Hyoungki; Bang, Seokwon (2005): 3D Virtual Prototyping of Home Service Robots

Using ASADAL/OBJ. In : ICRA, proceedings. IEEE International Conference on Robotics and Automation. Barcelona, Spain, April 18-22. IEEE. Piscataway, NJ, USA: IEEE, pp. 2903–2908.

Keefer 1993 — Keefer, Donald L.; Verdini, William A. (1993): Better Estimation of PERT Activity Time Parameters. In *Management Science* 39 (9), pp. 1086–1091.

Keeney 1989 — Keeney, Ralph L.; Winterfeldt, Detlof von (1989): On the uses of expert judgment on complex technical problems. In *IEEE Trans. Eng. Managemt. (IEEE Transactions on Engineering Management)* 36 (2), pp. 83–86.

Keung 2008 — Keung, J.W; Kitchenham, B.A; Jeffery, D.R (2008): Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 34 (4), pp. 471–484.

Kim 2005 — Kim, Moonzoo; Lee, Jaejoon; Hong, Youngjin; Bang, Seokwon; Kang, Kyo Chul (2005): Re-engineering software architecture of home service robots: a case study. In : ICSE, proceedings. 27th International Conference on Software Engineering. St. Louis, MO, USA, May 15-21. New York, NY, USA: The Association for Computing Machinery, pp. 505–513.

Kim 2009 — Kim, Minseong; Kim, Suntae; Park, Sooyong; Choi, Mun-Taek; Kim, Munsang; Gomaa, Hassan (2009): Service robot for the elderly. In *IEEE Robot. Automat. Mag.* 16 (1), pp. 34–45.

Kossiakoff 2003 — Kossiakoff, Alexander; Sweet, William N. (2003): Systems engineering. Principles and practice. Hoboken, NJ, USA: Wiley-Interscience.

Kulak 2010 — Kulak, Osman; Cebi, Selcuk; Kahraman, Cengiz (2010): Applications of axiomatic design principles: A literature review. In *Expert Systems with Applications* 37 (9), pp. 6705–6717. Available online at http://www.sciencedirect.com/science/article/pii/S0957417410002423.

Lawrence 1994 — Lawrence, Jeannette; Luedeking, Sylvia (1994): Introduction to neural networks. Design, theory and applications. 6th ed. Nevada City, CA, USA: California Scientific Software.

Leung 2002 — Leung, Hareton; Zhang, Fan (2002): Software Cost Estimation. In S. K. Chang: Handbook of software engineering & knowledge engineering, vol. 2. River Edge, NJ, USA: World Scientific, pp. 307–324.

Levin 2012 — Levin, Mark Sh (2012): Morphological methods for design of modular systems (a survey). Cornell University Library. Available online at http://arxiv.org/abs/1201.1712v1, updated on 2012-01-09, checked on 2012-09-18.

Li 2006 — Li, Dengke; Zhang, Henxi; Li, Shouan (2006): Development Cost Estimation of Aircraft Frame based on BP Neural Networks. In *Fire Control and Command Control* (09), pp. 699–701.

Lindemann 2005 — Lindemann, Udo (2005): Methodische Entwicklung technischer Produkte. Methoden flexibel und situationsgerecht anwenden. Berlin, Germany: Springer.

Linstone 1975 — Linstone, Harold A.; Turoff, Murray (1975): The Delphi method. Techniques and applications. Reading, MA, USA: Addison-Wesley.

Liu 2003 — Liu, Guoli; Tang, Xiaobing; Liu, Yuan-liang (2003): Prediction for the Missile Development Cost Based on Neural Network. In *Tactical Missile Technology* (1), pp. 23–26.

Lopez 2011 — Lopez, Joaquin; Perez, Diego; Zalama, Eduardo; Gomez-Garcia-Bermejo, Jaime (2011): Low cost indoor mobile robot localization system. In : ISDA, proceedings. 11th International Conference on Intelligent Systems Design and Application. Córdoba, Spain, November 22-24. Piscataway, NJ, USA: IEEE, pp. 1134–1139.

Ma 2010 — Ma, Junhai; Mu, Lingling (2010): Comparison Study on Methods of Software Cost Estimation. In V. E. Muchin, Zhiwei Ye (Eds.): EBISS, proceedings. 2nd International

Conference E-Business and Information System Security. Wuhan, China, May 22-23. Piscataway, NJ, USA: IEEE, pp. 1–4.

Mair 2005 — Mair, C.; Shepperd, M. (2005): The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In : ISESE, proceedings. International Symposium on Empirical Software Engineering. Noosa Heads, Queensland, Australia, November 17-18. Los Alamitos, CA, USA: IEEE Computer Society, pp. 509–518.

Masing 2007 — Masing, Walter; Pfeifer, Tilo (2007): Handbuch Qualitätsmanagement. 5th ed. Munich, Germany: Hanser.

Maurer 2006 — Maurer, Maik; Pulm, Udo; Ballestrem, Felix (2006): The Subjective Aspects of Design Structure Matrices. Analysis of Comprehension and Application and Means to Overcome Differences. In : Engineering Systems Design and Analysis, proceedings. 8th Biennial Conference on Engineering Systems Design and Analysis. Torino, Italy, July 4-7. ASME. New York, NY, USA: American Society of Mechanical Engineers.

Maurer 2007 — Maurer, Maik S. (2007): Structural Awareness in Complex Product Design. Doctoral thesis. Technische Universität München, Munich, Germany. Faculty of Mechanical Engineering.

Maurer 2008 — Maurer, Maik; Lindemann, Udo (2008): The application of the Multiple-Domain Matrix: Considering multiple domains and dependency types in complex product design. In : SMC, proceedings. International Convention and Exhibition. International Conference on Systems, Man and Cybernetics. Singapore, October 12-15. IEEE. Piscataway, NJ, USA: IEEE, pp. 2487–2493.

McConnell 2006 — McConnell, Steve (2006): Software estimation. Demystifying the black art. Redmond, Wash: Microsoft Press.

Meisl 1988 — Meisl, Claus J. (1988): Techniques for cost estimating in early program phases. In *Engineering Costs and Production Economics* 14 (2), pp. 95–106.

Minkiewicz 1997 — Minkiewicz, Arlene F. (1997): Measuring Object Oriented Software with Predictive Object Points. In : ASM, proceedings. Applications in Software Management - Workshops on Abstract State Machines. Atlanta, GA, USA. Available online at http://www.pricesystems.com/white_papers/Measuring%20Object%20Oriented%20Software%20with%20Predictive%20Object%20Points%20July%20%2797%20-%20Minkiewicz.pdf, checked on 2013-02-20.

Molokken 2003 — Molokken, Kjétil; Jorgensen, Magne (2003): A review of software surveys on software effort estimation. In : ISESE, proceedings. International Symposium on Empirical Software Engineering. Rome, Italy, September 30 - October 1. IEEE Computer Society. Los Alamitos, CA, USA: IEEE Computer Society, pp. 223–230.

MPRT 2012 — MPRT (2012): The Mobile Robot Programming Toolkit. The Mobile Robot Programming Toolkit (MPRT) initiative. Available online at http://www.mrpt.org/, checked on 2012-09-19.

Mundhenk 2003 — Mundhenk, T. Nathan; Ackerman, Christopher; CHung, Daesu; Dhavale, Nitin; Hudson, Brian; Hirata, Reid et al. (2003): Low cost, high performance robot design utilizing off-the-shelf parts and the Beowulf concept, The Beobot project. In : Intelligent Robots and Computer Vision, proceedings. SPIE Conference on Intelligent Robots and Computer Vision, October. SPIE International Society for Optics and Photonics. Bellingham, WA, USA: SPIE Press, pp. 293–303.

Musilek 2002 — Musilek, P.; Pedrycz, W.; Nan Sun; Succi, G. (2002): On the sensitivity of COCOMO II software cost estimation model. In : METRICS, proceedings. Eighth IEEE Symposium on Software Metrics. Ottawa, Canada, June 4-7. Los Alamitos, CA, USA: IEEE Computer Society, pp. 13–20.

Neter 1989 — Neter, John; Wasserman, William; Kutner, Michael H. (1989): Applied linear regression models. 2nd ed. Homewood, IL, USA: Irwin.

Nguyen 2007 — Nguyen, Vu; Deeds-Rubin, Sophia; Tan, Thomas; Boehm, Barry W. (2007): A SLOC Counting Standard. Center for Systems and Software Engineering, University of Southern California. Available online at http://csse.usc.edu/csse/TECHRPTS/2007/usc-csse-2007-737/usc-csse-2007-737.pdf, checked on 2012-09-18.

Niemueller 2010 — Niemueller, Tim; Ferrein, Alexander; Beck, Daniel; Lakemeyer, Gerhard (2010): Design Principles of the Component-Based Robot Software Framework Fawkes. In Noriaki Ando, Stephen Balakirsky, Thomas Hemker, Monica Reggiani, Oskar von Stryk (Eds.): Simulation, Modeling and Programming for Autonomous Robots, proceedings. Second International Conference. (in: Lecture Notes on Computer Science). Second International Conference on Simulation, Modeling and Programming for Autonomous Robots. Darmstadt, Germany, November 15-18. Heidelberg, Germany: Springer, pp. 300–311.

Nonaka 1995 — Nonaka, Ikujirō; Takeuchi, Hirotaka (1995): The knowledge-creating company. How Japanese companies create the dynamics of innovation. New York, NY, USA: Oxford University Press.

NRW 2002 — NRW, Statistisches Landesamt (2002): Research data centres of the Federal Statistical Office and the statistical offices of the Länder. Statistical Offices of the Federation and the Länder. Available online at http://www.forschungsdatenzentrum.de/en/index.asp, updated on 2002-12-05, checked on 2012-09-20.

O'Brien 2007 — O'Brien, Robert M. (2007): A Caution Regarding Rules of THumb for Variance Inflation Factors. In *Quality & Quantity* (41), pp. 673–690.

OpenJAUS 2012 — OpenJAUS (2012): OpenJAUS - JAUS Robotics Software Development Kit (SDK). Available online at http://www.openjaus.com/, checked on 2012-09-19.

OpenMORA 2012	OpenMORA (2012). Available online at http://sourceforge.net/p/openmora/home/Home/, checked on 2012-09-19.
OPRoS WIKI 2011	OPRoS WIKI (2011). Available online at http://210.115.36.127/doku.php, updated on 2011-06-27, checked on 2012-09-19.
Orca Robotics 2009	Orca Robotics (2009): Orca: Components for Robotics. Available online at http://orca-robotics.sourceforge.net/, updated on 2009-11-18, checked on 2012-09-19.
Oriogun 1999	Oriogun, Peter K. (1999): A Survey of Boehm's Work on the Spiral Models and COCOMO II - Towards Software Development Process Quality Improvement. In *Software Quality Journal* (8), pp. 53–62.
Osborn 1963	Osborn, A. F. (1963): Applied Imagination. Principles and procedures of creative problem solving. Third Revised Edition: Scribner.
OMRG 2008	Oxford Mobile Robotics Group (2008): MOOS : Main - Home Page browse. Available online at http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php, updated on 2012-12-16, checked on 2013-01-25.
Piperidis 2007	Piperidis, S.; Doitsidis, L.; Anastasopoulos, C.; Tsourveloudis, N. C. (2007): A low cost modular robot vehicle design for research and education. In : MED, proceedings. Conference on Control & Automation. Athens, Greece, June 27-29. IEEE. Piscataway, NJ, USA: IEEE, pp. 1–6.
Plinke 2000	Plinke, Wulff; Rese, Mario (2000): Industrielle Kostenrechnung. Eine Einführung. 5th ed. Berlin, Germany: Springer.
Polanyi 1985	Polanyi, Michael (1985): Implizites Wissen. 1st ed. Frankfurt am Main, Germany: Suhrkamp.
Porter 1996	Porter, Micheal E. (1996): What is strategy? In *Harvard Business Review* (November-December), pp. 61–78.
Porter 2000	Porter, Michael Eugene (2000): Wettbewerbsvorteile (competitive advantage). Spitzenleistungen erreichen

	und behaupten. 6th ed. Frankfurt am Main, Germany: Campus.
Porter 2004	Porter, Michael E. (2004): Competitive strategy. First Free Press Export Edition. New York, NY, USA: Free Press.
Prassler 2010	Prassler, Erwin (2010): Servicerobotik und die Entdeckung ihrer Langsamkeit. In *Economic Engineering* (5). Available online at http://www.economic-engineering.de/de/knowledge-corner/12-business-development.html, checked on 2012-09-17.
Psarros 2009	Psarros, D.; Papadimitriou, V.; Chatzakos, P.; Spais, V.; Hrissagis, K. (2009): FRC: A low-cost service robot for subsea flexible risers. In : ICAR, proceedings. 14th International Conference on Advanced Robotics. Munich, Germany, June 22-26. Piscataway, NJ, USA: IEEE, pp. 1–6.
Putnam 1978	Putnam, L.H (1978): A General Empirical Solution to the Macro Software Sizing and Estimating Problem. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 4 (4), pp. 345–361.
RAND Corp. 2010	RAND Corp. (2010): Delphi | RAND. Available online at http://www.rand.org/international_programs/pardee/pubs/futures_method/delphi.html, updated on 2010-03-01, checked on 2012-09-18.
Reid 2002	Reid, R. Dan; Sanders, Nada R. (2002): Operations management. New York, NY, USA: Wiley.
Ritchey 1998	Ritchey, Tom (1998): General Morphological Analysis. A general method for non-quantified modelling. In : Operational Analysis, proceedings. 16th EURO Conference on Operational Analysis. Brussels, Belgium, July, pp. 1–11. Available online at http://www.swemorph.com/pdf/gma.pdf, checked on 2013-02-20.
Rohrbach 1969	Rohrbach, Bernd (1969): Kreativ nach Regeln - Methode 635, eine neue Technik zum Lösen von Problemen. In *Absatzwirtschaft* 12, pp. 73–75.
Rowe 1999	Rowe, Gene; Wright, George (1999): The Delphi Technique as a Forecasting Tool: Issues and Analysis. In

International Journal of Forecasting 15 (4), pp. 353–375. Available online at http://www.sciencedirect.com/science/article/pii/S0169207099000187.

Rowe 2002 — Rowe, Gene; Wright, George (2002): Expert Opinions in Forecasting: The Role of the Delphi Technique. In Jon Scott Armstrong (Ed.): Principles of forecasting. A handbook for researchers and practitioners. 2nd ed. Boston, MA, USA: Kluwer academic, pp. 125–144.

Roy 2003 — Roy, Rajkumar (2003): Cost Engineering: Why, What and How? Cranfield, UK: Cranfield University.

Rush 2001 — Rush, Christopher; Roy, Rajkumar (2001): Expert Judgement in Cost Estimating: Modelling the Reasoning Process. In *Concurrent Engineering* 9 (4), pp. 271–284.

Sachs 2002 — Sachs, Lothar (2002): Angewandte Statistik. Anwendung statistischer Methoden ; mit 317 Tabellen und 99 Übersichten. 10th ed. Berlin, Germany: Springer.

Santillo 2005 — Santillo, Luca; Conte, Massimiliano; Meli, Roberto (2005): Early & Quick Function Point: Sizing more with less. In : Software Metrics, proceedings. 11th International Symposium on Software Metrics. Como, Italy, September 19-22. IEEE Computer Society. Los Alamitos, CA, USA: IEEE Computer Society, pp. 41–46.

Schehl 1994 — Schehl, Michael (1994): Die Kostenrechnung der Industrieunternehmen vor dem Hintergrund unternehmensexterner und -interner Strukturwandlungen. Eine theoretische und empirische Untersuchung. Berlin, Germany: Duncker & Humblot.

Schnell 2008 — Schnell, Rainer; Hill, Paul Bernhard; Esser, Elke (2008): Methoden der empirischen Sozialforschung. 8th ed. Munich, Germany: Oldenbourg.

Schönberg 2012 — Schönberg, Johannes Paul (2012): Hardware-orientierte Kostenabschätzung prototypischer Serviceroboterkonzepte mittels nichtparametrischer Regressionsanalyse. Master thesis. University of Applied

Sciences Technikum Wien, Vienna, Austria. Department of Mechatronics.

Shepperd 1996 — Shepperd, M.; Schofield, C.; Kitchenham, B. (1996): Effort estimation using analogy. In : Software Engineering, proceedings. 18th International Conference on Software Engineering. Berlin, Germany, March 25-29. Los Alamitos, CA, USA: IEEE Computer Society, pp. 170–178.

Shepperd 1997 — Shepperd, M.; Schofield, C. (1997): Estimating software project effort using analogies. In *IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering)* 23 (11), pp. 736–743.

Shepperd 2002 — Shepperd, Martin (2002): ANGEL Project. Empirical Software Engineering Research Group, Bournemouth University. Available online at http://dec.bmth.ac.uk/ESERG/ANGEL/, checked on 2012-09-18.

Shue 2012 — Shue, Sam; Hargrove, Claude; Conrad, James (2012): Low cost semi-autonomous sentry robot. In : Southeastcon, proceedings. Southeastcon. Orlando, FL, USA, March 15-18. IEEE. Piscataway, NJ, USA: IEEE, pp. 1–5.

Siciliano 2008 — Siciliano, Bruno; Khatib, Oussama (2008): Springer handbook of robotics. Berlin, Germany: Springer.

Smits 2010 — Smits, Ruben (2010): Robot Skills. Design of a constraint-based methodology and software support. Doctoral thesis. Katholieke Universiteit Leuven, Leuven, Belgium. Faculty of Engineering.

Sneed 2010 — Sneed, Harry M.; Seidl, Richard; Baumgartner, Manfred (2010): Software in Zahlen. Die Vermessung von Applikationen. Munich, Germany: Hanser.

SPR 2007 — Software Productivity Research (2007): SPR Programming Languages Table. Version PLT2007d.

SPR 2012 — Software Productivity Research (2012): Programming Languages Table. Available online at http://www.spr.com/programming-languages-table.html, checked on 2012-09-20.

statista 2012 — statista (2012): Statista - das Statistik-Portal: Statistiken, Marktdaten & Studien. Available online at http://de.statista.com/, updated on 2012-09-14, checked on 2012-09-20.

Statistisches Bundesamt 2012a — Statistisches Bundesamt (2012): Gesamtwirtschaft & Umwelt - Tarifindex - Entwicklung der tariflichen Monatsverdienste in Deutschland und in Frankreich - Statistisches Bundesamt (Destatis). Available online at https://www.destatis.de/DE/ZahlenFakten/GesamtwirtschaftUmwelt/VerdiensteArbeitskosten/Tarifverdienste/Tarifindex/Tabellen/MonatsverdienstDF.html, updated on 2012-06-25, checked on 2012-09-20.

Statistisches Bundesamt 2012b — Statistisches Bundesamt (2012): Startseite - Statistisches Bundesamt (Destatis). Available online at https://www.destatis.de/DE/Startseite.html, checked on 2012-09-20.

Stellman 2006 — Stellman, Andrew; Greene, Jennifer (2006): Applied Software Project Management. Sebastopol, CA, USA: O'Reilly.

Stensrud 1998 — Stensrud, Erik (1998): Estimating with Enhanced Object Points vs. Function Points. In : 13th COCOMO/SCM Forum, proceedings. 13th COCOMO/SCM Forum. Los Angeles, CA, USA, October: University of Southern California, pp. 1–5.

Stochel 2011 — Stochel, Marek Grzegorz (2011): Reliability and Accuracy of the Estimation Process - Wideband Delphi vs. Wisdom of Crowds. In : COMPSAC, proceedings. 35th Annual IEEE International Computer Software and Applications Conference. Munich, Germany, July 18-21. IEEE, IEEE computer society. Los Alamitos, CA, USA (Vol. 1), pp. 350–359.

Stösser 1999 — Stösser, Robert (1999): Zielkostenmanagement in integrierten Produkterstellungsprozessen. Als Ms. gedr. Aachen, Germany: Shaker.

Suh 2001	Suh, Nam P. (2001): Axiomatic design. Advances and applications. New York, NY, USA: Oxford University Press.
Sun 2007	Sun, Yi-Ran; Zhao, Song-Zheng; Liu, Wei; Xu, Heng (2007): Research on a Manufacturing Cost Estimating Method Based on ABC for Aeronautic Product. In : Wireless Communications, Networking and Mobile Computing, proceedings. International Conference on Wireless Communications, Networking and Mobile Computing. Shanghai, China, September 21-25. IEEE. Piscataway, NJ, USA: IEEE, pp. 4059–4062.
Surowiecki 2005	Surowiecki, James (2005): The wisdom of crowds. Why the many are smarter than the few. London, UK: Abacus.
Tani 2009	Tani, G.; Cimatti, B. (2009): Technological complexity: A support to management decisions for product engineering and manufacturing. In : IEEM, proceedings. The IEEE International Conference on Industrial Engineering and Engineering Management. Singapore, December 8-11, 2008. IEEE. Piscataway, NJ, USA: IEEE, pp. 6–11.
The Orocos Project	The Orocos Project: Roadmap ideas for 3.x | The Orocos Project. Available online at http://www.orocos.org/wiki/orocos/development/roadmap-ideas-3x, checked on 2012-09-19.
The Orocos Project 2012	The Orocos Project (2012): The Orocos Project | Smarter control in robotics & automation. Available online at http://www.orocos.org/, updated on 2012-12-03, checked on 2013-01-25.
The Player Project	The Player Project: Player Project. Available online at http://playerstage.sourceforge.net/, checked on 2012-09-19.
RDCT 2006	The R Development Core Team (2006): The R Project for Statistical Computing. R Foundation for Statistical Computing. Available online at http://www.r-project.org/, updated on 2006-04-21, checked on 2012-09-20.

RDCT 2012	The R Development Core Team (2012): R: A Language and Environment for Statistical Computing. Reference Index. Version 2.15.1. R Foundation for Statistical Computing. Available online at http://cran.r-project.org/doc/manuals/fullrefman.pdf, checked on 2012-09-19.
Touesnard 2004	Touesnard, Brad: Software Cost Estimation. SLOC-based Models and Function Points Model. Version 1.1 (2004), pp. 1–10. Available online at http://bradt.ca/docs/SWE4103-report.pdf, checked on 2012.09-18.
Trivailo 2012	Trivailo, O.; Sippel, M.; Şekercioğlu, Y. A. (2012): Review of hardware cost estimation methods, models and tools applied to early phases of space mission planning. In *Progress in Aerospace Sciences* 53 (0), pp. 1–17. Available online at http://www.sciencedirect.com/science/article/pii/S0376042112000140.
Ulrich 2012	Ulrich, Karl T.; Eppinger, Steven D. (2012): Product design and development. 5th ed. New York, NY, USA: McGraw-Hill/Irwin.
VDMA 2009	Verband Deutscher Maschinen- und Anlagenbau (2009): VDMA Kennzahlenkompass. Informationen für Unternehmer und Führungskräfte. Frankfurt am Main, Germany: VDMA.
VDI-Richtlinie 2221	VDI-Richtlinie 2221, May 1993: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte.
VDI-Richtlinie 2225 Blatt 4	VDI-Richtlinie 2225 Blatt 4, November 1997: Konstruktionsmethodik - Technisch-wirtschaftliches Konstruieren - Bemessungslehre.
VDI-Richtlinie 2225 Blatt 1	VDI-Richtlinie 2225 Blatt 1, November 1997: Konstruktionsmethodik - Technisch-wirtschaftliches Konstruieren - Vereinfachte Kostenermittlung.
Waibel 2011	Waibel, Markus; Beetz, Michael; Civera, Javier; D'Andrea, Raffaello; Elfring, Jos; Gálvez-López, Dorian et al. (2011):

RoboEarth. In *IEEE Robot. Automat. Mag.* 18 (2), pp. 69–82.

Wiechers and Schneider 2012 — Wiechers, Ralph; Schneider, Gesine (Eds.) (2012): VDMA Mechanical engineering - figures and charts. VDMA. Frankfurt am Main, Germany.

Wierda 1988 — Wierda, Leo S. (1988): Product Cost-Estimation by the Designer. In *Engineering Costs and Production Economics* (13), pp. 189–198.

Wikipedia 2003 — Wikipedia (Ed.) (2003): Source lines of code - Wikipedia, the free encyclopedia. Available online at http://en.wikipedia.org/w/index.php?oldid=503106479, updated on 2012-09-13, checked on 2012-09-18.

Wikipedia 2006 — Wikipedia (Ed.) (2006): Function point - Wikipedia, the free encyclopedia. Available online at http://en.wikipedia.org/wiki/Function_points, updated on 2012-07-03, checked on 2012-09-18.

Wikipedia 2009 — Wikipedia (Ed.) (2009): Open-source robotics. Available online at http://en.wikipedia.org/wiki/Open-source_robotics, updated on 2013-01-08, checked on 2013-01-25.

Williams 1994 — Williams, R. G.: Development Cost Prediction (1994). London, UK: The Institution of Electrical Engineers, pp. 1–4.

Willow Garage — Willow Garage: rosdep - ROS Wiki. Available online at http://www.ros.org/wiki/rosdep, checked on 2012-09-20.

Willow Garage 2008 — Willow Garage (2008): Packages - ROS Wiki. Available online at http://www.ros.org/wiki/Packages, updated on 2012-02-03, checked on 2012-04-07.

Willow Garage 2009a — Willow Garage (2009): About ros.org - ROS Wiki. Available online at http://www.ros.org/wiki/About%20ros.org, updated on 2011-08-31, checked on 2012-09-20.

Willow Garage 2009b	Willow Garage (2009): Documentation - ROS Wiki. Available online at http://www.ros.org/wiki/, updated on 2012-10-23, checked on 2013-01-25.
Willow Garage 2009c	Willow Garage (2009): File:PR2 Tabletop.jpg - Wikimedia Commons. Willow Garage. Available online at http://commons.wikimedia.org/wiki/File:PR2_Tabletop.jpg, updated on 2012-10-11, checked on 2012-10-24.
Willow Garage 2009d	Willow Garage (2009): ROS/Introduction - ROS Wiki. Available online at http://www.ros.org/wiki/ROS/Introduction, updated on 2012-02-03, checked on 2012-09-19.
Willow Garage 2009e	Willow Garage (2009): ROS/Tutorials/NavigatingTheFilesystem - ROS Wiki. Available online at http://www.ros.org/wiki/ROS/Tutorials/NavigatingTheFilesystem, updated on 2012-11-21, checked on 2013-01-25.
Willow Garage 2010	Willow Garage (2010): release/Releasing/First Release - ROS Wiki. Available online at http://www.ros.org/wiki/release/Releasing/First%20Release, updated on 2013-01-22, checked on 2013-01-25.
Wolfe 2012	Wolfe, Kevin C.; Moses, Matthew S.; Kutzer, Michael D.M; Chirikjian, Gregory S. (2012): M^3Express: A low-cost independently-mobile reconfigurable modular robot. In : ICRA, proceedings. IEEE International Conference on Robotics and Automation. St. Paul, MN, USA, May 14-18. IEEE: IEEE, pp. 2704–2710.
Wuang 2010	Wuang, M. S.; Huang, R. H.; Yang, C. L.; Lin, M. J. (2010): Key component supplier supports based on product complexity during New Product Development - A case study of Netbook PC manufacturers in Taiwan. In : Management of Innovation and Technology, proceedings. The 5th IEEE International Conference on Management of Innovation and Technology. Singapore, June 2-5. IEEE. Piscataway, NJ, USA: IEEE, pp. 1204–1207.

Xu 2007 — Xu, Bin; Hu, Hua; Ling, Yun; Yang, Xiaohu; He, Zhijun; Ma, Albert (2007): Efficient Collaborative Task Arrangement in Global Software Design via Micro-Estimation and PERT Technique. In : CSCWD, proceedings. 11th International Conference on Computer Supported Cooperative Work in Design. Melbourne, Australia, April 26-28. Melbourne, Australia: Swinburne, pp. 174–179.

Yenduri 2007 — Yenduri, Sumanth; Munagala, Sravanthi; Perkins, Louise A. (2007): Estimation Practices Efficiencies: A Case Study. In : ICIT, proceedings. 10th International Conference on Information Technology. Rourkela, India, December 17-20. Los Alamitos, CA, USA: IEEE Computer Society, pp. 185–189.

Zhou 2010 — Zhou, Fengyu; Tian, Guohui; Yang, Yang; Hairong Xiao; Wang, Xuewei; Wang, Wei (2010): Design and implementation of low-cost service robot mobile platform for intelligent space. In : ICAL, proceedings. IEEE International Conference on Automation and Logistics. Hong Kong and Macau, China, August 16-20. Piscataway, NJ, USA: IEEE, pp. 49–54.

Zia 2011 — Zia, Z.; Rashid, A.; uz Zaman, K. (2011): Software cost estimation for component-based fourth-generation-language software applications. In *IET Softw* 5 (1), pp. 103–110.

Developing a service robot poses many significant challenges due to its inherent complexity. One of these challenges is to approximate the development costs for a functional prototype even before detailed construction plans are available. The proposed approach explains a process that combines hardware and software engineering as well as established estimation techniques. Following the steps of the process yields a systematic and traceable cost prediction in early development stages

ISBN 978-3-8396-0749-7

FRAUNHOFER VERLAG