Environmental Engineering

Burkhardt Funk
Peter Niemeyer
Jorge Marx Gómez *Editors*

Information Technology in Environmental Engineering

Selected Contributions to the Sixth International Conference on Information Technologies in Environmental Engineering (ITEE2013)

Springer

Environmental Science and Engineering

Environmental Engineering

Series Editors

Ulrich Förstner, Hamburg, Germany
Robert J. Murphy, Tampa FL, USA
W. H. Rulkens, Wageningen, The Netherlands

For further volumes:
http://www.springer.com/series/3172

Burkhardt Funk · Peter Niemeyer
Jorge Marx Gómez
Editors

Information Technology in Environmental Engineering

Selected Contributions to the Sixth International Conference on Information Technologies in Environmental Engineering (ITEE2013)

Springer

Editors
Burkhardt Funk
Peter Niemeyer
Institute of Electronic Business Processes
Lüneburg
Germany

Jorge Marx Gómez
Department für Informatik Abt.
Wirtschaftsinformatik I
Universität Oldenburg
Oldenburg
Germany

ISSN 1431-2492
ISBN 978-3-642-36010-7 ISBN 978-3-642-36011-4 (eBook)
DOI 10.1007/978-3-642-36011-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013947063

© Springer-Verlag Berlin Heidelberg 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Enabling a sustainable development requires interdisciplinary approaches where computer science can provide the infrastructure for environmental data collection, analysis, simulation, decision support, reporting, and collaboration.

During the past decade methods and technologies at the interface of environmental engineering and information technology made important contributions to the transformation of organizations and processes toward sustainability. Material Flow Management systems, Tools for measuring environmental footprints, and smart grids are well-known examples for innovations in this area.

Since 2003, the conference series Information Technology in Environmental Engineering (ITEE) has established a platform for discussing the progress in the field. During the 6th ITEE 2013 conference practitioners and scientist met at the Leuphana University Lüneburg (Germany) to discuss recent developments, promising ideas, and new challenges in information management for supporting sustainability efforts.

Besides the special focus of this year's conference on information technology as an enabler for green logistics, the authors of the accepted papers cover a wide spectrum of topics from the perspective of different stakeholder groups. This includes topics such as big data, sponsored search advertising, mobile applications, energy consumption, and tools for collaboration.

This conference would not have been possible without many helping hands. We would like to thank all authors and participants of the conference for their contributions. We greatly appreciate the commitment and support of the program committee, namely Witold Abramowicz, Hans-Knud Arndt, Ioannis N. Athanasiadis, João Porto de Albuquerque, Jan Froese, Paulina Golinska, Lorenz M. Hilty, Horst Junker, Kostas Karatzas, Corinna V. Lang, Pericles A. Mitkas, Andreas Möller, Matjaž Mulej, Alexandra Pehlken, Andrea-Emilio Rizzoli, Adenilso da Silva Simão, Frank Teuteberg, Rüdiger Zarnekow. Last but not least, we want to thank the Leuphana team (Martina Darda, Karin Dening-Bratfisch, Madlen Schmaltz, and Norbert Tschritter) for helping at all stages of the conference.

Burkhardt Funk
Peter Niemeyer
Jorge Marx Gómez

Contents

Part I
Research Paper

Influence of Environmental Protection Requirements on Object-Oriented Software Design

Marat Abilov and Jorge Marx Gómez

Abstract The questions of environmental impact that company produce take important place nowadays. The ISO 14064-1 [1] standard specifies principles and requirements for monitor and control the greenhouse gas (GHG) emissions and removals. These requirements can be met by optimizing companies business processes (production, logistic, etc.) and by decreasing the power consumption of the companies' equipments. As side effect of these changes, the total costs of companies can be decreased as well. Companies' data centres and servers consume more than half of total spent electricity power. These servers are mostly used by companies' software systems. Hence, if the software systems require less calculation time, less space, there will be no requirements to keep the big energy consuming servers, and the most of tasks can go in cloud as well. Hence, the environmental protection requirements should be considered in developing software systems for companies. In this paper, we aim to give some literature review and propose the research on the topic.

1 Introduction

Software systems, which support the business processes of companies, are developed by using programming languages and methodologies. The most popular methodology today is Object Oriented (OO) Methodology, which helps to describe different domains using OO models.

M. Abilov (✉) · J. M. Gómez
Department of Business Informatics/Very Large Business Applications, Carl von Ossietzky University of Oldenburg, Ammerländer Heerstr. 114–118, 26129 Oldenburg, Germany
e-mail: marat.abilov@uni-oldenburg.de

J. M. Gómez
e-mail: jorge.marx.gomez@uni-oldenburg.de

B. Funk et al. (eds.), *Information Technology in Environmental Engineering*, Environmental Science and Engineering, DOI: 10.1007/978-3-642-36011-4_1,
© Springer-Verlag Berlin Heidelberg 2014

Environmental requirements can influence the software system design process in two ways:

1. They influence the changes in business processes and as the result the changes in software system design as well;
2. Software system by itself should meet these requirements: use less calculation time, and disk space, etc. These requirements can be treated as non-functional requirements (NFR).

In order to explore the problems of integrating these requirements to software design, the literature review of the works in this domain is explained in the following section.

2 Literature Review

2.1 Approaches for Software Power Estimation and Optimization

Hence, the power optimization can be treated as an environmental protection requirement, the review of works that deal with software power optimization and estimation will be provided. According to the review done, the existing approaches mostly deal with embedded software and on very low level.

One approach suggests the usage of symbolic algebra techniques in low power embedded software optimization [2]. The approach works on very low level of code and data types. The idea behind this approach was to optimize block of codes or algorithms of embedded software, so it would need less computation time, and correspondingly the less power. The first improvement over the peace of software is the consideration of moving from float-point to fix-point numbers where possible, because the fix-point ones require less calculation. The second is the energy profiling of code blocks and identifying the routines for optimization. Then these routines can be reformulated using polynomial approximation techniques. After optimization accuracy of produced code should be checked.

Measuring and estimation of instructions in assembly language over RISK processors were done by Russell and Jacome [3]. In this work authors developed power prediction model for low-level assembly program, that gives 99 % certainty with less than 8 % error.

Although the low-level approaches are important as well, in our approach we will consider mostly model-based, on more abstract level problems.

2.2 Approaches for Functional Requirements

The nearest research has been done by the work of [4]. In this work, the author tried to solve challenges existed in software development process with the proposed "Methodological Approach". This approach has 4 main stages: organizational modelling, purpose analysis, specification of system requirements, and derivation of OO diagrams. On each stage, different artefacts have to be developed. The artefacts from the last stage influence the subsequent software development stages. The most interesting stages, in connection to current research, are the last ones. During the System Requirement stage, "To-Be" Business Process Diagrams (BPD) in the form of EBPD have to be developed, that show how process should occur in organization after software system implementation. After that, the System Requirement Specification (SyRS) has to be developed in the form of Extended Task Description (ETD) templates. This proposed style of SyRS is a combined and improved combination of other approaches: among them are Lauesens Task and Support Descriptions [5], essential use cases [6], Info Case approach [7], and quality requirements from ISO 9126-1 standard [8]. The information presented in these templates is derived not only from BPD, but from previous stage's artefacts as well. These templates show mainly the functional requirements of the software system. After having the ETD specified, OO diagrams have to be derived from them according to set of rules presented in this work. Rules help to derive two main diagrams: the class diagram, and the state diagram. Among overall contribution of the reviewed work, there are several open issues in connection to current research:

1. This approach can mainly be used in waterfall model of software development, when all work on previous stages should be done in order to proceed to next stage. The connection to iterative, spiral or agile model was not specified in this work.
2. On every stage of this approach the validation is needed to be done by customer stakeholders, other validation techniques were not specified.
3. The process of analysing other non-functional requirements was not formalized and left for system analyzer to consider them in software architecture.
4. In class diagram a rule for specifying the control class was not defined.
5. The problem of parallel works in BPD was not analysed, and rules to consider them in state diagrams were not specified.

Analysis of this work shows that the next ideas can be effectively applied to current research:

- Enriched BPD, as an intermediate model between BPM and OO models.
- ETD templates, as a presentation of the expected behaviour of the software.
- Rules to derive OO diagrams from ETD, as a background, that can be extended to formalize this process for iterative methodology of software development.

In the work done by Loos and Allweyer [9], the authors analysed the Event-driven Process Chain (EPC) diagrams and made connections between then and UML diagrams. Three types of granularity were defined: object internal, object system inside and beyond the scope of an object system. Design of EPC diagrams has also to be conducted in three steps: high-level EPC with connection to information system packages, medium-level EPC with connection to classes, detailed EPC with connections to operations and attributes. The detailed EPC diagrams then can be used to derive the UML class diagrams and the state-chart diagrams of classes. Another UML diagrams were also analysed and relationships from EPC to them were established. Procedural model for applying the integration between EPC and UML was defined. The proposed approach can be used during requirement engineering and software design stages. Although the connection of different level between EPC diagrams and UML diagrams were presented, the concrete guidelines for establishing such diagrams were not specified. In class diagrams the procedure for finding relationships between classes was not specified.

The approach of deriving UML analysis models from use-case models (UCM) is presented in the work of [10].The approach deals with text-based, specified in restricted natural language (NL), UCM (RUCM) [11]. It contains 2 steps:

Step 1: The RUCM models have to be parsed and object model of them has to be generated. At first, the top level objects have to be identified: Use case, Use case specification, actors, brief description, the flows of events, pre- and post-conditions. Then, the sentences, objects are constructed from, have to be parsed by NL parser, the Stanford Parser [12].

Step 2: The UCMeta model have to be transformed into a UML analysis model. The overall algorithm of transformation and the set of rules were specified: for generating overall structure, for generating a class diagram, for generating a sequence diagram, and for validation.

When deriving the class diagram, three types of objects were analysed:

- Boundary objects, that handle interaction between actors and the system
- Entity objects, that are responsible for storing and providing access to data
- Control objects, that control the interaction of participating objects.

In UML these types of classes were specified with stereotypes: ≪Boundary≫, ≪Entity≫, and ≪Control≫. The first set of rules presented in this work deals with deriving this three types of classes and relationships between them. The idea of different stereotypes for classes and rules for deriving them can be found useful in current research.

In the work of [13], the authors presented the Component Bus System Property (CBSP) approach, which helped to connect requirements and architecture. This approach can be used in iterative software development model: the output of the first iteration can be used as an input in the second. In this approach, the decision technique is based on voting among multiple architects or experts. The CBSP process consists of 5 steps:

Step 1: Selection of requirements for next iteration. On this step, the most important requirements have to be analysed and prioritized among 2 criteria: importance, feasibility based on stakeholder decisions.

Step 2: Architectural classification of requirements by the relevance to CBSP dimensions, have to be decided by experts.

Step 3: Identification and resolution of classification mismatches, have to be decided by multiple experts independently. Level of consensus for requirement has to be calculated, and if the level is equal or more than largely then requirement should be accepted.

Step 4: Architectural refinement of requirements. On this step, requirements have to be rephrased and split, redundancies have to be eliminated. Then relevance coefficients of properties for each CBSP dimensions have to be decided among 4 common architectural styles.

Step 5: Trade-off choices of architectural elements and styles with CBSP. On this step requirements have to be refined and rephrased to CBSP model elements in such a manner, that no conflicts exist and all model elements at least largely relevant to one of the CBSP dimension.

Among overall contribution of this work, there are some open issues that are related to current research:

1. Analysed architectural styles are very high level; the derivation of detailed OO was not given in this work;
2. Business process as a type of requirements were not analysed in this work.

In the work of [14], the authors tried to connect OO and process-oriented (PO) models using formal approach. The procedure of translation from OO to PO models has 5 steps:

Step (a) Translation from State Machine Diagrams (SMD) to Heuristics Net using presented algorithm I;

Step (b) Translation from SMD to Annotation Repository using presented algorithm I;

Step (c) Translation from Heuristics Net to Petri Net using presented algorithm II;

Step (d) Creation of a skeleton structure of a process-centric model, using existing conversion plugins;

Step (e) Completion the process-centric model in the form of Yet-Another Workflow Language (YAWL) [15] model.

Obtained PO models are highly detailed. This work can be used to connect two different development approaches, also for analysing OO design from procedure perspective. Here are also some open issues in connection to current research:

1. This approach requires finalized OO models to be translated into PO ones.
2. This approach describes only one-way translation from OO to PO models. Although, the authors promised to start working on backward algorithm.
3. Validation technique was not provided in this work.

The approach, presented in this work can be used as a validation technique for the current research.

2.3 Approaches for Non-functional Requirements

The approach of integrating non-functional requirements and conceptual models: Entity-Relationship (ER), OO model, was done by the work [16]. The approach has several steps: on the first step the Language Extended Lexicon (LEL) [17] has to be developed. LEL registers the vocabulary of a given University of Discourse (UofD) general context, where the software have to be developed or operated [17]. While developing the LEL two principles have to be followed: maximum circularity, and minimum vocabulary. Each NFR has to be detailed using NFR graph, which defines the goal on the root and subgoals on the leaves. Subgoals can contain attributes, which can be general and data ones. General attributes characterize the whole system, while data attributes can be used for deriving ER and OO models. The paper presented the extensions to ER and OO models. In the OO model:

- Additional information, in the form of attached rectangles, was added to the class to show the names of used UofD and NFR.
- To improve traceability, NR_ prefix has to be added to classes, attributes and operations driven from NFR.
- Class names have to by symbols of the LEL.

Some heuristics to integrate the NFR into conceptual models were presented. This heuristics are not showing the straight-forward derivation of models, but the some hints that can help the software engineer in this process. Some ideas in this work can be used in current research:

1. Using LEL to describe the NFR;
2. NFR graph, to explore the influence of the NFR on design and;
3. Traceability idea: to present the name of used NFR and UofD in class diagram, that can be done, by using notes or comments.

The influence of the NFR over the decision of using software design patterns were analysed by the work [18]. For each system there might be multiple NFRs, and some decisions that support one of the requirements can hurt another. To address this problem NFR graph was used, that helps to reflect the decision trees for several NFR. This tree was used to describe the reason of usage of design pattern in the system architecture. The design pattern forces changes in the architecture, so author presented the way of connecting NFR graph with functional decomposition of the system.

3 Open Issues

To summarize the above research, there are several challenges in integrating the requirements as a whole, and the environmental protection requirements as a part into object-oriented software design:

1. Development of software system can be iterative, and not all information about business processes in organizations can be known before the design phase. Therefore, the new proposed approach in this work should be able to deal with incomplete information, and overcome the absence of some details within it.
2. Business processes of organization and non-functional requirements can influence the final design of the software system, so this approach should be able to deal with different types of requirements.
3. Relationships between classes are very hard to determine, and the problem increases when classes can be organized to perform some special behaviour. Some of these organizations are the examples of software design patterns. Design patterns and best practices can be used to solve the problems with software optimization, which as a result will give the less computation time, less power consumption and the less environmental impact as well. As a result, this approach should be able to define any kind of relationships in addition to show what kind of patterns can be applied.

4 Research Methodology

This study will be done by research project following the design research methodology [19]. The design science research contains the next steps [20]:

- Awareness of problem–understand the problem that occurs in practice;
- Suggestion for a problem solution from the existing knowledge base;
- Awareness of problem revisited. Problem can be detailed or generalized;
- Development of possible approach that can partially or fully solve the problem;
- Evaluation of the approach using empirical studies or formal methods;
- Conclusion shows the lesson learned and what influence the approach will have in the practice and science.

Empirical methods can be used in current research for validation the result of the approach. There are several empirical methods that can be used in software engineering context [21]:

- Controlled experiment—investigation of a testable hypothesis where one or more independent variables are manipulated to measure their effect on more dependent variables;

- Case studies—an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident [22];
- Survey research—used to identify the characteristics of a broad population of individuals;
- Ethnographies—help to understand how technical communities build a culture of practices and communication strategies that enable them to perform technical work collaboratively;
- Action research—attempt to solve a real-world problem while simultaneously studying the experience of solving the problem [23].

In most cases several of them can be used simultaneously in order to achieve the more rigidity of the research.

The next tasks are considered as support for answering the above-mentioned research questions:

1. Analyse environmental protection requirements, and their influence on changes in BPM and on non-functional requirements;
2. Analyse the influence of software requirements on the overall software design;
3. Analyse the types of relationships between classes and their derivation from BPM and other requirements;
4. Analyse the types of design patterns and their derivation from BPM and other requirements and
5. Design the approach for derivation OO models from BPM and other requirements that can be used in iterative, spiral and agile models of software engineering.

For the formalization of OO models derivation process, the ETVX (Entry, Task, Verification, and eXit) [24] approach can be used. ETVX is a table with 4 rows:

- Entry row—items required for the execution of the task;
- Task row—details of the activity;
- Verification row—checks and controls for the task;
- eXit row criteria, need to be satisfied in order to consider the task as completed.

References

1. ISO (2006) Greenhouse gases part 1: Specification with guidance at the organization level for quantification and reporting of greenhouse gas emissions and removals
2. Peymandoust A, Simunic T, De Micheli G (2002) Low power embedded software optimization using symbolic algebra. In: Design, automation and test in Europe conference and exhibition. Proceedings, pp 1052–1058
3. Russell J, Jacome M (1998) Software power estimation and optimization for high performance, 32-bit embedded processors. In: International conference on computer design: VLSI in computers and processors, 1998. ICCD '98. Proceedings, pp 328–333

4. de la Vara González JL (2011) Business process-based requirements specification and object-oriented conceptual modelling of information systems. PhD thesis, Polytechnic University of Valencia
5. Lauesen S (2002) Software requirements: styles and techniques. Addison-Wesley Professional, Reading
6. Constantine LL, Lockwood LA (1999) Software for use: a practical guide to the models and methods of usage-centered design. Addison-Wesley, Reading
7. Fortuna MH, Werner CM, Borges MR (2008) Info cases: integrating use cases and domain models. In: International requirements engineering, 2008. RE'08. 16th IEEE, pp 81–84, Sept 2008
8. ISO (2001) International Standard ISO/IEC 9126-1: Software engineering product quality part 1: quality model. ISO, International Organization for Standardization
9. Loos P, Allweyer T (1998) Process orientation and object-orientation-an approach for integrating UML and event-driven process chains (EPC). Publication of the Institut für Wirtschaftsinformatik, Paper, vol 144
10. Yue T, Briand L, Labiche Y Automatically deriving a UML analysis model from a use case model. Carleton University
11. Yue T, Briand L, Labiche Y (2009) A use case modeling approach to facilitate the transition towards analysis models: concepts and empirical evaluation. Model Driven Engineering Languages and Systems, pp 484–498
12. Stanford (2013) The Stanford Parser: a statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml. Accessed 06/02/2013
13. Grünbacher P, Egyed A, Medvidovic N (2004) Reconciling software requirements and architectures with intermediate models. Softw Syst Model 3(3):235–253
14. Redding G, Dumas M, ter Hofstede A, Iordachescu A (2007) Reconciling object-oriented and process-oriented approaches to information systems engineering. In: Proceedings of the 3rd international workshop on business process design (BPD07)
15. Van Der Aalst W, Ter Hofstede A (2005) Yawl: yet another workflow language. Inf Syst 30(4):245–275
16. Cysneiros L, do Prado Leite J, de Melo Sabat Neto J (2001) A framework for integrating non-functional requirements into conceptual models. Requirements Eng 6(2):97–115
17. Leite J, Franco A et al. (1993) A strategy for conceptual model acquisition. In: Proceedings of IEEE international symposium on requirements engineering, 1993, pp 243–246, IEEE
18. Gross D, Yu E (2001) From non-functional requirements to design through patterns. Requirements Eng
19. Hevner A, March S, Park J, Ram S (2004) Design science in information systems research. MIS Q 28(1):75–105
20. Vaishnavi VK, Kuechler Jr W (2007) Design science research methods and patterns: innovating information and communication technology. Auerbach Publications
21. Easterbrook S, Singer J, Storey M.-A., Damian D (2008) Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, pp 285–311
22. Yin R (2003) Case study research: design and methods. SAGE Publications, Beverly Hills
23. Davison R, Martinsons MG, Kock N (2004) Principles of canonical action research. Inf Syst J 14(1):65–86
24. Radice R, Roth N, O'Hara A, Ciarfella W (1985) A programming process architecture. IBM Syst J 24(2):79–90

Impact of Design on the Sustainability of Mobile Applications

Hans-Knud Arndt, Bartosz Dziubaczyk and Matthias Mokosch

Abstract Within the context of a consumer, who is anywhere contactable via mobile phone or a mobile access to the internet and his fast changing needs, it becomes more and more important to create a balance in the product-life-cycle. With regard to the design and sustainability of software the requirements to a programmer and the department of information technology in a company are getting higher, especially on mobile Applications (Apps) for mobile devices. The paper aims to explain what makes a good design for Apps and which types of Apps exist. Afterwards a description of the characteristics of a sustainable designed App and a comparison between Apple Design versus Metro Style (was changed due to copyright issues to "Windows 8-style UI") follows. In connection to the environmental friendliness of Apps it is to be recorded that material and energy have to be saved everywhere it is possible. Developers have the responsibility to create applications in a way that is as efficient as possible. The design of an App can be updated with just one mouse click which is far easier than updating hardware, because hardware can't be updated and has to be completely exchanged. The update principle is an important step towards sustainability. Software does not really pollute the environment but there is a kind of virtual pollution of the environment.

H.-K. Arndt (✉) · B. Dziubaczyk · M. Mokosch
Faculty of computer science—Working group Management information system, Otto-von-Guericke-University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: hans-knud.arndt@iti.cs.uni-magdeburg.de

B. Dziubaczyk
e-mail: bartosz.dziubaczyk@st.ovgu.de

M. Mokosch
e-mail: matthias.mokosch@st.ovgu.de

B. Funk et al. (eds.), *Information Technology in Environmental Engineering*, Environmental Science and Engineering, DOI: 10.1007/978-3-642-36011-4_2,
© Springer-Verlag Berlin Heidelberg 2014

1 Introduction

Today markets are characterized by an extremely fast moving and dynamic environment. By the increasingly more mobile consumers and thus rapidly changing customer needs, it is more and more important to create a balanced product life cycle. The contrast between a stationary computer and a small mobile device dis-appears not only with high-resolution display, but mainly by powerful processors and fast, reliable and inexpensive data traffic [1]. Thereby the requirements regarding to the design and sustainability of software, especially applications for mobile devices, are not less. The parameters for optimal functionality, security and usability are more complex for a mobile application than a desktop application [2].

The paper focuses on explaining the situation illustrated by the significant design ethics of the famous German industrial designer Dieter Rams. His design philosophy "Less, but better" represents the focus of this work. This paper aims to discuss based on the ten principles for good design by Dieter Rams, whether it is possible to ensure sustainability of mobile Apps and mobile devices through good product design.

2 Good Design for Mobile Applications

2.1 Braun Design as Creative Director of the Apple Design

Apple's iPhone released in 2007 turn the mobile internet through its innovative interface and its outstanding usability into a new experience. Before that vendors were like Nokia were the global leader in the field of mobile internet. In 2008 the supporting pillar of the app development was the platforms Symbian and Java Platform, Micro Edition. Nowadays, these platforms were hardly discussed. Many companies that specialize in these platforms are trying to develop their systems to the new platforms. The newly introduced platforms, such as Apple's iOS (2007) and Google's Android (2009) have made the breakthrough just by the new technical achievements, easy programming and a playfully easy handling and a new design language. Approximately 60 percent of all applications are developed for Android by which this operating system is considered to be the most popular, closely followed by iOS [1].

It has to be emphasized that mobile apps are not a new invention of Apple, which came with the release of the iPhone in autumn 2007. There were small applications before, but with different formats and on another scale. Java applications belonged to most of these applications. Also the handling was not the same as it is today's. The installation on a mobile device was complicated and time-consuming. In the first step the application had to be downloaded on the pc and in the second step they had to be transferred from the pc to the mobile device. Over

the time the thickness of a mobile device approximated more and more to that one of a sheet of paper. Although the data room can't be really expanded it is today possible to obtain millions of options on the display. To control and bundling these data Apple invented the principle of apps, which users could use for playing, communicating, shopping and informing or entertaining themselves. Today there is an App for almost everything. The iPhone takes over a leading role and shows that it is realizable in just a few easy steps to install and start mobile apps in a very simple way [1, 3].

Since the industrialization products were manufactured with a high degree of attractiveness to be able to sell them better. People buy and use more and more industrially manufactured products and endanger the environment. On that score the demand for the future has to be, that not only the designer but also the other areas within the production process have to take care of a lasting production. Unfortunately there are too many "junk products", which were bought and disposed after a little while because of missing functionality. Accordingly an immense demand at new products arises and the product life cycle gets into an imbalance. The great challenge is to produce less of those products, which waste unnecessary resources and strongly pollute the environment. Instead of this the focus has to be on manufacturing products, which fulfil the demands of their functionality and are an enrichment for life [4].

Over decades Dieter Rams, chief designer and member of the board of the company Braun, pursued the central idea of freeing the world from chaos and to redesign it entirely. Already at an early age he strove for a good industry design, which was rarely at that time. For Rams was the concentration on the essential and the simultaneously elimination of irrelevant characteristics aspects for a good design. Primarily the chaos has increased in the mass production in whereas factors like noise and the pollution of the environment are strongly weighted factors [5]. Everything starts in a small way. Also the beginning of new products or its further development begins in a small way for example the optimization of performance characteristics like a new user interface or an easier handling.

"Less but better! Much fewer but much better!" [6]. With these headwords Dieter Rams has changed the design language decisively. Due to the consideration that good design is not quantitatively measurable and that the world is overcrowded with mass market products, within the early 1980s Dieter Rams thoughts to himself what good design means for him. To canonize the bases of his work, he wrote 10 theses in the form of characteristics which distinguish between good and bad design: Good design is innovative, makes a product useful, is aesthetic, makes a product understandable, is unobtrusive, is honest, is long-lasting, is thorough down to the last detail, is environmentally-friendly and is as little design as possible [7].

This thesis arose from years of practice experiences and serves many design-oriented enterprises as a rough guideline to this day. Design is compared with the steady further development of today's technology and culture also a part of it and therefore develops as well [6].

The “10 thesis”, which Dieter Rams strictly followed in his design philosophy, reflect particularly the severe rationality. He put himself in the position of the user intensively, showed great sense of responsibility and was convinced that nothing is left to chance [5].

The German industry designer Dieter Rams was an idol for Jonathan Ive the master designer from Apple on the topic of abstraction and simplification to the necessary. While the creation of each new technical design blue print Apple leader Steve Jobs and Apple designer Jonathan Ive followed Rams main principle “lesser but better” when they thought about how to optimize the product’s design [8]. Steve Jobs developed his preference for a simple and functional design in Aspen, where he participated each year in the “International Design Conference”. Clear lines and forms are a symbol of rationality and functionality. Jobs was a great fellow of the Bauhaus-style, where the functionality and the essence of the products are central and Jobs wanted that Apple products look like the high-tech products of Braun, compact and bright. With this attitude he took the opposite design-pattern to Sony, whose design-pattern more and more became industrial black and heavy. Due to the introduction and evolution of their Walkman and the concept that sometimes “less is more” has a greater use for users. Sony was the first company to be successful on the portable player market. Helpful were the coinstantaneous reduction for the sake of mobility and growing individualisation [9].

Despite the fact that the Apple-products don’t have better technologies than those of their competitive companies and that they are far more expensive, they sell much better. It is agreed upon multiple facts that Apple’s homogeneous, harmonic design with its logically elaborated and user-friendly design-strategy, such as the image-based support campaigns for each single product released by Apple and their costumer-orientated advertising, Apple is put in the good position to have a continuous attractive product line-up [10].

According to the facts shown above, a label defines and separates itself from other labels by its design. The reason to buy an Apple product is based on its high design quality, which creates a significant additional value. Primary the costumers trust in a label is decisive, which can be achieved throughout a good design. This simultaneously defines the design’s additional value. Apple makes it possible for others to recognize its stability by evolving its design value according to its design strength and design continuity, which enables Apple to outdistance its competitors [11].

Dieter Rams, according to the facts already shown, can be called the forefather of the iPhone, which is characterized by its good design. Good design is till today a minority, whereas non-useful things, which have no self-explanatory quality, are far more common in most products. The environmental pollution takes place most likely on the visual plane. Another problem is the optical attrition. Cultures and with them the tastes blend into each other. A conclusion is that there are less different forms and everything becomes more akin [7].

The Meaning of design for Apple as a design-orientated company is not just beauty in its completeness rather than an amalgamation of three parts: The

simplicity and honesty of the configuration, the integration of the designer from the very start of the whole product development process and the equalisation of inventor and artist. Through all the diverse, innovative technologies and new challenges arise for the product design. The simplification of electronic devises allowed a better handling and achieved an innovative aesthetic. The maximum purism is not just the default for the outward appearance, but also for design of the surface of the operating system and the software. By pressing the Home-button, the only button the iPhone, the screen is activated and it appears a virtual slider, which requests the user to slide it sideways. On the unlocked screen the user see's just an orthogonal grid of the already installed miniature programmes. This clearly shows that the software is just as consequently simple designed as the hardware and the casing, to make sure that it is understandable and clear for the broad class of customers. Another fact is that the technology and the design must perform in sync, because the design has a huge effect on the company alignment [12].

2.2 Types of App Applications

The Apps are basically divided in two concept groups. Applications that can just be installed on own terminal devices and the existing operating system are also known as native applications. The counter model is the web-applications. These are the solutions, which are based on mobile optimized websites and download the compatible components of the applications from the internet [1].

The most important feature of a native application is the user-friendly handling that allows the immediate one-click-start of the App on the icon and a direct usage. This eradicates to type in long website addresses, the waiting for the website to be ready for usage and the scrolling and zooming if the website is not adjusted for the mobile usage [1]. Native applications are designed according to the criteria of the User Interface Guidelines platform and are developed and optimized for mobile purpose [13]. They are quasi made out of one piece, because the handling and the appearance are better adjusted to the devices than web-applications. Furthermore are device specific functions like the camera, the movement sensor and the Global Positioning System (GPS) useable without problems and allow the inventor the comfortable integration of the functions into the Apps. Last but not least this sort of Apps needs no active internet connection [1].

Another reason for the usage of the native Apps is the easy traceability in the Apps-store [14]. Due to the fact that more and more platforms get on the market, is the development of downloadable and self-installing Apps connected to a great afford and risk in marketing. The disadvantage is its low range, because the developer can't reach the broad masses and have to build an individual App for each of the significant operating systems. This problem does not only affect the support and marketing, but also has a long-term effect that shouldn't be underestimated. This long-term effect is primarily connected to the management of a contemplated evolution after a launch [1]. Even if the major advantage of the

native Apps is their independence they must pass through a time consuming and expensive certification process of the platform operator before they have to be distributed on a store.

Aside the native ones there are the browser-based solutions, the so called web-Apps, which more and more take the foreground role in the mobile industry. An elegant universal solution tries to combine all relevant platforms to minimize the development effort and costs [1]. To run a web-application a radio network for it completely acts in a browser. Furthermore do those Apps have no access to functions of the system such as the native Apps have [14]. Mobile web-Apps are controlled over a web-address and that's why they are independent from marketing policies of other companies. In addition do not all terminal devices have the same browser installed. All actually available browsers differ in their handling and quality. Despite that the whole development is cheaper and costs less time than the building and maintenance of a native App for all the today existing platforms [15]. The reason therefore is that a lot of companies are obliged in the software development to convert the content and context of a website to the parameters of a mobile terminal device [1].

In the meantime there appeared temporary solutions which usefully combine the important advantages of both models. Starting point is a hybrid-App that is based on a native-App core and a user-interface primarily based on standard-web-technologies. For the user behavior it means that the App is no longer running in a browser and is instead functioning like a native App.

This was enabled throughout the special frameworks named "PhoneGap" which allocates certain hardware functions via interfaces of web applications to directly activate the needed components. Therefore only the web-based components have to be developed by the company itself, because the native components are already implemented in the frameworks for the particular platform. On behalf of the native application it is possible to influence the individual device functions and the App can be published and distributed in a store. As it is a web application its development isn't too complicated and doesn't cause too much effort. The advantage of a hybrid App is that the problem of the diversification of browsers and operating systems is solved in a practical way [14]. The user has to waive in such a temporary solution the "Look and Feel" of a native App, because the App has to be compatible with many different operating systems [13].

An online survey of the agency "Culture to go" determines another rather surprising development of a prognosticated percentage on the continued existence of Apps of just 15 %. 41 of the 112 asked people said that the native App is just a makeshift for the not yet fully developed browser-based applications. On the other side 44 % think that both basic approaches can co-exist. Pure native Apps do have the right to exist, because of the applications that get more complex like games, navigation systems and they can be easily and targeted merchandised via the main distribution channel the App-Store. On the other side will web-Apps remain, because especially companies will use them based on their evolutionary advantages to be in a future-orientated position and that they can be scalable long-term solution [13].