

**USABILITY IN INCREMENTAL SOFTWARE DESIGN
- A USE CASE METHOD**

MIKA SÄRKIOJA

HELSINKI 16.2.2001

**MASTER'S THESIS
COGNITIVE SCIENCE
DEPARTMENT OF PSYCHOLOGY
UNIVERSITY OF HELSINKI
FINLAND**

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET

Tiedekunta/Osasto Humanistinen tiedekunta		Laitos – Institution Psykologian laitos	
Tekijä – Författare Mika Särkioja			
Työn nimi – Arbetets titel Käytettävyys inkrementaalaisessa ohjelmistosuunnittelussa – menetelmänä käyttötapaus			
Oppiaine – Läroämne Kognitiotiede			
Työn laji – Arbetets art Pro Gradu	Aika – Datum 16.2.2001	Sivumäärä – Sidoantal 85	
Tiivistelmä – Referat <p>Tässä Pro Gradu työssä käytettävyysarvioinnit sovitetaan inkrementaaliseen tuotekehitykseen. Tuotteen käytettävyysarvioinnit, jotka perinteisesti ovat olleet erillinen tuotesuunnittelun jälkeinen toimenpide, yhdistetään inkrementaalaisessa tuotekehitysprosessissa osaksi tuotteen vaatimusten määrittelyä.</p> <p>Menetelmä jolla käyttäjävaatimukset kommunikoidaan tuotekehitykseen on tapa soveltaa ohjelmistotuotannon mallinnuskieltä, UML-notaatiota. Käyttötapaus (use case) on osa UML-notaatiota, jonka avulla käyttäjävaatimukset määritellään. Tässä työssä ohjelmistotuotantoprosessin vaatimustenmäärittelyä modifioidaan inkrementaalisen tuotekehitysprosessin aikana, sisällyttäen käytettävyysarvioinneista saadut vaatimukset osaksi tuotesuunnittelua. Käyttäjävaatimukset luodaan suunnittelukokouksissa, joissa on läsnä sekä käyttäjä- että tuotekehitysryhmän jäseniä. Lisäksi käyttäjävaatimuksia tarkennetaan asiantuntija-arvioinneilla ja käytettyystesteissä. Tuoteidensuunnittelun suunnitteluratkaisut arvioidaan. Informaatio näistä käyttäjäarvioista kerätään osaksi käyttötapaus-notaatiota ja tuotekehitysprosessia. Suunnitteluratkaisuja muokataan iteratiivisesti kohti käytettyystutkimuksissa tarkentuvia käyttäjävaatimuksia.</p> <p>Käyttötapaus-notaation rakennetta analysoidaan. Työn mallinnukseen käytettävää mallinnusmenetelmiä käytetään keräämään tietoa tuotteen käyttöympäristöstä. Ohjelmistotuotteen suunnittelua tarkastellaan käyttäjän mahdollisten työnkulkujen suunnitteluna. Ohjelmistotuotannon mallinnuskieli rinnastetaan työnkulkujen mallinnukseen. Käyttötapaus-notaatiota käytetään mallintamaan ohjelmistotuotteen ja käyttäjän välistä vuorovaikutusta, lisäksi sen avulla havainnoidaan, kommunikoidaan ja modifioidaan käyttäjävaatimusten ja tuotekehityksen välisiä eroavaisuuksia.</p> <p>Formaalista ja luonnollista kieltä käytetään mallintamaan suunnitteluongelmia. Käyttötapauskset luonnollista kieltä käyttävänä mallinnusvälineenä, antaa käyttäjälle arviointitilanteessa mahdollisuuden lisätä yksityiskohdista puuttuvaa informaatiota. Tämä käyttäjäinformaatio voidaan käyttötapaus-notaatiota käyttämällä ottaa huomioon tehtäessä tuotekehityksen suunnitteluratkaisuja. Jos tuotteen käytettävyys on ohjelmiston suunnittelun onnistumiselle välttämätön edellytys, tämä käyttäjäinformaatio on kriittistä tuotesuunnittelun onnistumiselle. Formaali vaatimusmäärittelyt vaativat runsaasti työtä, sekä niiden kirjoittajalta että arvioijalta. Erityisesti jos tuotekehitysryhmän ulkopuolisten tahojen tulee arvioida suunnitteluratkaisuja. Käyttötapauskset antavat myös tuotekehitysryhmän ulkopuolisille tahoille apuvälineen, jonka avulla he voivat arvioida tuotekehitysratkaisuja.</p>			
Avainsanat – Nyckelord Käytettävyys, Käytettävyys tuotekehityksessä, inkrementaalinen ohjelmistosuunnittelu, käyttötapaus, UML, tuotekehitys, radioverkkosuunnittelutyökalut, vaatimusmäärittely.			
Säilytyspaikka – Förvaringställe Kirjasto			
Muuta tietoja – Övriga uppgifter Tutkielma on kirjoitettu englanniksi			

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET

Tiedekunta/Osasto Faculty of Arts		Laitos – Institution Department of Psychology	
Tekijä – Författare Mika Särkioja			
Työn nimi – Arbetets titel Usability in incremental software design – a use case method			
Oppiaine – Läroämne Cognitive Science			
Työn laji – Arbetets art Master's thesis		Aika – Datum 16.2.2001	Sivumäärä – Sidoantal 85
Tiivistelmä – Referat <p>This master's thesis integrates usability into incremental design process. Usability evaluation that traditionally has been considered of separate action of product design process is integrated into process of creating design requirements.</p> <p>The method to communicate user requirements into design is writing and re-defining use cases that are part of ULM notation, methodology to model software design. The use case itself is only a notation, but it is used as part of an incremental and iterative development methodology aimed to produce design. In this work the process use case creation and modification in incremental software development is modified to include information from the user. The customer requirements are created in participatory workshops, expert evaluations and in usability tests. Design is tested, Information is collected and feedback is provided into design process. The design is modified and the modified design is iterated with the user.</p> <p>The use case structure is analysed. The field study methodologies of workflow methodologies are used to collect the information from user environment, the context of use. The design of a software product is considered as a design of future workflows. Use cases are used to model the intended interaction between the product and intended user.</p> <p>The formal and natural modelling of a design problem is quite a different. Use cases as naturalistic model leave more space for interpretation, this allows user to fill in the detailed information that is missing from the design environment. If customer centred products are intended to be made this information from the user is crucial to success of the software development project. The formal and technical specifications require much work when generated, as well as when external stakeholders are spending time commenting the design. Use cases, if used, provide external representation of the design. This external representation can be used by all stakeholders of a design project to coordinate and communicate their actions.</p>			
Avainsanat – Nyckelord Software usability, Use case, UML, Design, Incremental design, Cellular radio network planning tools, Requirement specification.			
Säilytyspaikka – Förvaringställe Library			
Muita tietoja – Övriga uppgifter			

TABLE OF CONTENTS

PREFACE

ABBREVIATIONS AND TERMS

1. INTRODUCTION	1
2. DESIGN AND USABILITY IN DESIGN PROCESS.....	4
2.1 Human-Computer Interaction, usability and interaction design	7
2.1.1 Quality-in-use – design approach of usability.....	10
2.2 External and internal representation	12
2.3 Naturalistic and formal problem modeling	14
2.3.1 Naturalistic problem modelling.....	14
2.3.2 Formal problem modelling.....	15
2.4 Design and knowledge	16
2.4.1 Design Ability	18
2.5 Contextual design.....	19
2.5.1 Contextual Inquiry.....	20
2.5.2 Work modelling	22
2.5.2.1 The flow model	23
2.5.2.2 The sequence model	24
2.5.3 Consolidation	25
2.5.4 Work redesign	26
2.6 Unified Modeling Language.....	27
2.7 The Rational Unified Process.....	28
2.7.1 Use case driven.....	28
2.7.2 Architecture centric	29
2.7.3 Iterative and incremental.....	29
2.7.4 Development cycles	30

3. USE CASES.....	33
3.1 Ontology of a use case.....	33
3.2 Contents of a use case	35
3.3 Collecting the information.....	37
3.3.1 Analysis phase.....	39
3.3.2 Synthesis phase	40
3.3.3 Complexity.....	42
3.4 Use case template.....	43
4. CASE STUDY: IN DEVELOPING WCDMA RADIO NETWORK PLANNING TOOL.....	45
4.1 Usability in design project.....	46
4.1.1 Usability during the software design project start-up phase	46
4.1.2 Product definition and design activities	48
4.1.2.1 Concept definition workshops.....	48
4.1.2.2 Refining and updating use cases	49
4.1.2.3 Refining task analysis.....	50
4.1.2.4 Participatory capacity & traffic UI design.....	50
4.1.2.5 Main concept UI specification	51
4.2 Scenario testing.....	52
4.2.1 Method	53
4.2.1.1 User	53
4.2.1.2 The critical workflow path	54
4.2.1.3 Use cases and test scenarios	55
4.2.2 Results	57
4.3 Sample use cases and use case diagrams.....	67
4.3.1 System level context of use case	67
4.3.2 Sample of use case workflow path.....	68
4.3.3 Samples of traffic modelling use case diagrams	69
4.3.4 Samples of traffic modelling use cases	70

5. DISCUSSION.....	74
5.1 Help the user to design - participatory design.....	74
5.2 Completeness of design and usability testing	75
5.3 Design environment	76
5.4 Further research questions.....	80
6. CONCLUSION	81
7. REFERENCES.....	82

APPENDIX A: Software design development methodologies

APPENDIX B: The 3G planning process in nutshell

APPENDIX C: Radio network planning tool



Preface

This master's thesis was written for Radio Access Optimizer department of Nokia Networks during winter 2001 in Helsinki. The basis for this work was started at Radio Network Planning Tools department of Nokia Networks during year 2000 in Espoo.

I would like to thank Mrs. Anneli Korteniemi for the opportunity to write this thesis and Mr. Petri Salminen his professional advice during the work.

I would also like to thank the supervisor, Professor Pertti Saariluoma, for his guidance and comments.

Also special thanks for Tiina for helping me to forget all the small sorrows, when needed. And for making my life wonderful.

Further, I would like to thank my colleagues, friends and family for their support and encouragement during the work.

Helsinki, 16th February 2001

Mika Särkioja

Abbreviations and terms

2G	2 nd generation
3G	3 rd generation
DL	Down Link
BSS	Base Station Subsystem
BSC	Base Station Controller
BTS	Base Transceiver Station
ETSI	European Telecommunications Standards Institute
GOMS	Goals Operators Methods Selection Rules
GSM	Global System for Mobile communications
GSM900	GSM system operating in 900 MHz band
GSM1800	GSM system operating in 1800 MHz band
HCI	Human-Computer Interaction
HTA	Hierarchical Task Analysis
ISO	International Standardization organization
KPI	Key Performance Indicator
Low-fi	Low-fidelity
OMG	Object Management Group
OML	OPEN Modelling Language
QoS	Quality of Service
RFQ	Request For Quotation
RNC	Radio Network Controller
SHO	Soft Hand Over
Std.	Standard deviation
RUP	Rational Unified Process
UL	Up Link
UML	Unified Modelling Language
UMTS	Universal mobile telecommunication services
WCDMA	Wideband CDMA, Code division multiple access

1. INTRODUCTION

Usability activities and especially the usability engineering approach, which have traditionally been used in software development, had as its focus to produce measurable and reliable test results. Another emphasis has been cost effective usability activities. However little has been thought how to integrate these usability analysis results into the design process, and how to do requirement refining as early as possible. The goal of usability activities is to ensure a usable product. The earlier in the design process the usability activities are carried out and communicated into design, the more likely is that the product is usable in the end of the product development cycle. In this work this approach to communicate usability activities early in the development process is solved using use cases in every stage of design. This also integrates the usability activities into the design process. The benefit of this approach is verified by examples taken from a case study.

The perspective of this work is to emphasize the problem areas in Human-Computer Interaction in design and requirement phases, considering problems from knowledge creation and transformation point of view. After a brief Human-Computer Interaction and usability engineering introductory phase, use cases, that are part of a software design methodology called Rational Unified Process (Jacobson, 1992) are selected as a base method for requirement communication and usability ensurance throughout the design. Within Rational Unified Process selected usability methods are integrated into the process of creating use cases. The goal is to ensure good communication and understanding of user needs as well as to provide methodology to ensure manageable problem solving environment for the designer. The end-users' active participation in all stages of design is emphasized. This points out the issue of how and what kind of

knowledge is needed from design to end-user and vice versa. The target is also to evaluate this method from the aspect of how to communicate user needs to the designer when design solution is created. Problem formulation and solving methods are based on cognitive science's theories. Practical methods are taken from usability literature. Empirical data is collected from Nokia Networks' software design environment.

Focus

The focus of this work is to define methods for usability that can be integrated to an incremental and iterative software development project. Contextual design, a Human-Computer Interaction design methodology developed by Holzblatt and Beyer (Holzblatt and Beyer, 1998), is embedded into the software design methodology Rational Unified Process (Jacobson, Booch and Rumbaugh, 1998). The main hypothesis is that the use case method can be used to communicate user needs and usability research results to a software design project. First use cases are used to capture initial design requirements, then by defining more concrete use cases in design workshops; they are used to capture the user preferences in more detail. In addition, this method enables that during the development project, changes of design or changed user needs can then be communicated to all parties of the project. The use case approach to usability in software design is tested by using a case study. From the usability research point of view this approach creates a proposal of how to solve two problems in usability: How to identify early enough critical knowledge that is needed from the user to the design? And how to communicate results of usability activities and user requirements to software designers, documentation people and other parties in the software development process?

Designing Human-Computer Interaction is a much larger field than just designing usable software, however this study is restricted to considering software application development only.

The case study was made in software design environment at Nokia Networks, Radio Network Planning Tools department. A team of software designers was creating the very first design of what was to be the very first radio network planning tool for planning the third generation of mobile networks, which did not exist yet at the time. The expertise of radio network planning is quite a challenging task, as is designing complex software systems. The examples and processes used in this work, experiences and empirical data is taken from Nokia Networks, radio network tool development. Empirical problems rise from the need to ease the software product developer's problem solving, and how to make it coherent with constantly more refined end user's needs that were not known. Also the technology was under development so a large amount of assumptions were made, and then later in the project redefined when more information was available. In this intensive information-processing environment all methods that helped in producing simple and working design were appreciated. Software designers had a need to get appropriate information of the end user's needs. On the other hand, the end user needs knowledge of the product design in order to give constructive feedback for the designer. To make the situation more complex, both needed information about the domain that was under development (3G cellular radio network system, WCDMA). The delivery times were tight, so an incremental software development approach was used. As a conclusion, the requirements were constantly under changes and all parties in the project had a need to have access to redefine design requirements.

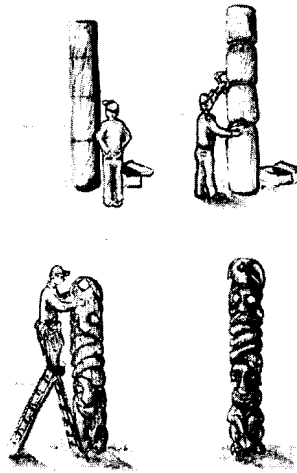
2. DESIGN AND USABILITY IN DESIGN PROCESS

Design is a process in which firstly a design concept is created. This concept is re-defined and tested in order to find out the possibilities of the concept, and then the concept is possibly realized as a product (Ulrich and Eppinger, 1995). Usability means how easily the end users reach their targets and how satisfied they are with using the product. (ISO/FDIS 13407, 1999) Usability assurance during this process of creation focuses that each time a new design is created designer's (mental) representation of the design solution is coherent with the end user's (mental) representation of the product use.

The design concept is the abstract level of any realization, a mental construction of what could be product alike (Ulrich and Eppinger, 1995). It is not necessary to work on details and solve all practical problems at once; quite often this is not even possible due to incomplete understanding of all details (Jacobson, Booch and Rumbaugh, 1998). The design concept helps communicating the initial abstract design ideas for the other members of the team. It allows early criticism and redefinition of the goal of design, as well as some means to approximate the scope and the workload that is required from individual members of the team to realize the product. At large, design is cognitive activity. Most of the time consumed is spent on formulating design problems, either from customer requirements, or how to implement technological innovation (Norman, 1998). If a technological innovation is made, this is implemented to product to add the value of it – in such the technological research turns into more valuable and competitive products. The designer formulates these requirements into a practical design problem. Some design problems can be answered by multiple different solutions. The design solution is one hypothetical potential

way to solve the design problem. The solution can be either accepted as such, discarded, or redefined in latter stages of the design. Most design problems are relative stable, the solutions for the problems can be modified several times (Lunequest, 1993). The goal for usability in the design process is to verify with the end user every step that creates some solution to the design problem, thus to verify every step that creates a practical solution for how to act in working environment. The problem in general is how to verify the design solution before actually implementing the design solution? The design, if not in balance with the end user's needs, requires much learning, time, and causes work errors to the end user. Creating design based on misinterpreted requirements also wastes the designer's time. Sometimes designers need to spend time solving wrong design problems that need not be solved at all.

The design is not a linear process, instead it consists of steps that each create solution and answers - as well as creates some design problems for the future (Ulrich and Eppinger, 1995). The designer and the design team have some shared vision of what should be the outcome of design when it is in an acceptable level (Kulak and Guiney, 2000). However in order to archive this goal the design concept is realized into product via various design "loops" –iterations that each add some details into the high level concept.

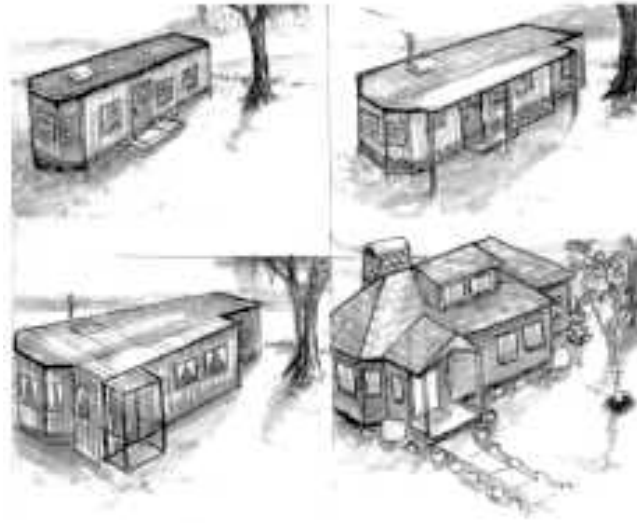


Picture 1 An example of iterative design, each round iteration makes the totem pole more finalised

Within each of these iterations added design solutions are evaluated against the design goal. Also their implications to other design solutions are evaluated in order to decide which actions are required in other parts of design, as well as to take full advantage of created solutions and minimize the duplicate work. The design process consists of series of iterations until the product is in acceptable level (Nonaka and Takeuchi, 1995).

As well as design consist of series of iterations the (software) product life cycle consist of series of releases, increments that each is sold to the customer (Jacobson, Booch and Rumbaugh, 1998). These individual product releases have enriched set of functional features, but they are based on same high level design concept – and are actually products of same process of iterative product design (Kulak and Guiney, 2000). However the design problem and the focus of design between different increments is radically different. The design team learns and develops individual as well as organizational knowledge along the process (Tuomi, 1999) This makes realization of the first increment cognitively demanding This cognitive workload diminishes in latter increments, thus the design team is able to

produce more complex design solutions in shorter development time cycles (Kulak and Guiney, 2000).



Picture 2 Picture of incremental design. Each iteration is a house that can be lived in. They are all products of a same construction process.

Usability in design process has goal to ensure good results and communication in design process. Proposed design process emphasizes understanding of users' needs and expectations as part of requirement design (Sommerville and Sawyer, 1997). It also integrates this knowledge into requirements and allows end user to verify design solutions.

2.1 Human-Computer Interaction, usability and interaction design

The phenomenon under study in Human-Computer Interaction is humans interacting with computers. The roots of Human-Computer Interaction (HCI) are in 70's software psychology, and it is a discipline with solid basis in experimental psychology. The intention was to accumulate pieces of empirical knowledge by controlled experiments

(Ehn, 1998). The scientific methodology would ensure that the knowledge was true. In order to provide general and useful knowledge, theories are induced from available data. The HCI theory should account for the data, but also be general enough to predict outcomes of new situations¹.

The difference between scientific Human-Computer Interaction research and usability engineering is that usability engineering is a process grounded into classical engineering. The impact of general HCI theory on professional software development has been quite limited (Ehn and Löwgren, 1997). Engineering is a process of economically building a working system that fulfills needs, not a process of building a perfect system with infinite resources (Wixton, 1997). Usability engineering emphasizes the empirical measurements of usability needs (Nielsen, 1993). Without measurable usability specifications there are no means to determine usability needs and measure if the product fulfills those needs.

Usability engineering typically consists of three steps, the first one of which is the user and task analysis, in which the intended users and their work tasks are analyzed. In the second step, in analogy to requirement specification, the usability goals are specified with measurable targets. Then the specification is used as a control instrument, in which the product prototype is designed, usability tested and redesigned until the usability target is met (Wixton, 1998). In this view of usability the measurements are very important. Several criteria for measurable usability engineering goals have been proposed. Nielsen proposes the following widely recognized usability definitions (Nielsen, 1993):

¹ An example of widely recognized general Human-Computer Interaction theory is the work done by Card, Moran and Newell in the *Psychology of Human-Computer*

- **Learnability:** the system should be easy to learn so that user can rapidly start getting some work done with the system.
- **Efficiency:** the system should be efficient to use, so that once the casual user has learned the system, a high level of productivity is possible.
- **Memorability:** The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn all over again.
- **Errors:** The system should have low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors should not occur
- **Satisfaction:** The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

In usability engineering the usability of the system is divorced from its utility (Nielsen, 1993). Usability engineering is primarily concerned with the users' efficient and error-free access to the services of the system, but not the appropriateness of the services. In usability engineering a strong emphasis is put on measurability, and this affect the research paradigm to de-emphasize utility. It is very hard to formulate measurable utility goals in lab settings beyond the obvious self-estimates (Landnauer, 1995).

The primary criticism against the usability engineering approach came with concerns that a strong focus on measurability runs with a risk of concentrating on easily measurable aspects, thus perhaps at the expense of more important considerations (Whiteside and Wixton, 1987). The primary focus of experiential usability research is the person's experience at the moment experienced. Usability becomes

purely subjective, experiential property of the interaction between a specific user and a computer in a specific moment of time (Ehn and Löwgren, 1997). Usability in this experiential perspective takes utility aspects into account as well:

"If the [usability] goals are not grounded in something really meaningful to the users, the resulting product will be useless to them." (Whiteside and Wixton, 1987., p. 20)

2.1.1 Quality-in-use – design approach of usability

There are three practices which are each using quite different approaches to studying the phenomenon that is called usability: scientific, engineering and design approach of usability. In recent years the development of usable systems has increasingly been seen as design work where "design" is understood as creative activity that is considered with the shaping of possible future realities. In Human-Computer Interaction the primary design considerations are situations of use, involving users as well as the system (Norman, 1993).

Based on Habermans' (Habermans, 1985) discussion of rationality in communicative action three phases can be characterized in terms of a system developer's main interest: objective, social and subjective.

The **objective** interests are based on a cognitive-instrumental rationality, concerned with the evaluation of objective facts. They lead to focusing on observation, empirical analysis and instrumental control for the purpose of rational design of the systems. The structure of a system, hardware and software are objective in the sense that they are inherent in the construction of a system and less dependent of context and interpretation.

Social interests are oriented towards a moral-practical rationality, related to evaluations of social action and norms and practices of social interaction. Interpretation, communication and the establishment and expansion of action-oriented understanding becomes in focus. Design becomes a process of anticipating possible breakdowns for the users in future use situations with the computer artifact being designed; anticipation in order to avoid breakdowns, as well as enabling the user to recover from them. This participatory design approach has two influential authors: The philosopher Hubert. L. Dreyfus and computer scientist Terry Winograd. Dreyfus argues for the relevance and importance of skill and everyday practice in understanding the use of computers (Dreyfus, 1972). Winograd brought this view into computer science and design of computer artifacts.

"Design is interaction between understanding and creation. And to be involved in practical use and understanding is that origin of design is found."(Winograd, 1986)

The social interests of design emphasize the functional aspects of the system. The functional aspects of a system concern are system's actual, contextual purpose and use. Typically different users have different purposes and usage of a system. Organizational dynamics and impacts belong to functional aspects, as do functions beyond the simple utilities of a system, like using a laptop as a sign of personal effectiveness (Keinonen, 1998).

Subjective interests rise from an aesthetic-practical rationality, addressing the sincerity in emotional and artistic expression. The form of a system expresses the experience of the user. Form is not a property of the system, rather than a relation between the system and the user. This further means that it is subjective, contextual,

contingent of user's previous experience (Ehn and Löwgren, 1997). A well-developed aesthetical sense and the ability to continuously improve the feel for quality are crucial for designing the use (Keinonen, 1998). In well formed design disciplines (such as architecture or industrial design), it is seen necessary for the designer to have good understanding of exemplars, traditions, epochs and styles of the discipline: classicism, functionalism, modernism, postmodernism, and so forth. The subjective interests of the design require the designer to concentrate on user experiences; to find out how it really is to use software. An analogue from architecture design would be that the designer new of a church would attend a service in some particular well recognized example instance, like the Duomo in Florence, to incorporate knowledge of aesthetic value system into design.

2.2 External and internal representation

The relation between external and internal representation of the product is problematic in the sense that the development team, having internal understanding of the functionality, misses the understanding of the user needs and difficulties of using the product. The users do not have access to the internal representation of the tool, They see the User interface and try to accomplish their goals using the services provided by the tool. They build external representation of the product; this external representation of the software functionality is based on interactions the user has with the product. In addition to concrete experiential interaction with the tool or prototype, the external representation is based on information provided by the development organization: Manuals, training, On-line help, Use cases... etc.

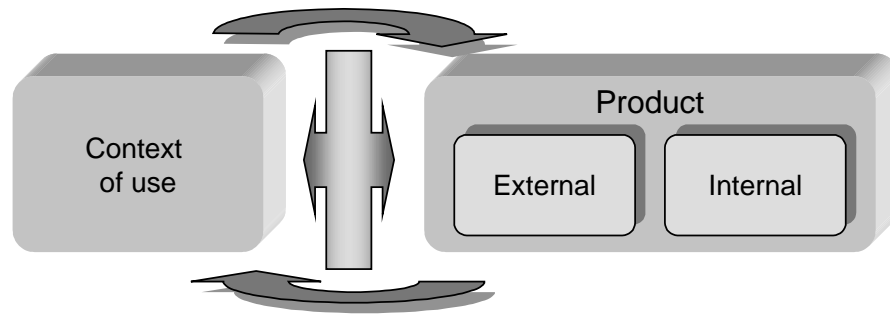


Figure 1 Context of use and internal and external representation (Salminen, 2000)

Usability in design could be understood as iteration between the product design and context of use. The more these iterations between the context of use and product are done, the more information is available to be included from the context of use. The use cases aim to describe the product design solutions in a transparent methodology that allows the end user, or end user representative to take more active role in validating the design solutions. The end user representative can consider the proposed design solutions as design proposals, they can be validated or re-defined – or discarded if so agreed.

The internal and external representations are made more coherent by using common methodology – use cases - to describe the concept and design solutions. When the users and other stakeholders outside the design team understand design, it can then also be criticized and modified even without detailed understanding of the system realization. The use cases are used to communicate changes in software design to both parties. Either by having internal or external representation the participants can influence the design when use cases are used.

2.3 Naturalistic and formal problem modelling

Use cases can be understood as a naturalistic modeling methodology in the sense that formal functionality is described in events that relate the context of use. The use cases serve as well the need of software designer to analyze and design the system functionality relating to the formal modeling that required to realize the functionality using some programming language. In human – computer interaction situations it is important to emphasize the formulation of knowledge in the problem solving a domain in naturalistic way, and study the interaction between naturalistic and formalistic view of knowledge in the software development context of use.

2.3.1 Naturalistic problem modelling

Natural problem solving is what makes every human capable of coping with the environment. The evolution has made selection of the fittest – and smartest. Humans have capability to extract meaningful context dependent phenomena and give a correct response on those. A human tries to control his actions according to environment demands, and human genre. Basis for this theoretical approach is taken from the evolution theory and current philosophical approach where autonomous living units, autopoietic systems, produce organization and thus solve problems to maintain themselves (Varela, 1991). An organism tries to adapt to his environment. When adapting to the environment demands, the organism formulates set of actions (mental or physical) that can be performed as routine acts. Routine act requires only procedural thinking, thinking can be performed without reformulating beliefs that describe environment, or set of actions that need to be performed to archive desired goal. In contrast to procedural

thinking, reflective thinking is required when this routine set of actions fails to create desired result as feedback. Reflective thinking re-formulates beliefs considering environmental laws or sets of own behavior. When a routine action fails to achieve the desired result, it creates a problem domain where actions and desired results are inadequate. The basis for reflective thinking is this action that has failed or disturbed. The inadequate feedback starts the process of reflective thinking (Kolb, 1984):

- 1) First the actor tries to conceptualize the problem. He makes an estimate of what is desired and what is causes the inappropriate result.
- 2) He conceptualizes basic guidelines and sets of rules for problem solving by formulating the problem. This conceptualization leads to information gathering and gives boundaries for the problem.
- 3) The next step is to analyze the boundaries; possibilities are resources to solve the problem. Based on these the working hypothesis is built.
- 4) Working hypothesis is tested in mental environment. If corrections are needed these corrections are made.
- 5) The final step of naturalistic problem solving is to verify thinking by action. The new set of actions is performed and only if it produces correct results the problem solving is complete.

2.3.2 Formal problem modelling

Aristotle started collecting syllogisms. Euclid placed the abstract laws of geometry. From the mathematicians and philosophers of early 20th

century we got the laws of formal logic as well as the first theory of computational problem solving, that has then been successfully implemented into every-day life by computers. Through training and education humans learn formal problem solving methods. Laws of physics, mathematics and logic. Formal problem solving methods produce deductively correct information. Computer programming languages are a good example of formalistic communication. Even though passing formal education very large part of human information processing is not based on formal decision theorems. Humans do err, associate, and estimate. Environmental demands do not give time for formal decision making – or sometimes using that much cognitive resources would be a waste. Humans do think and live in natural world. Rather using some rough heuristic than complete decision tree. Turing machines, like computers don't have this option. Formal information processing takes into account all information available and uses specific criteria to verify the next step of action. Computers do use different problem formulation logic than humans and this causes some of the problems of Human-Computer Interaction.

2.4 Design and knowledge

In Human-Computer Interaction design is problematic in the sense that a designer, when creating a concept of artifact, also creates a concept of design for using the artifact in the real world. The worlds as we see consist of artifacts that have a large amount of build-in knowledge (Norman, 1988). Each time an artifact is created also the user environment changes. The inventions and design affect the tasks and perception of the user. The Human-Computer Interaction, in its large sense, describes the design of a software application, and the design of how it fits into the context of use (Holzblatt and Beyer, 1998). The context of use is the environment where humans use the artifact. When defining what is context of use, the easiest way is to refer to the real world. Thus context of use is the real world where

application and humans interact. The ISO standard definition for context of use is: users, tasks, equipment (hardware, software and materials), and the physical and social environments in which the product is used [ISO 9241-11:1998 definition 3.5].

The conceptualization of the design process facilitates the understanding of what happens in design and how designers use different kinds of knowledge. It consist of three non-sequential modes of operation: conceptual, constitutive and consolidary (Lundequist and Ullmark, 1993).

The **conceptual** step is guided by the designer's (or design teams) vision, which is typically not very distinct: unclear in parts, hard to communicate. To clarify the vision the designer matches it to the repertoire of known structures, which are called formats. The matching process is a physically tangible activity in which externalizations of different kinds are used. Externalizations – like use cases - are used to guide the work and communicate with the stakeholders.

In the **constitutive** step, the concept from the conceptual step is confronted with typical use situations in order to raise questions about requirements and constraints. The concept is modified and extended and possibly abandoned in favor of a new wave of conceptual activity.

The **consolidary** step is concerned with refining a proposed solution in terms of simplicity, elegance and appropriateness for long-term use. Value based judgement and experience are crucial components of consolidation.

2.4.1 Design Ability

The specific competence of a designer, the design ability, has come in the focus, not least because of the importance of tacit knowledge as well as contextual knowledge in design and of conditions of creativity (Nonaka and Takeuchi, 1995). In design the existing knowledge of individual designer is a starting point. Organizational knowledge consists of individual actor's knowledge as well as how these individual organizational actors interact (Tuomi, 1999). In the individual level, the designers of the design team have their procedural and explicit knowledge. In the organizational level, some social practices and the working context, the shared inter working methods, create the implicit organizational, knowledge bases - the organizational tacit knowledge (Tuomi, 1998). In addition to tacit knowledge the organizational knowledge consist of documented working guidelines, project management processes, knowledge management systems, and access to internal documentation via intranet (Tuomi, 1998).

The importance of tacit and contextual knowledge has already emerged in participatory design approaches, where design was done involving a user in design team. The design approach of usability suggested new criteria for usability.

"The key criterion of system's usability is the extent to which it supports the potential for people who work with it to understand it, to learn and to make changes. Design for usability must include design for coping with novelty, design for improvisation, and design for adaptation."(Adler and Winograd, 1992, *ibid.*, p.7)

2.5 Contextual design

The experiential view of usability formed a basis for the software development approach known as contextual design (Whiteside et al., 1988; Wixton et al., 1990; Holzblatt and Beyer, 1998). Contextual design is described as a cyclic process of requirements generation, design, implementation and evaluation, following the ISO 13407 human-centered design.

ISO 13407: Human-centred design activities

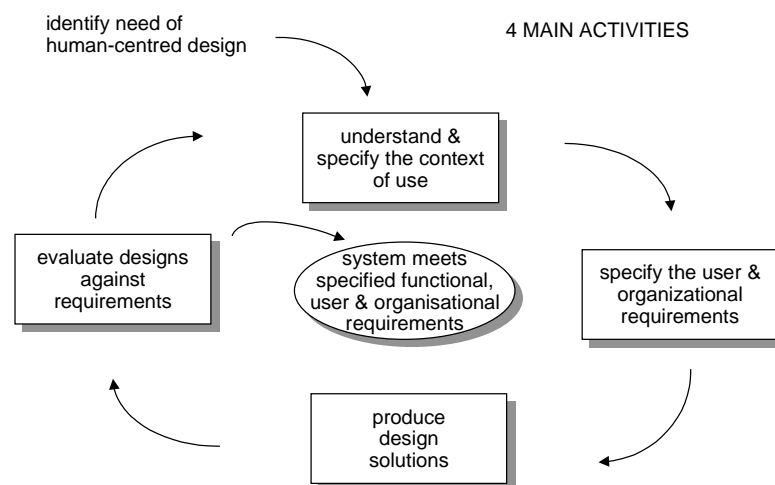


Figure 2. Human-centred design activities

A set of techniques is used to ensure that the developer understands the user and customers, and this understanding is reflected to the system. A contextual data collection method is called **contextual inquiry** (Holzblatt and Beyer, 1998). It is based on ethnographic and field research techniques that are used to acquire the necessary understanding of user's work. Diagrams and modeling languages are used to produce a coherent interpretation of the collected data. The work is redesigned in a participatory effort, and if needed evaluated using further contextual inquiry. The conceptual structure of the system is designed, and after that the conceptual structure is mapped to the user interface. Conceptual design is tested using paper mock-ups; as the shared understanding grows, the prototypes can be more detailed. The emerging understanding also prompts further contextual inquiry and work redesign iterations.

2.5.1 Contextual Inquiry

Interviewers observe users doing their real work in their real workplace. Interviewers have permission to interrupt users at any time to ask questions, and to converse about work. Interviewers also ask users' opinions on the interviewers' design ideas, and may ask users for their ideas of improvement in their work and current interface. Four principles guide this information gathering technique: **context**, **partnership**, **interpretation** and **focus** (Holzblatt and Beyer, 1998).

The principle of **context** requires getting as close as possible to be physically present. Staying in context enables the interviewer to collect ongoing experience in detail. The actual structure of work into the smallest detail can be collected. Concrete naivistic language is used to avoid abstractions and interviewers own interpretations of work structure.

The goal of **partnership** is to make the interviewer understand the work structure in collaboration with the user. The only person who really has the tacit knowledge (and into some extend the explicit knowledge) of the work is the one doing it (Nonaka and Takeuchi, 1995). In a traditional interview the interviewer controls what is asked and discussed. The interaction according to the principle of partnership is such that while the work is progressing, the user is engrossed in doing it, and the interviewer is busy observing the details as the work unfolds (Whiteside and Wixon, 1988). The interviewer is looking for patterns and structure, and creating a model of reasons behind user actions. If some observation is made that does not fit into the initial pattern or if the interviewer notices a structure underlying the work, the interviewer interrupts and explicitly verifies the observation. This causes both the user and interviewer to withdraw from the work. When the reasons behind the observation are clarified the user and interviewer return to work.

The principle of **interpretation** underlines that good facts are only a good starting point. Designs are built on the interpretation of facts (Habermans, 1985). It is not enough to observe and bring back observations. Interpretation is a chain of reasoning that turns a fact into an action relevant to designer's intent. From the fact, the observable event, the designer makes a hypothesis, an initial interpretation of about what the fact means or the intent behind the fact is. This hypothesis has implication for the design, which can be realized as a particular design idea for the system. Designers will interpret facts, and re-interpret what the facts mean. The interpretation should be verified to be correct by letting the user verify the designer's interpretation.

Focus defines the point of view that the interviewer takes while studying work. What aspects of work matter and what do not? The interviewer needs data from specific kind of work, and this needs to be observed. Focus also gives the interviewer a way to keep conversation topics useful without taking control entirely away from the user. Having focus means that the interviewer can see more of those aspects he is focusing at.

2.5.2 Work modelling

Work models are used as a graphical language to capture knowledge about work (Holzblatt and Beyer, 1998). They provide a shared focus and means to communicate the observations of the individual interviewer to the design team. Work models are an external concrete form to record and communicate. The workflow model represents the coordination necessary to do the work. The sequence model represents detailed steps for how to reach the goal. The work models model the current existing workflow which has been observed, in contrast to the use cases model that is the intended workflow (Kulak and Guiney, 2000). The flow model exists to describe the current context of use. It is created before any design to enable design solutions to be grounded into a context of use.

2.5.2.1 The flow model

The workflow model represents the coordination necessary to do the work. It offers bird's eye view to the organization. It shows the people and their responsibilities, the communication paths between people independent of time, and the things communicated (Holzblatt and Beyer, 1998). People and organizations are bubbles in the model, annotated with their position or responsibilities. The flow is indicated by arrows between the bubbles, with the kind of communication written on line. Artifacts are shown as light blue boxes, and the informal communication is shown in white boxes.

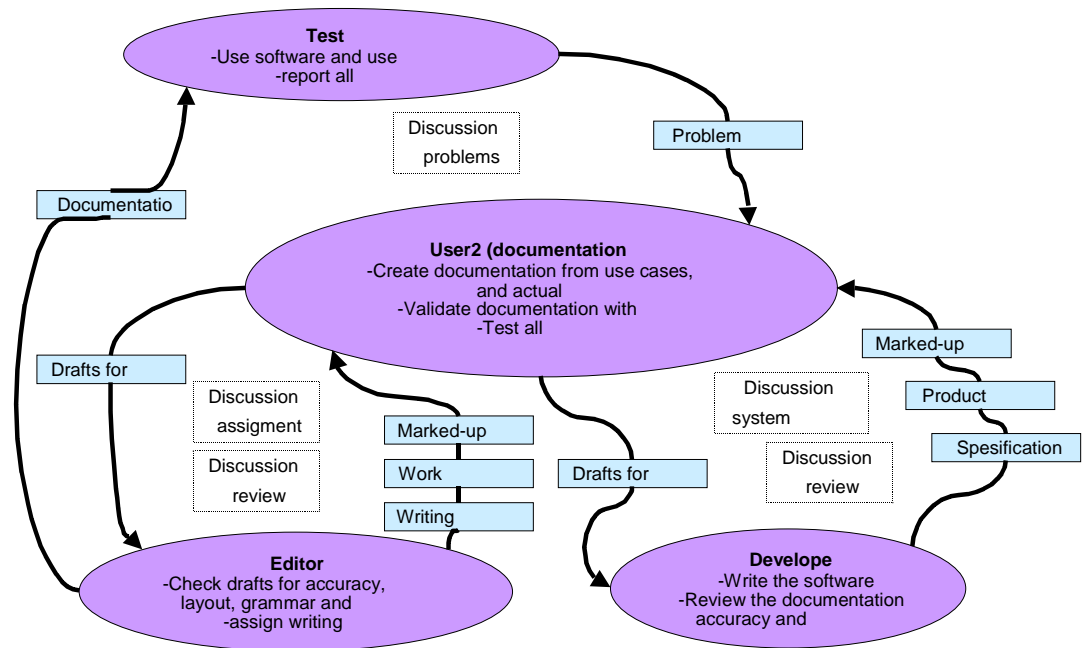


Figure 3. Flow model

2.5.2.2 The sequence model

The sequence model represents detailed steps of work to reach the goal. Understanding the real goal for user action is essential for understanding the actions taken to reach that goal. The designer can redesign, modify and remove steps done by the users to reach their goal(s), as long as the user can reach his goal(s). The steps of action reveal users' strategy and goal, as well as what matters to him. All work, as it unfolds, becomes a sequence of actions; each action is a step to achieve some goal. The sequence model represents the steps by which the work is done as well as triggers that launch some set of steps.

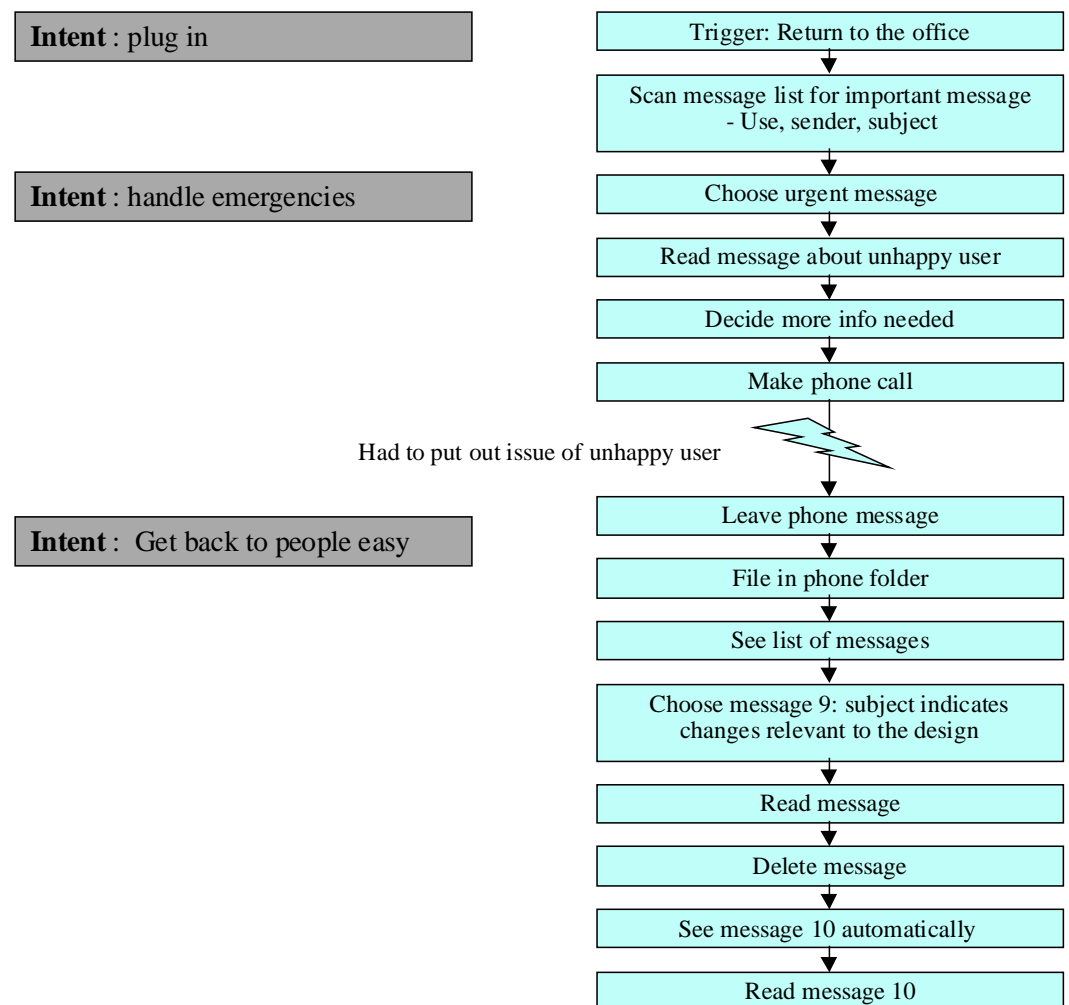


Figure 4. Sequence model

2.5.3 Consolidation

Consolidation is a step of design that creates one shared common understanding of users' needs (Kulak and Guiney, 2000). Systems are designed to user populations, not individual users. Rather than developing a system that answers the needs of one individual user or user group, the system needs to support whole user population. The larger the amount of intended users, the larger is the variety of work practices that that designed system needs to take into account (Holzblatt and Beyer, 1998). Without external representation the design team has only their individual opinions and unarticulated knowledge to base their decisions on (Sommerville and Kotonya, 1998). They have no concrete way to communicate and justify their design. The observed work models represent individual users in a specific context of use. Consolidation creates one shared common understanding of individual users' workflow. This gives the design team ability to make strategic actions to achieve long term design goals.

The consolidated workflow is generated from individual empirical observations, workflows that each captures one instance of user population and their work. The individual workflows are abstracted and the common patterns that workflows include are included in the consolidated workflow. This step of abstracting common patterns from individual workflows is called consolidation. According to participatory design paradigm, consolidation should be done in co-operation with users (Muller, Halswanter and Dayton, 1997).

The challenge of consolidation is to do explicitly, on purpose and externally work models of whole user population from particular instances and events (Holzblatt and Beyer, 1998). At this point instances of customer experience have been captured. Consolidated work models show how the individual examples of work practice are

instances of patterns that define the whole user population. These consolidated models provide concrete representations of whole user population patterns of work.

2.5.4 Work redesign

When designing a new software system the real invention is a new way for people to work. Even the smallest software tool must fit into larger work practice. In every case, what makes a system interesting is the new work process it makes possible (Norman, 1998).

The design team delivers a work practice, but the way it is delivered is through a system solution. This includes the system itself – the hardware and the software - but it also includes documentation and training (Jacobson, Booch and Rumbaugh, 1998). Requirements go beyond the description of the real world to invent and choose one specific solution to a need. The work is re-designed. The design of work describes the actions of the user and the actions of software, thus this is the critical step of Human-Computer Interaction design.

The method to model this critical step of design proposed in this work is use cases (Jacobson, Booch and Rumbaugh, 1998). Use cases model the intended interaction of user and (software) design. They provide a method to model and design future workflows. In contrast the flow model and the sequence model that describe the current existing work, the use cases describe the re-designed intended workflow.

Myth of invention tells about some brilliant person goes up to the mountain (in software case into garage), and suddenly invents something new out. This is very unlikely to work out in real world (Kulak and Guiney, 2000). The first design ideas of future work is re-designed several times during design process (Nonaka and Takeuchi, 1995). The work model, the sequence and the contents of it are

revised and completed in more detail, the more knowledge the design process produces of the intended [software] design and workflow (Senge, 1990). Use cases as a method to document and communicate these changes are constantly revised when new design is created.

2.6 Unified Modelling Language

The older branches of engineering have long found it useful to represent designs in drawings. Various drawings have existed since the early programming days. Developers need a language to communicate with others and provide a setting in which the individual designers can think and analyze (Jacobson, Booch and Rumbaugh, 1998). The design also needs to be communicated to external stakeholders such as users. The Unified Modeling Language (UML) is a standard modeling language for software, to visualize, specify, construct and document the artifacts of a software system. Use cases are part the Unified Modeling Language This language was bought to existence in 1997 and has been adopted as a standard by the Object Management Group (OMG), an industry consortium (Kulak and Guiney, 2000). Basically, the UML enables the developers to visualize their work in standard blueprints or diagrams.

UML is a notation, a way to document system specifications. It is not a methodology (Jacobson, Booch and Rumbaugh, 1998). A notation simply structures the system documentation. A methodology, in contrast, consists of a step-by-step guide of how to construct a system. UML is not dependent of any particular methodology, nor do methodologies have to be consistent with UML. It requires only that the computer system developed has object-oriented components.

UML was a result of collaborations between three famous object oriented methodologists, the "Three Amigos": Grady Booch, James Rumbaugh and Ivar Jacobson (Kulak and Guiney, 2000). They

combined their own modeling notations. The result is that the computer industry has a common language or notation, with which to specify object oriented systems². The UML notation consists of several formal notations, each of which describes some aspect of software. The notations have interactions and based on one notation others can be generated.

2.7 The Rational Unified Process

The Rational Unified Process (RUP) is a software development methodology that the founders of UML have developed. In a long line of software development methodologies the Rational Unified Process is the latest paradigm and currently being adopted in software design. There are two major factors that have contributed to the development of this methodology: Software products are becoming more complex and the time to deliver the products (time to market) is becoming a more critical factor of the overall success of software products (Jacobson, Booch and Rumbaugh, 1998). There are some main principles of RUP that characterize the development process: the process is use case driven, architecture-centric, iterative and incremental, and the development cycle is re-defined.

2.7.1 Use case driven

Software is brought into existence to serve its users. To build a successful system, the user needs need to be captured. Use cases are capturing the intended interaction between a system and an external actor. In UML this external actor refers not only to human users but also to anything that is outside the designed system and interacts with it. Interaction of this sort is a use case. A use case is a piece of

² Note: OPEN modeling language (OML) competes with UML

functionality in the system that gives a user a result of value, e.g. to accomplish a user's goal. Use cases capture the functional requirements. All the use cases together make up the use case model, which describes the whole functionality of the system. However, the use cases are not only a method to capture requirements, they also drive the development process. Based on the use case model the developers create design and implementation models that release the use cases. Also the design solutions are verified against use cases, to ensure that the proposed design is what was intended. Also the testers test that the implementation correctly follows the use cases. Use case driven means that the whole development processes goes through a series of workflows that derive from use cases. Use cases bind the workflows together.

2.7.2 Architecture centric

Architecture and concept provides the structure in which to guide the work in the iterations. Use cases are not selected in isolation. They are developed in parallel to the system architecture. Use cases drive the system architecture and the architecture influences the selection of use cases. Every product has a function and a form. The use cases describe the functionality of a system and the architecture is a form that is realising the system. Thus they have to be developed in parallel. The architecture of a system is analogous to a skeleton covered with skin [user interface], but a little muscle [software] between skeleton and the skin (Jacobson, Booch and Rumbaugh, 1998). The system is the whole body with the skeleton, muscle and skin.

2.7.3 Iterative and incremental

Developing a commercial software product is a large task, requiring time and effort (Sommerville and Sawyer, 1997). It is practical to divide the work into smaller mini-projects, increments that can be

controlled. Each mini-project is an iteration that results in one increment. It includes specifying/refining use cases, the analysis phase, design, implementation and testing. The iteration deals with a set of use cases that expand the services of a software product to satisfy user goals (Regnell, 1999). Secondly, it manages the risks within each mini-project. In each iteration the developers identify and specify the relevant use cases, create the design, implement the design, and verify that the components satisfy the use cases.

2.7.4 Development cycles

The Rational Unified Process repeats itself in a series of cycles that each results a **release** to customers. A release is a product ready to be delivered to customers. It consists of the software, manuals and associated deliverables like training (Jacobson, Booch and Rumbaugh, 1998). Each cycle consist of four phases: Inception, Elaboration, Construction and Transition. The phases are further divided into iterations.

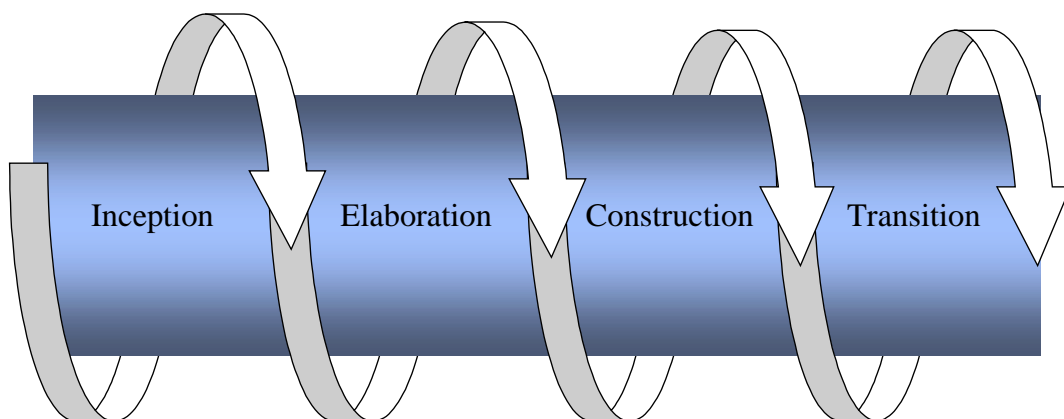


Figure 5. Inception, Elaboration, Construction and Transition the phases of Rational Unified Process

The inception phase develops the idea of a product into a shared vision of the product: It results in the plan of primary functionality, what the system is going to do to each of its major users, described in high level use cases. It includes the tentative architecture and the initial concept of a product. In this phase, the costs for developing the product are also estimated (Jacobson, Booch and Rumbaugh, 1998).

In the elaboration phase the concept of the product exists. Most use cases are described in detail and the architecture is designed. Detailed project plans can be made. During this phase of development the most critical use cases are realised (Jacobson, Booch and Rumbaugh, 1998).

During the construction phase the product is built. The architecture grows to become an almost functional system. The concept is realised. The vision evolves into a product and work practice that can be transferred to user population (Norman, 1998). At the end of this phase the product should include all the use cases agreed to be developed (Jacobson, Booch and Rumbaugh, 1998).

In the transition phase a release is made for some selected pilot users. A small number of experienced users will have the product in use to evaluate the tool and report deficiencies. The transition phase includes the training of customer personnel and correcting bugs found in the delivery release (Jacobson, Booch and Rumbaugh, 1998).

Design and usability in design process

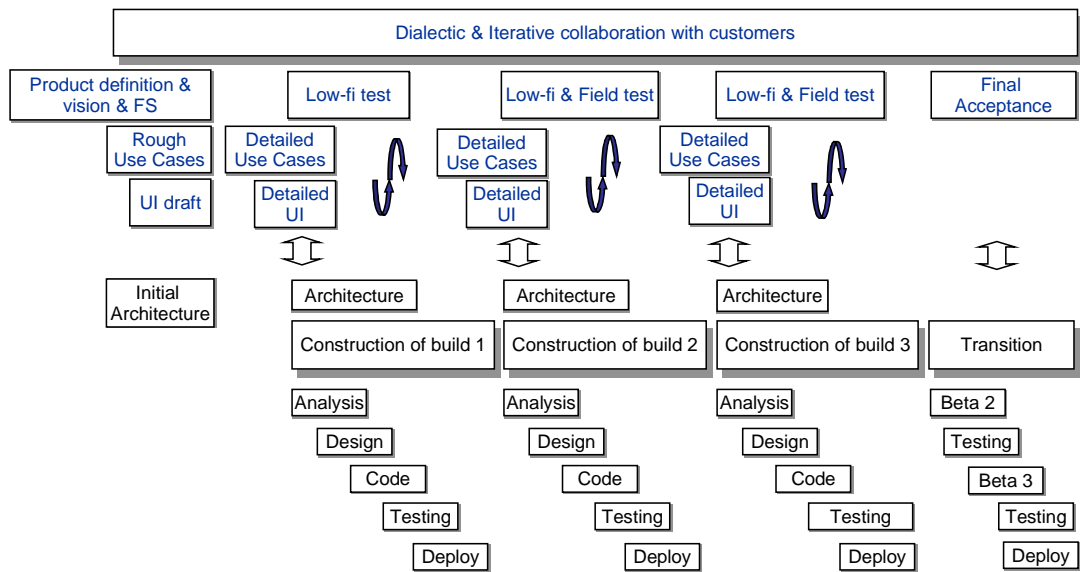


Figure 6. Illustrates the phases in RUP, phases of a release cycle and activities within on increment as well as possible usability testing activities (Salminen 2000).

3. USE CASES

A use case consists of one specific task that the user tries to accomplish with a tool. Only one possible workflow is described in a use case (Regnell, 1999). The use case is a realistic description (as realistic as it is possible to create from available information.) It is a description of the workflow, where all the intentions and actions of the user are explicitly described (Jacobson, 1992). The language is naivistic, and all concepts are clearly and explicitly explained (Kulak and Guiney, 2000). The demand for explicit description rises from the need to effectively test the designers' mental model against users' and other designers' mental models of the task. In software product development, the overall demand for coherent mental representations is also important, because the expert knowledge needs to be enabled for the end user to use, even without the complete understanding of the problem solving environment (Zachary, 1997). Use cases need to be realistic so that they can actually be verified or falsified in the real context of use by expert user.

3.1 Ontology of a use case

The distinction between what is inside and outside the system is designed in the first step (Regnell, 1999). The designed system is called the **target system**, and the context of use in which the system will operate is called the **host system**. The **users** of the target system are part of the host system. Inside the target system there is a number of services. A **service** is a package of functional entities (features) that is offered to users to help them reach their goals (Regnell 1999).

Users can be of different types, called **actors**. One user is an instance of an actor (Kulak and Guiney, 2000). The actor represents a set of users that share common characteristics of how and why they use the target system. **Goals** are objectives that the users have when they use the services of a target system (Regnell, 1999). Using goals, the users can be categorized into different classes of actors.

A **use case** is a model of situation where users try to accomplish their goal(s) using one or more services provided by designed system. (Jacobson, Booch and Rumbaugh, 1998). Use case can model either a successful or unsuccessful attempt to reach goal(s). Every use case has a **context** that dictates what the environment of usage is like and defines the **preconditions**, the properties that need to be filled in the host and target systems before the use case can be done. The context also defines the **postconditions**, the properties of the host and target systems after the use case has been performed (Regnell, 1999).

A use case can be divided into parts called **episodes**. The same episode can occur in many use cases. Each episode consists of three kinds of **events**: **stimuli** (messages from users to the target system), **responses** (messages from the target system to users) and **actions** (intrinsic events within the target system that need no user participation). The stimuli and responses, if specified in detail, can have parameters that carry data in and out from the target system (Regnell, 1999)

One use case can be realized in a **scenario**. The scenario is a realization of a use case in a sequence of a limited number of events in linear time order (Kulak and Guiney, 2000). A use case may cover several scenarios and it might include repetitions and interruptions that might happen any time. A scenario is a specific realization of a use case, and no repetitions or alternatives are evaluated in it.).

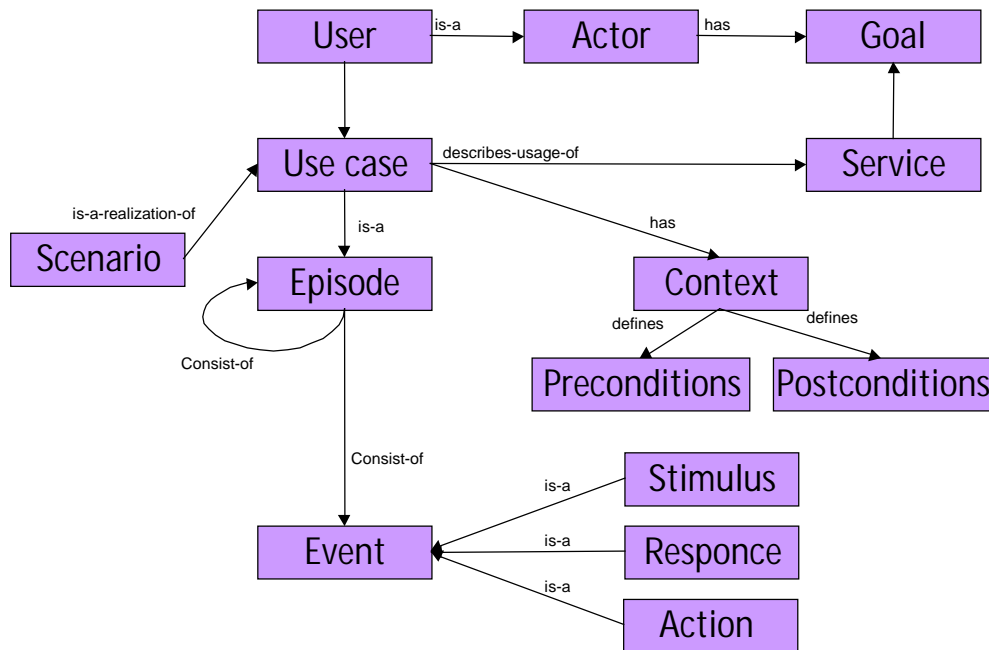


Figure 7. Use case concepts (Regnell, 1999.)

3.2 Contents of a use case

What criteria should we have when creating and describing use cases? The designer faces the problem of granularity: how large should the scope of one use case be, and to what level of detail should the use cases be modelled? The introduction of goals and services is the first step to define the conceptual framework. The abstraction level of the used terminology dictates the abstraction level of the use case (Holzblatt and Beyer, 1998). The scope of one use case is at minimum to cover how one goal is achieved or not achieved by using one

service (Regnell, 1999). If several use cases are realised simultaneously, the scope is extended to cover all these cases.

A scenario is a complete description of some specific instance; the use case is not. That means that in a use case the actual small details are filled in only to the level where it is possible to avoid misunderstandings (Jacobson, Booch and Rumbaugh, 1998), nor any particular individual actor is used. The use case, unlike a scenario, is a generalized workflow description of all individual users that would do the task (Kulak and Guiney, 2000). In realistic use case actors can do the task in the way described in the text and fill in the individual user's details.

Top-down and bottom-up approaches can be used when creating use case models. A top-down approach starts with eliciting a number of use cases that are refined in their structural properties in a series of iterations. Requirements are not considered as separate from each other. During the elicitation phase the individual requirements are evaluated and, if possible, put in the right place in the context (Sommerville and Kotonya, 1998). Task is divided into subtasks. The idea of a hierarchical task analysis (HTA) is to divide the larger tasks into more manageable subtasks (Sommerville and Kotonya, 1998). Also in this way the similarities between subtasks can be analyzed and subtasks combined. A bottom-up approach starts with concrete examples of scenarios that are generalized and synthesized into use cases. Instead of starting from the functional or use case approach requirements are gathered from story-like scenarios. These scenarios then will be abstracted into use cases. The two approaches complement each another and together with an appropriate conceptual framework they can be used in iterative design to produce more a

detailed design. These series of iterations help incorporating new knowledge in to the design.

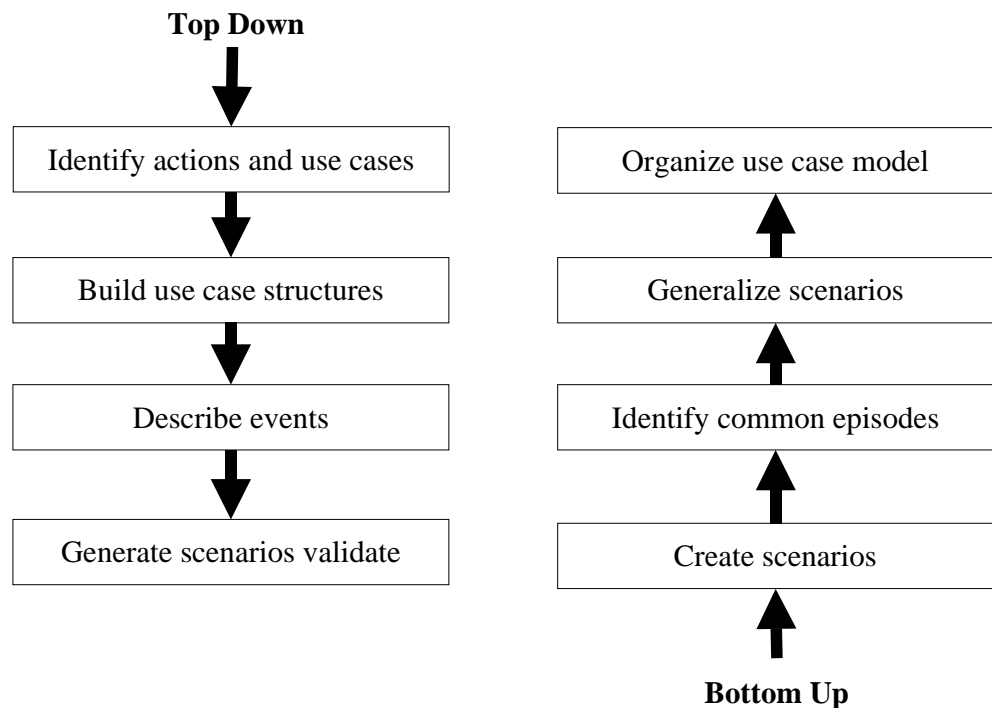


Figure 8. Creating use cases Top-down- bottom-up

3.3 Collecting the information

In the observation and research phase the observations and empirical information is collected. This includes identification of context and description of actors, their goals and actions using well-defined language (Holzblatt and Beyer, 1998). The focus of design governs the information analysis. In the analysis phase also the terminology is gradually extended and revised as more and more use cases are identified. Data dictionary is devised to unify terminology used in use cases (Regnell, 1999). This is especially important when many people are creating use cases. Uniform terminology can be enforced by continuous inspections (Regnell, 1999). Also the design documentation that relates to use cases, needs to have consistent terminology with use

cases. Consistent design terminology reduces misunderstanding and misinterpretation of requirements (Sommerville and Kotonya, 1998).

Context tells us how some specific task relates to other tasks (Whiteside and Wixon 1988). The context tells as close as possible to the ideal situation of being physically present and observing what actually goes on at the observed context of use. Communicating the user tasks and their interactions is the most basic requirement for a use case. The use case targets to describe the procedural experience in contrast to the summary of experience (Regnell, 1999). The details of a task make a difference. If people are asked to describe experience they highlight a few good and a few bad things, rather than explaining the experience in detail. This summary of experiences enables communicate prioritized highlights of actions intensively in a short time, but detail information is lost (Holzblatt and Beyer, 1998). The procedural experience descriptions are important for the use case, because they enable to see how details relate to each other, and what causes one task to be important in relation to other tasks (Kulak and Guiney, 2000).

Concrete data (vs. abstractions) is collected as a basis for a use case; this explicitly observable information is used with as little as possible interpretation or explanation of intentions (Holzblatt and Beyer, 1998). The action *per se* is described. All abstraction is avoided. The abstraction is in the observer's mind, not in the users context. Some other might interpret the user activity in a different way (including the user) and the interpretations are not the purpose of a use case. In human understanding the abstraction can help relate actions thus giving them a context and meaning outside the observed context of use. In usability, controversial to many other sciences, the meaning of action in larger contexts is not considered to be valuable information for the design (Holzblatt and Beyer, 1998). The focus is to observe action in its context describing the activity in concrete language as

much in detail as possible. Use cases are targeted to be a communication medium that each party can comment and interpret based on their own interests, but the language should not carry any interpretation. The interpretation and abstraction can be done in latter stages in design.

The focus of the design guides the use case documentation, in the same way as it guides the design (Holzblatt and Beyer, 1998). Information that helps the designer to come closer to his goals is relevant. The focus of design decides which information is to be collected. Any information that could be helpful to for achieving design goals is helpful. The focus helps to get closer to the goal, however, the design focus and the designers subjective focus should be kept separate. The designer biased conceptualizations and assumptions, if used as basis for focus, lead not to see the user assumptions and conceptualizations. During a customer interview or usability inquiry, design focus should be widened also to enable the user to alter the focus (Holzblatt and Beyer, 1998).

3.3.1 Analysis phase

The requirements for the design fall into two central categories: Functional and non-functional requirements (Sommerville and Kotonya, 1998). The functional requirement captures the nature of interaction in between the use case and its context of use. The non-functional requirement restricts the types of solutions one might consider.

The functional part of use case models the relevant internal states of the target system and the host system. The externalized conceptual model of the design helps designer to communicate with the user (Ehn and Löwgren, 1997). The conceptual model is incomplete if the context of use is not also modeled, this leads to the problem of

modeling (or restricting) the context of use accurately enough to reduce the complexity of the use case (Regnell, 1999).

The non-functional requirements such as reliability or human factors specify the formal manner (vs. natural description). Some constraints are not known at the initial design phase and are added during the design iterations. Different types of non-functional requirements include: Interface constrains, performance constrains, operating constrains, life cycle constrains, economic constrains and political constrains (Sommerville and Sawyer, 1997).

The first activity in analysis phase is to identify and describe actors and use cases. The second step is to unify the used terminology (Regnell, 1999). Especially if the design team consists of several persons, and the use cases are created by several persons, the data dictionary that harmonizes the used terminology is important. The terminology is gradually extended when new use cases are identified (Holzblatt and Beyer, 1998).

In order to avoid some typical problems with natural language the use case descriptions and user terminology should be revisited across several use cases. A systematic numbering of events supports traceability within and across different models of the analysis and syntethis phase (Regnell, 1999). Furthermore such conditions help making identification of identical conditions and sequences of events in other use cases. Also this is advised in order to make descriptions shorter, and to re-use same design pattern (Jacobson, Booch and Rumbaugh, 1998).

3.3.2 Synthesis phase

In the synthesis phase the use cases are formalized, and integrated, and the relation between the different use cases is defined. This phase

creates a synthesized model of the needed use cases. This use case model dictates the order of the high-level user goals (Regnell, 1999) This synthesized model is verified against user goals. The synthesis phase is iterative in nature and after series of iterations the correct and complete of use cases are available (Kulak and Guiney, 2000). In the synthesis of separate use cases the use case model is build.

In the formalization activity of the analysis phase the format of each use case is identified (Regnell, 1999). The result of this activity is a collection of use cases represented in a formal graphical language.(see examples Figure 15 and Figure 16) Relations between use cases describe the sub use cases or - in detail - events which the high level use case consists of. This formalized model is described in the use case diagram.

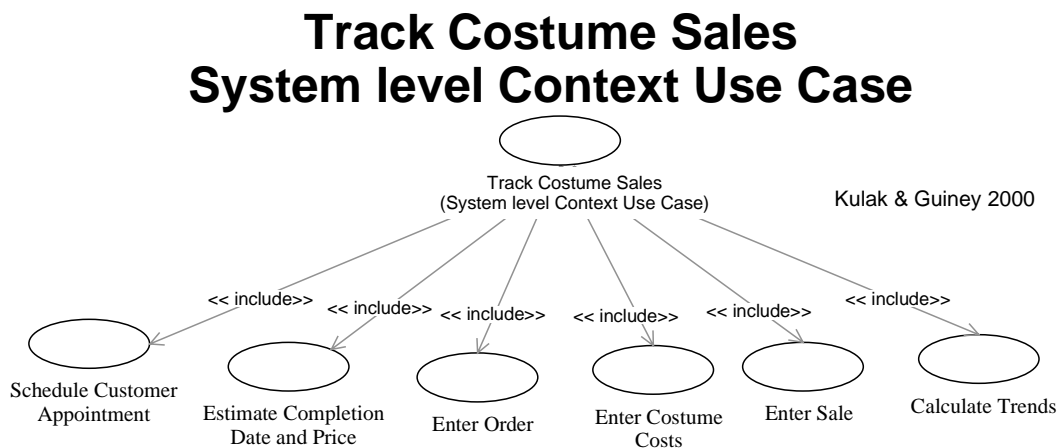


Figure 9. Use case diagram

The formalization activity enables the designer to identify the similar patterns across different use cases (Regnell, 1999). The similar patters, when recognized can then be realized using the same functionality, when the end product is constructed. From user point of view this harmonizes the user interface. Similar tasks are done similar order (or in the identical order if the realization is done only once using the same sub-component).

The use cases when analyzed and synthesized are then validated by the users. Validation is a step of design where the design solutions are tested to verify that that specific solution gives some additional value to the user or to the task the user is performing (Jacobson 1998). A representation of design in this phase consist of individual use cases, the synthesized use case models that are described using use case diagrams, and the critical workflow represented using the top level use cases. The three representations of system are to be verified and modified by the user and by the designer while the incremental software design process proceeds.

3.3.3 Complexity

When dealing with complex systems the structure of the use case model is critical. It is vital to have modularization constructs for dividing the use case model into manageable units (Jacobson, Booch and Rumbaugh, 1998) This reflects the modular construction of software by several designers. Another common way to deal with the complexity is the use of hierarchical models to refine the design concepts into more detail (Sommerville and Kotonya, 1998).

The level of abstraction can be different in different use cases. These levels of abstraction can be identified. At the **environment level** the environment is described by associating the related actors, services and goals to each other. At the **structure level** the episode structure of each use case is described with sequencing, alternatives, repetitions, exceptions and interruptions. At the **event level** the episodes are described in detail in terms of which events occur in each episode (Regnell, 1999).

In environmental level the use cases can be organized according to what service they describe (Regnell, 1999). A single usage of one service can be combined together into one manageable unit. The

concept of service is critical when deciding how to divide large use case models into subunits.

The concept of episode supports the reuse of use case parts, and thus avoids duplication of work. This will help in managing use case models and creating more a compact model.

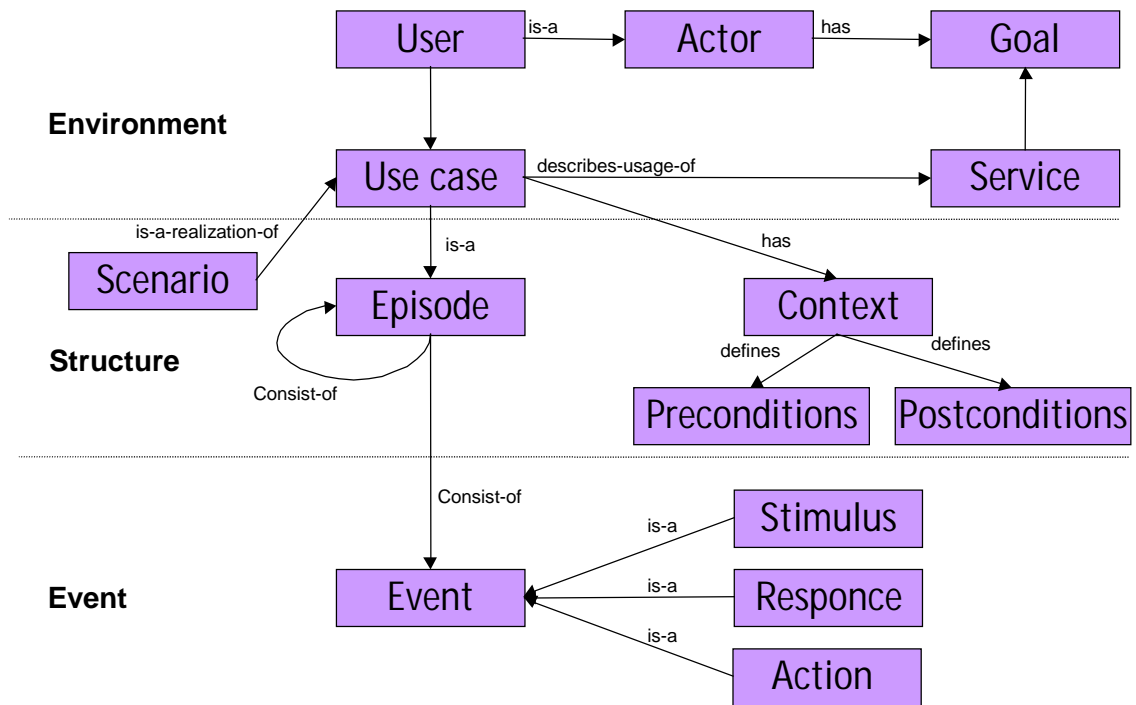


Figure 10. Concept relations and levels of abstraction (Regnell, 1999.,p 92)

The hierarchical division between environmental, structural and event level helps to manage complexity as abstraction level of use cases can be viewed. The hierarchical division provides an environmental overview as well as a detailed event level description (Regnell, 1999).

3.4 Use case template

Most of UML elements are diagrams. A use case is a page or two of text that each oval in use case diagram represents. A common

accepted template is needed in a project. An example of a use case template by Kulak (Kulak and Guiney, 2000):

Table 1. Example of use case template (Kulak 2000)

Field in the use case template	Description of contents
Name	The use case name provides unique identification.
Iteration	Iteration relates to the stage and completeness of use case. For example Kulak (Kulak and Guiney, 2000) defines four stages: façade, filled, focused and finished.
Summary	In one or two sentences the summary describes the interactions that occur in each use case.
Basic course of events	The main description of functionality. This part of a use case describes the steps that actors and system go through to accomplish the goal of this use case. The actor always takes the first step
Alternative paths	The basic course of events represents the "simple, correct path" through the use case. No errors or missteps occur.
Exception paths	The alternative paths describe the less-common paths that need to be taken into account. The uncommon situations. A single use case typically includes several types of alternatives. Each alternative should indicate which step in the basic course of events in the starting point.
Triggers	Exception paths show the interactions when errors happen. Triggers describe the entry criteria for the use case. They are a list of conditions that must be true when an actor begins a use case.
Assumptions	Here are documented issues that designer assumes that are true but might not be.
Preconditions	Preconditions are set of conditions that must be in place before the interaction can occur. They model the non-functional contextual requirements. The relation between use case and the real world
Postconditions	Postconditions like preconditions model the non-functional contextual requirements, and relation between use case and the real world. After the use case is finished the postconditions are satisfied.
Related business rules	Business rules are unwritten rules that dictate how a company conducts its business. This part of the use case allows documenting the business rules that relate to requirements in this use case (cultural models)
Author	Creator or modifier of this use case, (and in addition the history of use case.)
Date	Date when use case in each level of completeness was written Example: Façade complete: date Filled complete: date Focused complete: date Finished complete: date

4. CASE STUDY: IN DEVELOPING WCDMA RADIO NETWORK PLANNING TOOL

Contextual design and low-fi usability tests as well as user reviews were used to validate and refine use cases in Nokia Networks, Radio Network Tools software design. This chapter describes the use case method in practice. The method was used in designing two releases of Totem Vantage software product in Nokia Networks 1999/2000. The method was first applied in experimental practice when Totem Vantage 1.0 release was made. During the Totem Vantage 1.0 project the incremental and iterative design methodology was tested for the first time and several problems were faced in this transition period, when new design methodology was learned. The classical waterfall model milestones and project management - tracing the use case status and content of increments - as well as integrating system testing into incremental methodology caused severe problems. The incremental and iterative design methodology was later used more successfully when Totem vantage 1.1 release was designed. At that time design team had learned the new methodology, also the project was managed using use case realisation matrix Table 8, that helped tracking the status of the project as well as coordinated work between team members. The use case realisation matrix was used to plan the contents of each increment, while the project proceeded.

The first part of the case study describes the usability activities in different stages of these two software projects. The second part consists of scenario testing methodology and testing reports. The third part includes some samples of use cases, use case diagrams and workflows.

4.1 Usability in design project

In between the design phases usability activities were adjusted and combined into design activities in this project. Also the organisation adopted new work practices and a new design methodology. During the project several usability-testing methodologies were applied. Feedback from the first phase was used to integrate the usability activities into incremental software design. The exact design specifications as well as UI descriptions and Usability test results are left out due to the classified nature of those documents.

4.1.1 Usability during the software design project start-up phase

The objectives and targets of usability work were defined in a vague level at the beginning of the project. The software development organization was in competitive situation meeting tough requirements from customers to shorten the release cycles and at the same time to improve the usability of its' products. Also it met fierce competition from competitors. As trying to meet these challenges the incremental and iterative software development methodology was adopted. Usability resources were hired to improve the common understanding of the user needs and to increase the organizations competence, as well as enhance the internal communication within the design team and internal customers. The usability plan was made to met these goals, in practice the plan was made to manage usability work and activities during the design process, in order to achieve functionality that meets users' needs thus ensuring high quality for the designed software. The general usability plan worked as an agenda for user meetings.

The project plan included the idea of managing the software design using Rational Unified Process. The usability plan and activities had to be integrated into the new approach. How this was done in practice was left open in the first phase when designing the first release. It was

defined later in more detail when designing the second release when the competence and knowledge of the new working methods had developed. The software development methodologies were developed at organization as well; the project management saw usability as one method to increase the quality of the product as well as a method to define and communicate requirements into the design.

The hidden objective in the usability plan was to integrate usability methodologies into the product process, thus increasing software design process maturity, without creating a separate usability evaluation control process. Usability was part of the product design not restricted to product design evaluation, in the late stages of the design. The information produced in usability reviews was used to create alternate design decisions.

The usability plan included several activities that each added the design some new requirements at the stage of the design when these requirements could be worked on. Usability activities were carried out throughout the product lifecycle. Starting from the concept definition and ending with customer training and feedback sessions. Instead of field study the knowledge of user context was initially acquired by inviting constantly expert users to evaluate design proposals and to work with unfinished prototype after integration testing, thus detailed usability testing with product prototype were done after integration testing. However, even before any integrated product was available for testing, low-fi and paper prototype testing was carried out.

Due to incremental development practices, some parts of the design were designed in different times/parties (subconductors) and had been already well designed and usability inspected. Thus the usability activities were mainly concentrated on to develop the functions that were lacking detailed usability inspection and testing.

4.1.2 Product definition and design activities

4.1.2.1 Concept definition workshops



Photo 1 Concept workshop 17.1. 1999

Concept design workshops³ (Salminen, 1999) were held to create a shared vision of the concept core functionality. The one week long multidisciplinary session included users, core design team members and some system experts. The common understanding of the core functionality was reached using the "wallpaper technique" (Muller, Halswanter and Dayton, 1997). This workshop also gave preliminary ideas for UI design paradigms. The definition and specification of workflow - the network planning process was carried out using use cases. The result of concept design workshop was high level use cases (façade) of the core functionality of intended product (Kulak and Guiney, 2000).

³ Note the term "consept", in concept design workshop, refers to high-level definition (restriction) of the intended workflow of that is carried out using the system. In this it is quite different from term "consept" in scientific literature.

4.1.2.2 Refining and updating use cases



Photo 2 high-level use cases from concept workshop

The concept workshop created several high-level use cases. These high-level use cases were re-defined by domain experts (experienced designers with knowledge of user domain and a system expert). Experienced network planners (user, of intended product was a **radio network planner**) validated the re-defined (filled) use cases (Kulak and Guiney, 2000). When the technology grew more mature these use cases were updated. They also were collected and kept in one document that also represented the structure of intended workflow.

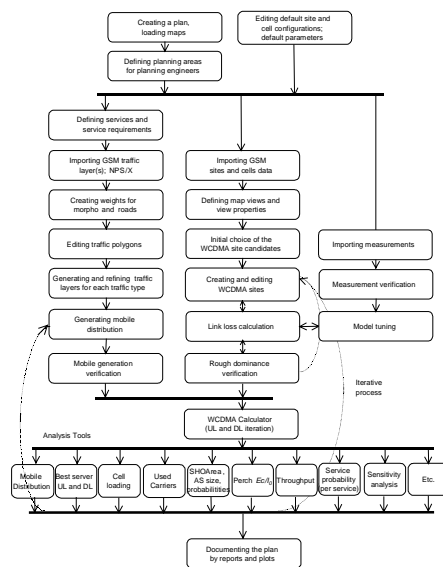


Figure 11 consolidated workflow (Salminen, 1999)

4.1.2.3 Refining task analysis

The filled use cases did not include all information needed for designing the user interface. Completed task analysis was made to fulfil the small details missed when designing individual use cases (Sheridan, 1997). In addition, the user interface design had some requirements that were independent from the workflow that the use cases were describing. The software tool was intended to be used in laptop computers. The output of the refined task analysis was a UI demo done using Visual Basic⁴. This prototype was re-defined (iterated) until an acceptable level and design was found. In parallel to specifying the UI, network domain model of the real tool was specified. The network domain model described the core concepts and their relations. From the usability point of view it defined the data dictionary. Concepts used in use cases had to be the same that were used in the network domain model. Refining the core concepts at this stage helped to ensure that designers started using the same vocabulary that the users were using (interpretation of these words was be different causing some problems, but this ambiguity of language approach is left out from this work). The network domain model, when validated, was core document; the domain experts' knowledge was modelled there.

4.1.2.4 Participatory capacity & traffic UI design

Some core design team members did not participate the concept workshop. They were responsible for capacity and traffic subsystem design. This part of the design was re-designed and this rough level design needed to be refined in detail in co-operation with experienced users (radio network planners) and core design team members.

⁴ Note paper prototypes existed already as a result of teamwork in concept workshop. The Visual Basic demo was based on those.

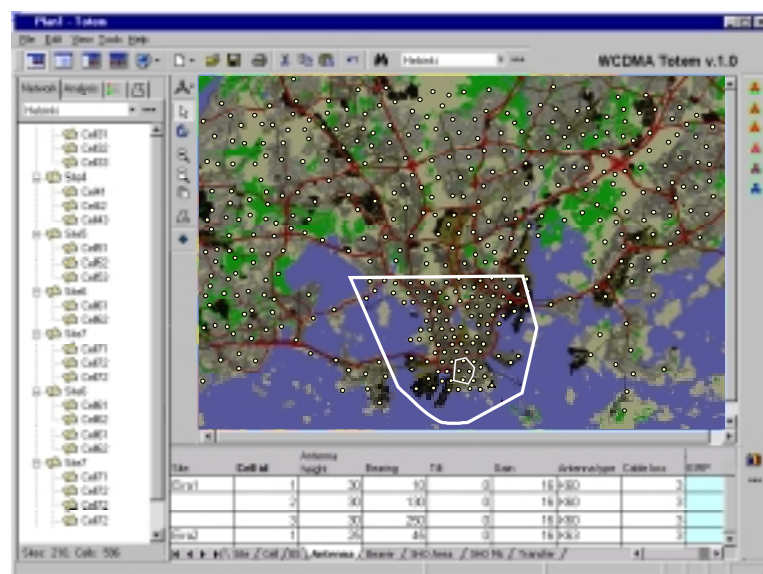
4.1.2.5 Main concept UI specification



Photo 3 Main UI specification in concept workshop

The main concept UI specification served as a document to communicate the design target to design team. In Totem Vantage 1.0 project no use cases were used for this purpose in a serious manner, the high level use cases as a workflow described the intended steps of functionality. The core UI specification communicated the UI design to relevant parties, and thus it was updated constantly

The UI specification dictated the form of the visual interface, while use cases dictated the functionality. In the Totem Vantage 1.1 project Main UI concept was already quite finalised and only subsystem UI specifications were designed.



Picture 3 Totem Vantage main UI prototype

4.2 Scenario testing



Photo 4 Usability testing

Several usability tests and cognitive walkthroughs were performed during the project. The data is presented in a format of a clip from a UI test report. The intended workflow is presented. Critical use cases were inspected in more detail, while the less critical use cases were to be tested later. The problems found when running the test scenarios are listed. Also some proposed design solutions found are presented in Table 3. Unsuccessful events in test scenarios and corrective actions. However each design problem and solution are dependent of the context of use in which they occur. The presented design problems and solution suggestions are general level examples of found deficiencies.

The actual UI test settings consist of various types from semi-informal user interviews to formal UI testing, including a test plan and video recording. In Table 3 the relation between the design problem found in the unsuccessful test scenario and a use case is shown.

Usability inspection and testing activities:

- Expert evaluations, UI cognitive walkthrough and heuristic evaluation
- Low-fi usability testing (by using paper mock ups and Visual Basic prototypes)
- Specifying usability evaluation scenarios and usability test tasks
- Usability testing with a real and integrated tool (video recorded)

4.2.1 Method

The major testing activities are summarized below:

- Users and their workflow are initially modeled. (This workflow comes from concept definition workshop)
- Test scenarios for each use case are created in cooperation with user
- Test scenarios are present to user(s) in usability tests
- Unsuccessful events in test scenarios are collected
- Use cases are defined more in detail or the designed workflow is changed. If design already exists it is corrected.

4.2.1.1 User

When identifying users they can be in different types, called actors. A user is thus an instance of an actor. An actor represents a set of actors that have common characteristics in respect of how they use target system. In this study only one actor is used and identified. The user is identified as: **a radio network planner**, a mobile radio air-interface professional with university degree of radio technology as well as several years of experience on the practical design of cellular (GSM) networks. The WCDMA is new technology for them. They have the theoretical background knowledge, but no experience of 3G networks.

4.2.1.2 The critical workflow path

The design had an intended critical workflow path of how to use the system. This was initially created in the concept design phase of the design. This workflow modelled the core use cases, and placed them in order to model the critical functionality of the intended use of system. The critical path describes the design functionality that is vital for the intended use of the product. All the functionality is on the critical path; if the design fails to produce good solutions for user tasks on the critical path, the overall performance of the product suffers. The critical path goes through the main functionality of the product. The Totem Vantage 1.0 critical workflow path is presented here.

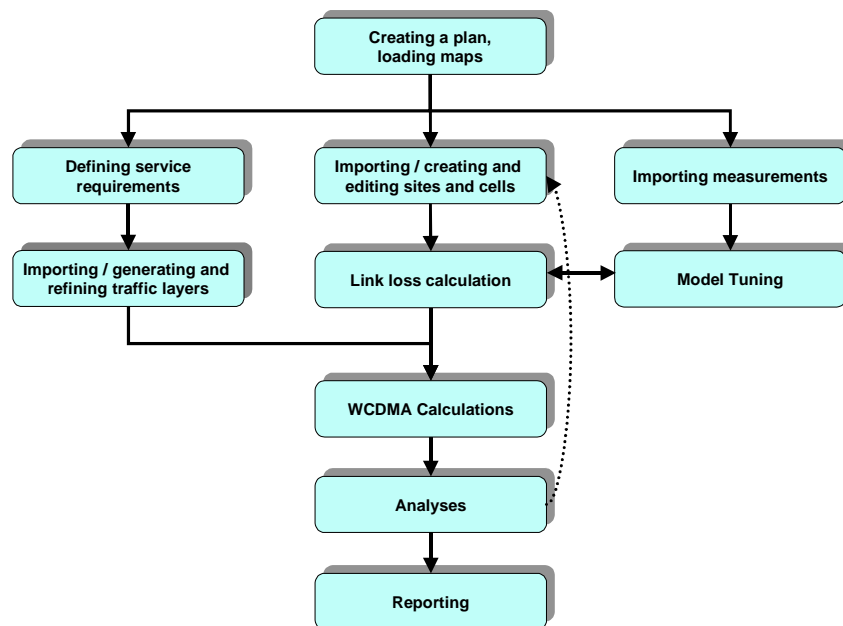


Figure 12. The critical workflow path of this study and the use cases that are included there

When the scenario testing was being organised parts of certain core functionality could be taken separately into focus and tested. One user testing session could, for example, test the traffic modelling path of the intended workflow. Also when the design was changed these new parts of workflow could be validated and verified with users. The parts of the workflow could be tested (run) independently.

4.2.1.3 Use cases and test scenarios

In each use case the actor has a set of goals. Goals are objectives that users have. When an objective has been met, the user has accomplished the use case, and can move on to accomplish some other use case with other an objective. In a use case the actor uses services of the target system in order to accomplish his goal(s). The services are the functionality the design offers. The goals are those tasks the user has in context of use – the subtasks of the task the system is created for.

Table 2. Test scenarios shows example of inspected core use cases and test scenario questions that were created to test the use cases. However pre-generated test scenarios were only used when number of users were performing the same task. When testing with only one user, user generated the scenarios; thus he incorporated the contextual information into evaluation situation.

Table 2. Test scenarios

<u>Use case</u>	<u>Test scenarios</u>
Introduction	Welcome Totem Vantage pilot user. You find yourself in Tokyo, and your task is to deliver an initial WCDMA plan that covers the downtown area. Network planning tool has been installed to your laptop and the maps are available on CD-ROM. Network dimensioning has been done already.
Manage maps	<p>The first task is to create the basis for a plan by setting up the map properties. You have to set the map in a directory - map directory Also you need to set up the map and projection specific parameters and create the visual appearance for your map.</p> <p>The Maps are found from the CD. There is special application outside planning tool called "admin tool.exe", that is for setting up map properties.</p>
Create plan	Map has been created congratulations, now you have to make a new plan. From RFQ you get the bandwidth, operators name etc. Give your plan the needed properties.
Create site templates	Easy way to create sites is first define some default templates that can be used later to create actual sites. Find out what are the available hardcoded templates. Edit one that fits into your purpose.

Set Propagation Model parameters	<p>Create template called Tokyo WCDMA site, that has 3 cells, Antenna heights, azimuths, etc... you get from RFQ. Set some antennas and propagation model to cells</p> <p>Measurements for Editing propagation model are not yet available, you have to live with current situation and set some propagation model parameters that are likely to give correct results.</p> <p>Edit the propagation model until you are happy with values, save your values</p>
Create sites	<p>All set for laying out the sites, you are planing for specific area and no site locations are fixed, you have to determine the site locations for this initial phase of network plan.</p> <p>Make sites using default templates. You can lay out and delete sites until you are satisfied with first initial plan layout.</p>
Calculate link loss	<p>Verify that sites are at least giving enough coverage for planned area. This is also used as basis for WCDMA calculations</p> <p>If not happy with the LLOS results make some changes in network configuration and calculate LLOS again.</p>
Define Bearer Services	<p>The WCDMA network is able to provide different types of service, each taking a different amount of resources. In this stage you have to define how are the end-users using network resources</p> <p>Create 3 different service types (from, RFQ) 144 kbit/s packed data service, 64 kbit/s data service and 10 kbit/s voice service...</p>
Create traffic layers	<p>Before you can allocate any traffic to any area you have to define the maximum area traffic can be distributed, this is done using the initialise traffic layers dialogue.</p>
Distribute Traffic	<p>Now create the areas where your subscribers are, look at your RFQ set the areas (using rectangles) and give the subscriber number for each service at the area</p>
Generate mobile distribution	<p>Traffic information is given to WCDMA calculator using a "snapshot" of situation in a network, this means that some mobile terminals are randomly generated in the map. WCDMA-calculations are performed using this "snapshot" of a network situation. Generate the mobiles on the map and then view where are the mobiles located.</p>
WCDMA-Calculation	<p>Run the WCDMA-calculation for the area you are interested, take a cup of coffee. This might take a while.</p>
Analyse results	<p>Verify the network quality. Locate the weak spots of the plan and make some analysis of what might be the cause.</p>

Mass editing	Edit the perch pilot power parameter and default antenna height for all sites using the network browser.
Produce Plots&Tables	When you are ready with a plan print a plot of the problem areas using colour printer, use legend that has the name of the plan and problem identified.

4.2.2 Results

A test scenario is a specific realisation of a use case in a test situation. A test scenario can be either successful or unsuccessful accomplishment of one or more goals. A use case can cover an unlimited number of scenarios as it may include alternatives and repetitions. A scenario is a specific realisation of a use case.

Table 3. Unsuccessful events in test scenarios and corrective actions shows events in unsuccessful test scenarios found during usability tests.

Table 3. Unsuccessful events in test scenarios and corrective actions

Use case/ problems found when running test scenarios	Design problem type	Design solution type
<p>Importing site candidates</p> <p>According to test subject the Importing wizard idea is good. The UI should give a hint that duplicate site names are not allowed. Planners wondered what would happen in case those two sites will have the same location.</p> <p>Suggested solution was to show a confirmation dialogue for the user.</p>	<p>Service needed</p> <p>Event found in use case context</p>	
<p>Creating and editing site templates</p> <p>The content of the templates were refined during the first four test sessions, so only the last test subject had possibility to carry out this task. He was very satisfied to the design. The only problem was to "finding where the templates are". It is suggested that the name of the "insert site" –operation is to be renamed in a way that it will give also the hint of using the templates.</p>	<p>Navigation problem</p>	
<p>Creating and editing sites from the templates</p> <p>All test subjects find the concept of templates very useful and powerful feature. Especially the possibility to create several sites at the time by assigning imported site candidates and the parameters taken from the templates were found to excelled idea. Also the support of the</p>		<p>User accepted new workflow</p>

<p>multiple selection in free order of when selecting the candidates and the templates was convenient and efficient functionality.</p> <p>Adjusting map views All test subject were very satisfied to following features: panning, scrolling, activating map layer and saving the most used views as favorites. However combined zoom back and zoom in operation was not appreciated. Especially navigating to previous view after panning operation was found to be unintuitive. It is suggested to have separate "go to previous view" operation and toolbar button. Also more direct map control like "zoom home" and zoom in by factor x is needed to increase the efficiency of map controlling. These operations could be located in map pop-up menu.</p> <p>During polygon drawing activity, the users should have the control over the view. The idea of automatic scrolling when the cursor is moved near the edge on the map. However the functionality could benefit being more "aggressive"</p> <p>Polygon control Only the last test subjects had the possibility to test this functionality by prototype. The consistency with functionality of the MS PowerPoint was appreciated Editing functionality of the polygon was regarded as important due to iterative nature of the traffic modeling. The most important finding was the lack of control when defining overlapping polygon interaction.</p> <ol style="list-style-type: none"> 1) Traffic need to be set in area surrounding polygon(s) excluding inner polygon(s) from that area. 2) Drawing polygons having a shared borderline. (In practice case kehä 1) using the first-drawn polygon line when drawing another. 3) Spread subscriber number into several active polygons. Add separate polygons to give them common properties (Spread known subscriber number [2000 users] for several selected polygons <p>Traffic modeling use cases/ Defining service types Defining service types as specified in REQ in given data transfer rate was consistent with user expectations. However when user defined mobile service type he considered missing the mobile terminal sensitivity controls. There was discussion about data rate (bitrate), mobile speed (km/h) and mobile terminal type (receiver sensitivity/transmit power of mobile) determining subscriber profile. First stage to model traffic is to define these settings. TotemW gave an easy access defining service types when they were exclusive. The inclusive service type case, when mobile type uses two</p>	<p>Navigation problem Event found in use case context</p> <p>Service needed</p> <p>Consistency with commonly used software</p> <p>Event found in use case context Conflict with use case and context of use</p> <p>Service needed</p> <p>Conflict with use case and context of use</p> <p>Conflict with use case and</p>	<p>Scenario</p> <p>Scenario</p> <p>Scenario</p>
---	--	---

<p>service types, was not considered.</p> <p>Traffic modeling use cases/ Spreading traffic Precondition for traffic spreading is given traffic REQ maps from the customer. These can be based in four different formats: 1) Different service areas specified (by drawing on map) and subscriber population given for each area. 2) Census based traffic distribution (subscriber number/squarekilometer) 3) Cell load is given from existing network and cell dominance based traffic distribution is allocated to map 4) Grid with subscriber population in each area is given. Grid might have different resolution in downtown and rural.</p> <p>In usability inspections the format 1) was selected. Several polygons were drawn each defining one area given in REQ and subscriber population in that area. Some polygons where overlapping and some sharing same borderline. All defined service areas were drawn on the map with polygons.</p>	<p>context of use</p>	<p>Scenario</p> <p>Scenario</p> <p>Scenario</p> <p>Scenario</p> <p>Selected scenario</p>
<p>TOTEM VANTAGE 1.0 USER INTERFACE INSPECTION AND REFINEMENT 4.2.2000</p> <p>Coverage display Currently no way to show LLOS raster. Proposed solution each Site/Cell pop up menu has Display coverage option, which is enabled when LLOS calculation has been, done for that site/cell. Otherwise this option is disabled. Composite coverage (displaying several cells at same time so that strongest signal strength/pixel is shown) sites are selected – analysis menu display composite coverage When once done can be selected from map layer control box and from view menu.</p> <p>Close/delete plan options Close option will be added to File menu after open plan option. Confirmation dialog is needed. Delete option shall be added in later releases after open / save as option in file menu</p> <p>Propagation model editor Currently propagation model cannot be edited.</p> <p>Area control Areas cannot be controlled and they are not listed anywhere. Seems like navigator won't have polygon browser in 1.0 and some</p>	<p>Workflow</p> <p>Service needed</p> <p>Workflow</p> <p>Service needed</p> <p>Service needed</p> <p>Service needed</p> <p>Service needed</p> <p>Conflict with use case and</p>	

Case study: in developing WCDMA radio network planning tool

<p>workaround method has to be implemented. Areas should be named (default names Polygon1, polygon2 ... area1, area 2...) and areas should have hide/show delete rename select... options.</p> <p>Areas will have to be listed in some dialogue that they can be selected</p> <p>Major issue considering polygons is traffic modeling: Is there a traffic modeler that is able to distribute subscribers at areas (user-defined polygons) and combine that traffic information into traffic layer</p> <p>If distributing subscribers is not supported areas are only used for selection purposes, and then area management is not considered as critical as in a case that polygons are used for distributing traffic.</p> <p>Temporary suggestion was floating window that would be opened from toolbar.</p> <p>Status bar</p> <p>At least coordinates, terrain height and terrain type will be shown when user clicks a location at the map</p>	<p>context of use</p> <p>Service needed</p> <p>Service needed</p>	<p>Question</p> <p>Design Suggestion</p> <p>Design Suggestion</p>
<p>TOTEM VANTAGE 1.0 USABILITY IMPROVEMENTS AFTER E3</p>		
<p>Performance of the basic functionality</p> <p>Performance of Inserting, moving and selecting objects on the map is not good enough. Response time should be 0.5 s instead of the seconds.</p> <p>Anyway the response times has been improved a lot last two weeks. Good work! We are close to acceptable UI performance. However selection of the objects on the map has to work much faster it does today (12.3.)</p>	<p>Too long response time</p> <p>Too long response time</p>	<p>Non functional problem</p>
<p>Traffic modeling</p> <ol style="list-style-type: none"> 1. Traffic modeling initialization (e.g. rectangle support for initialization or automated initialization, directory settings moved to e.g. admin tool), fixed partly 2. Indication of the number of the mobiles inside the analyzed area (a dialog will be added before calculation are made, design made, implementation done) 3. Explicit and Consistent UI indicating the purpose of the each dialog (e.g., Distribute Traffic, Generate Mobiles. Add one / two sentence hint of modeling purpose for every traffic modeling dialog. 	<p>Conflict in use case</p> <p>Service needed</p> <p>Visual design</p>	
<p>Bearer Services:</p> <p>Two tabs</p> <p>Content of the general tab: Name, Description,</p>	<p>Too much information in one view</p> <p>Conflict in</p>	

Case study: in developing WCDMA radio network planning tool

<p>Type, Bitrate</p> <p>Content of the "Traffic" tab: Activity, Traffic / Subscribers</p> <p>Add Tip!</p> <p>Traffic Layers: Rename it "Traffic Distribution" or " Model Traffic for Areas"</p> <p>Areas name should be shown on the dialog.</p> <p>Columns: Subscribers inside area, Subscribers / Km2, Traffic, Traffic / km2</p> <p>Only numeric data input allowed</p> <p>Tip! Enter or edit number of subscribers for each service inside selected target area. Traffic is calculated from user activity data</p> <p>Buttons: "Ok", "Apply" and Cancel if the dialog will stay modal.</p> <p>However the dialog should work in the floating mode, which allows the user to draw the polygons (direct manipulation)</p> <p>Provide feedback on the map (subscribers / km2)</p> <p>.</p> <p>Mobile list generation:</p> <p>Rename it : "Generate Mobiles"</p> <p>Add column "Scale"</p> <p>Make Traffic / subscriber column not editable.</p> <p>Tip! Scale factor can be used for increasing traffic inside selected area. (Phases, Special hot spots)</p> <p>4. Document Traffic modeling & Analysis process strategies. The parameters should be documented more detailed.</p> <p>5. Traffic modeling by using include / exclude polygons</p> <p>Suggestion for Edit Traffic dialog layout (floating).</p> <p>Map objects</p> <p>1. The selection of the area objects should be indicated clearer (e.g. increases border line width and reversed line color)</p> <p>2. Selection indication color inconsistent between site and area selection</p> <p>3. There should be an easy and efficient option for showing the road vectors above and under coverage raster and analysis (map component ok, UI missing)</p> <p>4. The site icon and the label should have maximum size</p> <p>5. The labels should be hidden automatically when these can't be read anymore after zoom out</p> <p>6. The selection of the cells should be should be disabled after certain zoom factor for avoiding user errors (site selection instead)</p> <p>7. Rectangle and polygon border width should</p>	<p>terminology</p> <p>Conflict in terminology</p> <p>Service needed</p> <p>Conflict in terminology</p> <p>Service needed</p> <p>Conflict in terminology</p> <p>Errors made</p> <p>Service needed</p> <p>Service needed</p> <p>Workflow</p> <p>Service needed</p> <p>Conflict in terminology</p> <p>Service needed</p> <p>Errors made</p> <p>Service needed</p> <p>Design information missing from use case</p> <p>New use case</p> <p>Visual design</p> <p>Visual design</p> <p>Conflict in visual design</p> <p>Service needed</p> <p>Errors made</p> <p>Visual design</p> <p>Errors made</p> <p>Difficult</p>	
---	---	--

<p>be broader</p> <p>8. Normal mouse icon during selection</p> <p>9. Right clicking should also select the object (site, cell, polygon and rectangle), before opening the pop-up menu</p> <p>10. Selection of the site should be possible also from the site label (label color inverted too).</p> <p>Pop-up menus of the map</p> <p>1. The site pop-up menu's first row should indicate the selected site(s)</p> <p>2. All cells related to the site should be listed at the end of the site menu</p> <p>3. The cell pop-up menu's first row should indicate the selected cell(s)</p> <p>4. Map property pop-up menu is missing</p> <p>Main Menus</p> <p>1. "Save As" missing</p> <p>2. Remove following menu items: View>Rotate View>Scale Traffic Insert>Antenna, Insert>Cell Tools>Parameter template editor? Tools>Site Template editor?</p> <p>2. Remove analysis icons</p> <p>Disabled menus</p> <p>1. Disable menus which are not relevant for selected object, toolbar too</p> <p>2. Add dialog, which will tell which task need to be made before certain mode e.g. If you select the best server analysis without traffic modeling done properly, you will be informed.</p> <p>Main toolbar</p> <p>1. Remove Find icon</p> <p>2. Redesign window management icons</p> <p>Antenna Editor</p> <p>1. New layout design e.g. "Properties" button should locate near antenna list</p> <p>2. Remove unnecessary frames</p> <p>3. Antenna properties should be splitted by two tabs</p> <p>4. Dblclick should perform the properties-operation</p> <p>5. Attachment icons are not needed</p> <p>6. Make properties window modal dialog</p> <p>7. Remove Menus and Toolbar</p>	<p>selection</p> <p>Conflict in visual design</p> <p>Consistency with commonly used software</p> <p>Service needed</p> <p>Visual design</p> <p>Service needed</p> <p>Visual design</p> <p>Service needed</p> <p>Service needed</p> <p>Errors made</p> <p>Visual design</p> <p>Errors made</p> <p>Service needed</p> <p>Visual design</p> <p>Visual design</p> <p>Navigation problem</p> <p>Visual design</p> <p>Too much information in one view</p> <p>Service needed</p> <p>Workflow</p> <p>Visual design</p>	
--	---	--

<p>Browser</p> <ol style="list-style-type: none"> 1. Enter-key acceptance of the editing should move the focus to the next row, not to the next column (like Excel, e.g. very useful for site layout design) 2. Column headers without bold font 3. The first mouse click to the cell should select the cell, the second activates the editing mode 4. Right clicking should accept entered value and set focus to table cell 5. Warning message is needed if name already exists for renaming operation 6. Dialog for editing the columns content needs a new layout. 7. Error message "Invalid parameter value" should be replaced by smart tip 	<p>Service needed</p> <p>Visual design</p> <p>Service needed</p> <p>Service needed</p> <p>Errors made</p> <p>Visual design</p> <p>Visual design</p>	
<p>Window management</p> <ol style="list-style-type: none"> 1. Maximize / restore map (Legend and Navigator frame should remain the same as before operation) 		
<p>Area management</p> <ol style="list-style-type: none"> 1. Simple floating window should replace modal dialog. Add button to the toolbar for opening it 2. All operations should be located in pop-up menu (consistent with browser) Also traffic related operations should be included in this menu, - the content of the menu need refinement. 3. Direct renaming should not be supported (renaming operation should be activated first). 4. Polygon and Rectangles should be named explicitly, not by general "Area" 	<p>Service needed</p> <p>Service needed</p> <p>Errors made</p> <p>Conflict in terminology</p>	
<p>Site Template and Creating Sites</p> <ol style="list-style-type: none"> 1. Inserting mode should stay active (creating multiple sites) 2. Creating planners own template should be more intuitive. e.g. allow direct editing, ask name for template after closing it. 	<p>Service needed</p> <p>Conflict with use case and context of use</p>	
<p>Adjusting map properties</p> <ol style="list-style-type: none"> 1. Adjusting information items of the map is missing. Absolutely needed 	<p>Conflict with use case and context of use</p>	
<p>Creating the maps</p> <ol style="list-style-type: none"> 1. Creating process is not intuitive for normal planner. This part has to be documented very well. 2. Defining morpho color set should use the same color panel than the threshold functionality does. 3. There could be a link from the new plan dialog to admin tool, but how the saving mechanism will then work. The whole process could be replaced by "Map Vizard" if the task 	<p>Consistency problem</p> <p>Errors made</p>	<p>Documentation needed</p>

<p>should be made by normal planner.</p>		
<p>Legend and threshold</p> <p>1. Apply or redraw operation is missing</p> <p>2. Threshold dialog functionality inadequate (negative values, inconvenient automatic updating functionality)</p> <p>3. Selection of map color set is missing</p>	<p>Service needed</p> <p>Service needed</p> <p>Service needed</p>	
<p>Analysis</p> <p>1. DL Load target should be UL load target.</p>	<p>Conflict in terminology</p>	
<p>USABILITY INSPECTION FOR TOTEM VANTAGE 1.1 BUILD 1 AND 2</p>		
<p>Define Bearer Services</p> <p>Content reviewed:</p> <p>e.g. Packet / Circuit switched => RT, NRT</p> <p>e.g. Asymmetric services modeled</p> <p>General tab :</p> <p>Voice check box unit should be removed</p> <p>Voice =>"Voice service"</p>	<p>Conflict in terminology</p> <p>Remove service</p> <p>Conflict in terminology</p>	
<p>Define Relative Weights</p> <p>Relative (current) weight should be calculated also in percentage values.</p> <p>After Enter keypress the focus should go next row</p>	<p>Event found in use case context</p> <p>Service needed</p>	
<p>Distribute Traffic on Roads</p> <p>Dirtribution between roads and morphos still unclear</p>	<p>Conflict with use case and context of use</p>	
<p>Export Data</p> <p>Export =>"Export..."</p>	<p>Conflict in terminology</p>	
<p>Filters</p> <p>Redesign recommended, change request generated;</p>	<p>Errors made</p>	<p>Redesign recommended, change request generated to redesign the subsystem</p>
<p>Other findings</p> <p>Last drawn polygon or rectangle should remain selected</p>		

Case study: in developing WCDMA radio network planning tool

<p>Navigator visual design recommended Map scale indicator; ruler</p> <p>Water morpho type should have an options to be drawn over analyses</p>	<p>Visual design Service needed Service needed</p>	
ROUTE TOOL USABILITY INSPECTION/ USABILITY INSPECTION FINDINGS		
<p>In general Route tool is still bit incomplete. Help is missing</p>	<p>Service needed</p>	
<p>User Manual missing for tuning strategies</p>		<p>Add documentation</p>
<p>Route tool doesn't use object action paradigm supported by Map or Navigator</p>	<p>Consistency problem</p>	
<p>-The user can't select cell from either from the Navigator or from the map</p>	<p>Service needed</p>	
<p>-Import measurement – dialog redesign is recommended</p>	<p>Visual design</p>	
<p>Cell, measurement and carrier combination could be indicated clearer</p>	<p>Visual design</p>	
<p>The Route Tool should tell which propagation model is used and tuned</p>	<p>Service needed</p>	
<p>As the default the graphs should display both predicted and measured graphs. User shouldn't have to click them on.</p>	<p>Service needed</p>	
<p>Based on current graph-scales decision making is almost impossible</p>		<p>Comment</p>
<p>-Result : y-axis should make greater difference (P-M)</p>	<p>Service needed</p>	
<p>-Statistic : P-M should have same scale</p>	<p>Visual design</p>	
<p>-Regression : what is x-axis</p>		
<p>On x-axis used morpho type should be displayed</p>		
<p>Tuning link loss parameters (morpho correction factors) should not need opening and closing several dialogs</p>	<p>Navigation problem</p>	
<p>more straight causal connection between changes made in link loss model and affects on the model tuning graphs</p>	<p>Service needed</p>	
<p>easier access for changing link loss parameters when measured and predicted graphs are displayed in Route Tool</p>	<p>Navigation problem</p>	
<p>Import file format example contains ground height – if imported no way to mass update ground height</p>	<p>Event found in use case context</p>	
<p>Duplicate found but tool doesn't tell which sites (name coordinate)were duplicates</p>	<p>Event found in use case context</p>	
<p>Hard to find specific area when several areas exist in plan</p>	<p>Event found in use case context</p>	
<p>Hard to remember service requirements etc for areas</p>	<p>Event found in use case context</p>	

Table 4. Found problem types

Design problem types	
Conflict in terminology	
Conflict with use case and context of use	
Consistency problem	
Consistency with commonly used software	
Design information missing from use case	
Difficult selection	
Errors made	
Event found in use case context	
Navigation problem	
Service needed	
Too long response time	
Too much information in one view	
Visual design	
Designed workflow	

4.3 Sample use cases and use case diagrams

Use case diagrams are used to show the relations between the use cases. The system level context use case describes the main services that the tool provides. The use case workflow describes the intended use of the product. The Example of the traffic modelling use case describes how one high level usecase consist of several usecases, and how during the design phase these relations change, new use cases emerge and the existing ones are worked in more detail.

4.3.1 System level context of use case

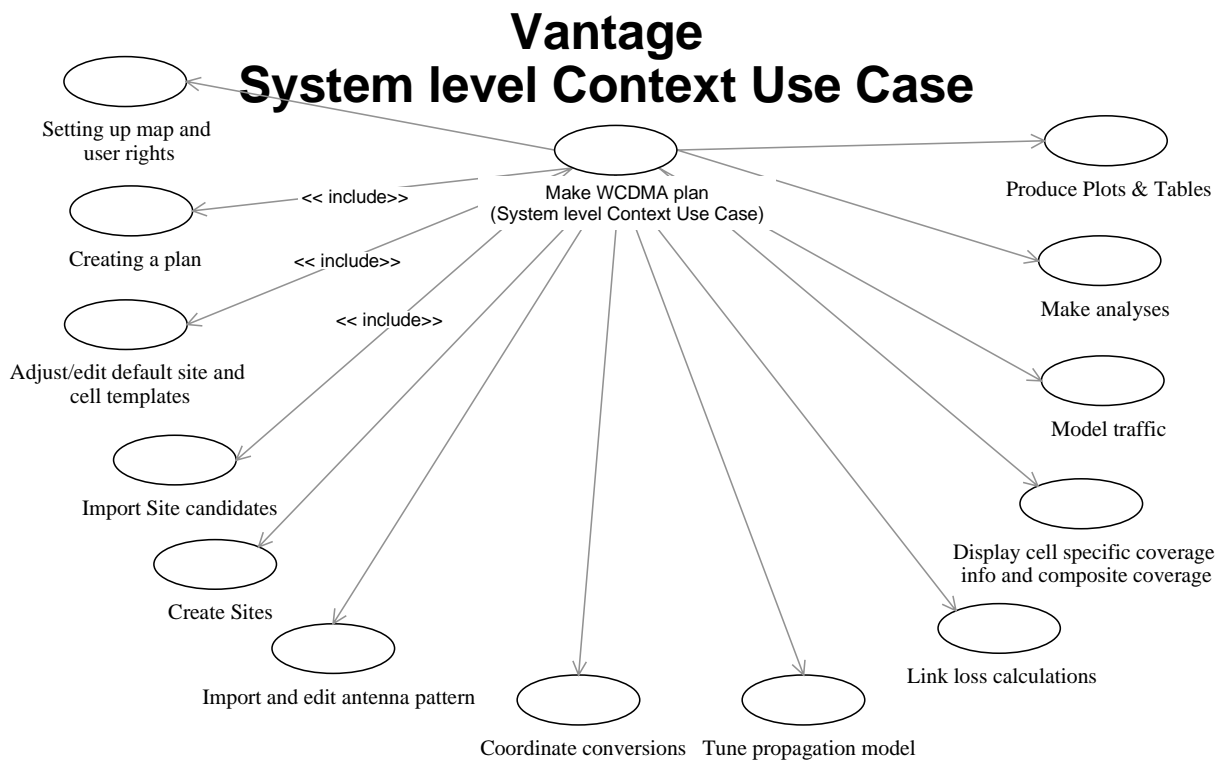


Figure 13. System level context use case

4.3.2 Sample of use case workflow path

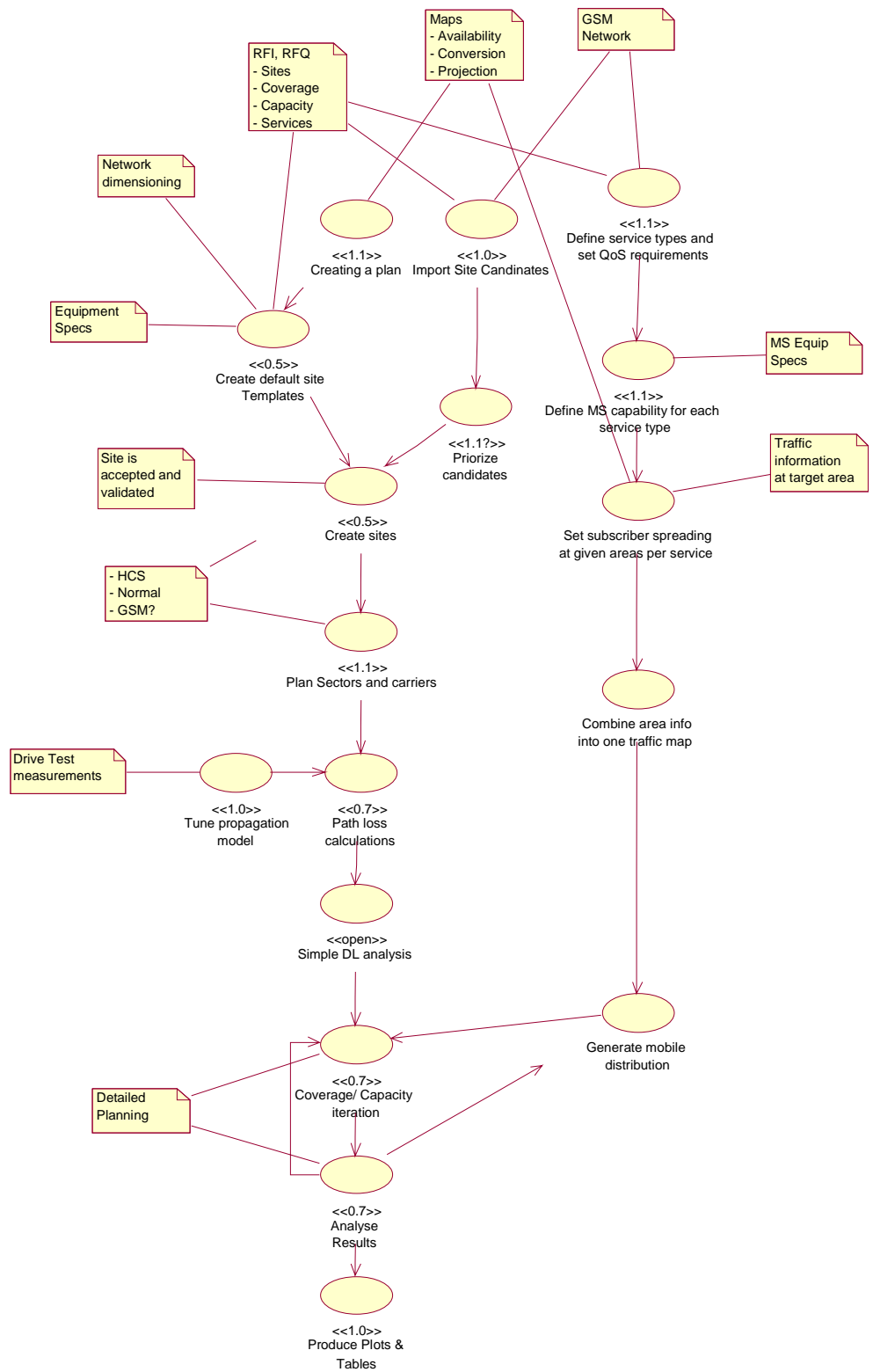


Figure 14. Use case workflow

4.3.3 Samples of traffic modelling use case diagrams

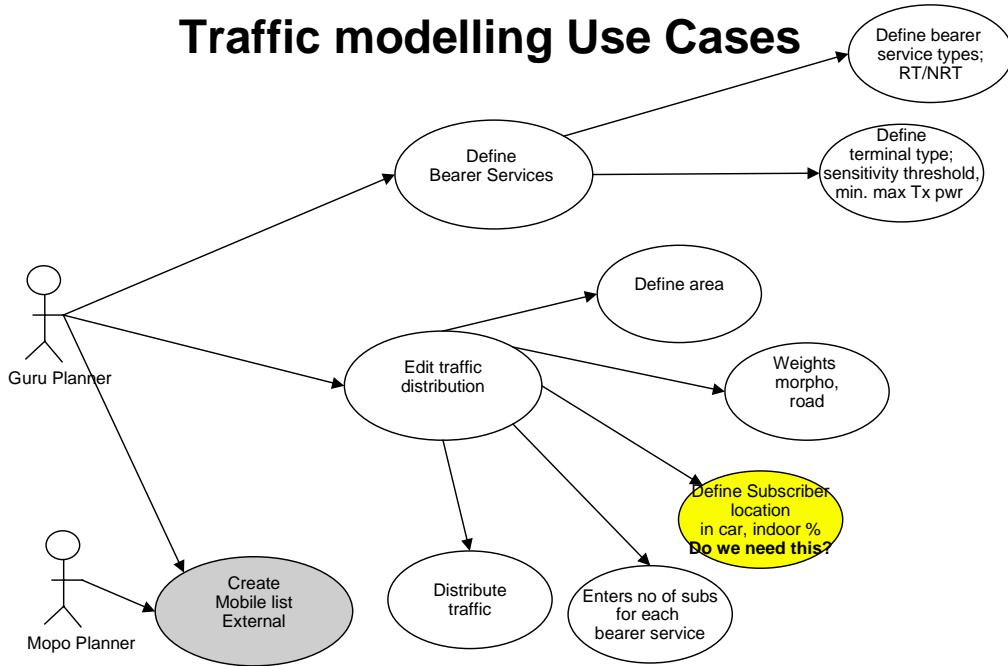


Figure 15. Traffic modelling usecase diagram 28.6.2000

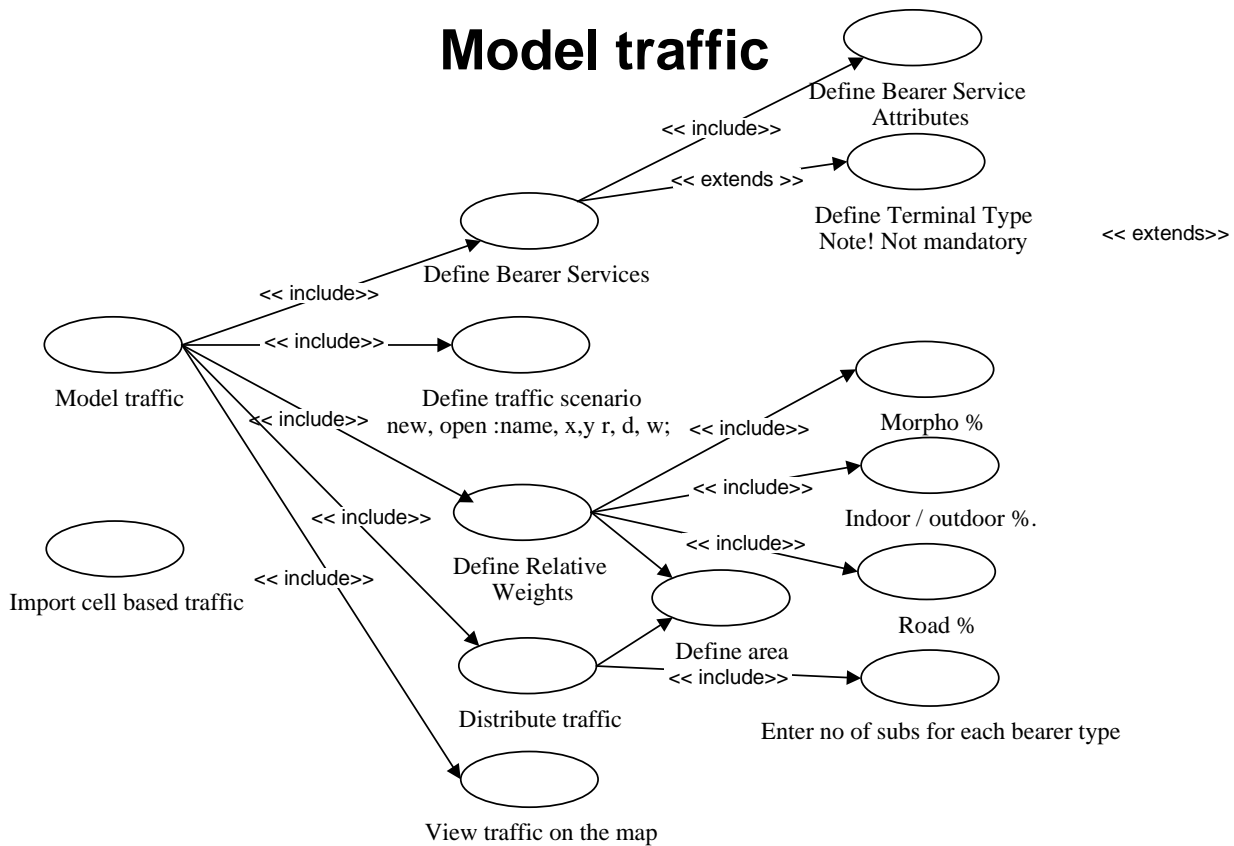


Figure 16. Traffic modelling usecase diagram 11.10.2000

4.3.4 Samples of traffic modelling use cases

Table 5. Prepare traffic density maps use case (concept workshop 13-18.1.1999)

Status	
Changes to E1	
Need /Goal	The planner prepares traffic densities for detailed coverage/capacity analysis. The traffic density information substantially improves the accuracy of the detailed analysis, and the WCDMA calculator needs the list of mobiles to work at all.
Frequency	
Usability reqs	
Context of use and user profile.	
Preconditions	
Description	<p><i>1. Input traffic parameters – This could be even hard coded in Totem 1.0</i></p> <p>The planner gives the traffic parameters for each traffic type. They are: traffic type name, bit rate, circuit switched or packet. He checks the data traffic model and accepts the default parameters.</p> <p><i>2. Edit "f1", save (Edit traffic area polygons)</i></p> <p>The planner loads the morpho map to be used as a basis. He draws the total traffic area polygon around the whole demo plan area. He gives the total traffic, for each traffic type (bearer service) for the total area polygon and the morphographic weights. Totem saves the polygon and the data.</p> <p>He takes the paper map with hand-drawn high density areas which have been done with the customer before, and draws a high density polygon on screen. He gives the total traffic, for each traffic type (bearer service) for the high density polygon and the morphographic weights. He draws the rest of the polygons and gives the traffic subtotals. Totem saves the polygons and the data.</p> <p><i>3. Start traffic density calculation [needed only if it is saved as raster]</i></p> <p>The planner starts the calculation. Totem performs the calculation. The planner saves the density rasters with a name.</p> <p><i>4. Start mobile list calculation [this child not mentioned in the workshop use case]</i></p> <p>The planner checks the total traffic of the density map. He gives the multipliers for the mobile files. He starts the mobile calculation. [Totem displays a progress meter ... very likely need]. Totem indicates when the calculation is ready.</p> <p><i>5. Validate: Mobile distribution, save</i></p> <p>The planner displays the mobiles on map. He saves the list as a file, and gives it a name.</p>
Exceptions	
Illustrations	
Postconditions	
Example	
Open questions	

Table 6 *Distribute Traffic use case (11.10.2000)*

Name	Distribute Traffic
Need	Traffic distribution is needed to store continuous, 'statistical' traffic distribution on basis, of which mobiles are located, when they are generated.
Version	0.1
Actor(s)	Planner
Frequency	Several times during a session, when traffic is distributed. At least once for each traffic scenario.
Preconditions	Bearer Services have been defined.
Description	<p>(1) Define an area for which there is input information about the quantity and distribution of traffic. To area definition belong such operations as select, deselect, exclude, and include.</p> <p>(2) Define morpho and road weights into weight layer within the selected area.</p> <p>(3) Give number of subscribers for each service.</p> <p>(4) Distributed traffic of selected bearer service (onto its layer.) Traffic of all bearer services can be distributed at a time and/or traffic of selected bearer(s) at a time.</p> <p>(5) View traffic on map (one by one).</p> <p>Alternative means for distributing traffic are for instance:</p> <ul style="list-style-type: none"> - percentage of morpho and/or road types - percentage within each polygon in a larger area - percentage on each bearer service
Exceptions	
Postconditions	Traffic layers have been generated MS lists can now be generated
Open issue	Indoor / outdoor percentage of traffic.

Table 7. Show Traffic on Map use case (11.10.2000)

Name	Show Traffic on Map
Need	Traffic distribution on map is useful to show to better enable positioning of sites into such areas, where there is also traffic. Visualisation is also needed to enable visual verification of distribution.
Version	0.1
Actor(s)	Planner
Frequency	In every session. Several times during a session. If possible, several time when a 'Distribute Traffic' dialog is open.
Preconditions	Scenarios, bearer services, traffic distribution must have been made.
Description	Show weight layer Show traffic layers(s) for each bearer service (nr subscribers/km2) Show UL/DL traffic with appropriate unit (mErl, kbit/s) Modify colour coding to be appropriate and show again.
Exceptions	
Postconditions	Aimed layer is shown on map.

Case study: in developing WCDMA radio network planning tool

Table 8 Sample of use case realisation matrix

Use Case	Use Case Status	Design + Implementation	Integration + Testing	Documentation (Tech Spec, UM)	SYTE status
Explore Single Analysis Results	Ok	ok		Test cases should be checked and updated if needed	Could be tested, browser views missing
Explore Statistical Analysis Results		Mobile: ongoing Sensitivity UI/Manager: ongoing Browser views: Peter and Markus will check the needed design, inform Risto WCDMACalc:		Test cases should be made	
Cell based traffic (Import cell traffic)	Ok	Import: UI about ok, import not ok Traffic: traffic distribution ongoing Browser: not started, requires further study	Import: ongoing Traffic: Browser:	Not started, tech spec exist	
WCDMA analysis with BTS capacity restrictions	Exist in BaseStationCapacity.doc	WCDMACalc: about ok WCDMAManager: - NWmodel: ok. DB: ok. CellUI: ok.	WCDMACalc: WCDMAManager:		
.					
.					
.					

5. DISCUSSION

Usability activities in this project were rather different from traditional usability evaluation and testing paradigm. The usability activities were more or less a constant interaction between the user and the user representative inside design team. The design was modified in a series of iterations, and when a new design was created the to design solutions were validated and modified to improve the quality of the end result. This was done by modifying the modeling language that dictated the functional part of the system, but rather than modifying the functionality *per se*, the interactions between the intended system and user were modeled. This was done in context of use, in environment where user was trying to accomplish his goals. The context of use could be made using paper prototypes or scenarios or using incomplete product as a prototype for piloting purposes. The main issue was that the feedback was collected from the user in naturalistic context, in a situation where the user could relate to the task. The functional specifications could not be reviewed in formal review meetings or unnaturalistic settings. The user did have to walk through the steps of intended workflow and fill in the missing details, as well as to criticize the missing or bad design that required user to perform tasks in order that was inconvenient.

5.1 Help the user to design - participatory design

The user will have the product when it is ready at the end of the development cycle. Most probably the user is using an earlier increment of the design, or has some experience of a similar product. The user will be the one that will work with the product, and he knows the work. The participatory design will use the user as an information source and help user to make decisions between the

possible alternatives. The design in co-operation with the intended user runs with a less risk of producing unsatisfied products also the commissioning phase is less sharp. Selected pilot users are using and evaluating the design at the very early stages and help incorporating knowledge of new design requirements at very early stages. If the incremental development methodology is used the increments can also be evaluated by selected pilot users. When the release is made and delivered for a larger user population it is thus already been verified several times by pilot users. And the design is to be modified to help user to perform his tasks. Participatory design makes the interaction between pilot users and (interaction) design personnel constant and frequent.

5.2 Completeness of design and usability testing

Use cases could be tested and modified even if no completed software design exists at the time. Design can be explicated by using use cases, or writing them out as more concrete scenarios or implementing (paper) prototypes. After usability tests, the suggested modifications could already be made as event descriptions in use cases. Thus the design can be changed even before any functional software implementation is available for usability testing. When the tested use cases become more realistic and also critical context knowledge was added into (specification of) the design. Test scenarios /prototypes provided the user a concrete representation of the intended product. The intended users are able to test the intended design solutions in the context of use. The workflow and detailed design decisions can be made in co-operation with the user. The design is made more accessible to comment, and if the intended workflow need to be changed for context of use requirements, the change requests can be rapidly taken into account and prioritized within design the project. If the development organization follows iterative and incremental design methodology the critical design solutions can be focused on at the

early stages of the design. And if some critical workflow is identified, the basic functionality can be prioritized and re-designed if needed, at an early stage, without wasting time and effort to completely implement the solution, just to be disregarded or discarded by the intended users.

The examples show how a task that is incompletely defined in the early stages becomes more and more detailed. The use case template that is in the first level missing many fields and the events are described in vague language become more and more detailed, also the terminology and steps come more distinguished. The structure of tasks do change and tasks are split into more detailed subtasks. These subtasks are organized optimally, this changes the use case diagram structure. The modeling language is flexible to allow such changes in design, but at the same time the user can relate to the tasks that the system is intended to perform.

5.3 Design environment

Use cases and use case models are methods to design, create, communicate and test design ideas between the developers, but also with users at every stage of the design process. Use cases, as a method of naturalistic task modeling - in contrast to formalistic modeling like technical specifications - help designers of software to communicate their design ideas into users using such language that is easily accessible without detailed technical understanding of the architecture or the existing design. Users can comment and modify use cases to help iteratively shape the design into more practical solution. Due to this iterative nature of the design the product in the end turns out to have a more usable design than without these iterations taking place between the end-user representative and the designer. The enhanced communication and early criticism helps the designer not to fall into the design "stake pit", that is, to spend time in designing a solution

that is not trivial, and might have several possible design solution candidates. Prioritizing the user needs in the design without proper feedback makes designing difficult, and if not solved makes using the product hard.

When the software product development environment is observed and we have quite a complex problem solving environment, several actors are interacting in problem solving:

- A) The user (end user of designed system)
- B) The designed software system.
- C) The developer of the software system
- D) Expert who has information for the system

End users use naturalistic problem-solving methods. Expert knowledge is based on education and formal rules of problem solving are applied if the adequate information and time is available. In general, end users only use formal decision making when it fits to a recognized problem coherent with educational knowledge.

A software system is created to ease users to perform some task. End user is not relying on the formal theory of the phenomena, rather his own interpretation of the problem. The software provides access to expert knowledge. The software system (User interface) gives hints for problem solving of methods that can be used in this problem case, leading to the basic perception of the problem. The end user is thus trying to solve his part according to naturalistic principles.

The software system can only rely on the information it has available. It has information that it processes in according with the user commands. Software system is a formal computing machine. According to given input a given general set of rules leads to next step that causes specified output for the user.

The developer of software system has a naturalistic problem: How to create a program that supports this specific problem. Thus his problem context is naturalistic but his methods are formal. This leads him to formulate the problem-solving domain in a formalistic way. Use cases are designed to re-formulate designers thinking, to model the user environment also in naturalistic way. Thus his design decisions can be easily communicated to end user that do not have access to formal problem domain.

In software development the basis for good design is well formulated requirements. Accurate requirements include a correct and detailed user problem solving environment description. Thus making critical the document that describes the user actions, motivation and means – that end user has to archive these goals. They describe as well the order of mental and physical actions. Although the designer is looking for set of fixed rules that help him formulate the basis for design, the context of use is more easily validated if the actions are described in naturalistic language. Such way less interpretation is made and the context is nor restricted.

One naturalistic problem can be solved in multiple ways; also according to different formalisms it can be formulated in multiple ways. In the software development environment the formalistic design tends to change rapidly according to time and design limitations. However, the user environment is to be changed in a much slower phase. The user contest of use changes when new artifacts (e.g. new software products are introduced) or social environment or working

methods change. These changes have a different nature than the design solution changes. Environment has some user needs that stay relatively constant and design solutions try to answer those needs. Environment documentation is thus has a longer life than the design documentation. In a process of design features are added, re-shaped and removed but for one increment of design the user environment target goals stay constant. This makes naturalistic problem modeling context basis for constantly changing design solutions. Also if the design does not help solving the naturalistic problems of users the product must be re-designed. Multiple possible formal realizations always exist. Those can be evaluated based on resources they take to implement as well as how well they take into account the user problem-solving domain.

The software design and development environment is constantly facing more and more pressure to produce more customer tailored solutions. The use case method is one step that helps incorporating the customer requirements and also providing external representation of design. Through more open communication of status and the problems currently in design the user requirements can be modified constantly to fit into the design and the design can be modified to fit into customer requirements in detail.

5.4 Further research questions

This study presents the usability evaluation methods integrated into design. The design methodologies are constantly under changes and improving, but this example helps integrating usability evaluation reports, usability experts and users' knowledge into design process. Even if not having complete technical detailed knowledge of system architecture, these parties can use external view of the product interaction, test it, and modify use cases. However the traditional view of engineering as a pipeline that produces solutions out of selected customer requirements has to be re-defined. The requirements in the first phase of design need not to be clear or selected, they can even change during design process. The prototypes and designed interaction workflows create new context of use that produces more detailed requirements. In this sense the requirements are selected all along the design process and the product need to be re-evaluated. Incremental design process will also have to change the early product definition phases. In successful design the information must flow both ways. More information available brakes the old myth of inventor that goes into garage and invents something new out, however increased information do not solve the problem of how is creative design made, how is the creative output of people shaped into concrete idea of how to shape the future realities? If design is seen as a creative activity, what are those individual as well as organizational aspects that help creating better design?

6. CONCLUSION

Use cases can be used as a working model of the product. The realization of intended product and influence of intended workflow in context of use can be usability tested even if no completed implementation exists. The earlier feedback from users enables fast changes in product. If the Participatory design methodology is used, the users can be used to validate design solutions. Increased feedback loops between the user context of use and design add more detailed knowledge into design. This increased amount of knowledge enables designers to do more user-friendly design. Use cases can be used as knowledge sharing documents between users, designers, testers and documentation people. The redundancy of creating several specifications and documents is diminished, when use case documents are updated regularly and re-used when test documentation or user documentation is to be created. The use case documentation works as an interface between different stakeholders of a software project. Thus people in physically different location and different organization can follow the development of the design.

Usability activities are broken into constant interaction between the intended users and the development project. Usability activities are not considered as separate (evaluative) actions of design. Instead usability is integrated into ongoing design activity as an interpretation of the use cases to users - creating scenarios out of use cases - and interpretation of the user requirements into use cases -incorporating the deficiencies found when presenting prototype or model to the intended user to use cases.

7. REFERENCES

- Alasuutari, P. (1994). *Laadullinen Tutkimus*. Tampere: Vastapaino.
- Alder, P., and Winograd, T. (1992). The usability challenge. In Adler, P., and Winograd, T. (eds.) *Usability: Turning the technologies into tools*, pp. 3-14. New York: Oxford university press.
- Anderson, J.R. (1990). *The adaptive Character of Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Cole, M. (1996). *Cultural Psychology: A Once and Future Discipline*. Cambridge, MA: The Belknap Press of Harvard University Press.
- Dreyfus, H. L. (1972). *What Computers can't do – A critique to artificial reason*. New York: Harper & Row.
- Ehn, P, and Löwgren, J. (1997). Design for Quality-in-use: Human-Computer Interaction Meets Information System Development. In Helander, M.(ed) *Handbook of Human- Computer Interaction*, pp. 299-314. Amsterdam: North-Holland.
- Guba, E.G.(1990). The alternative paradigm dialog. In E.G.Guba(Ed.), *The Paradigm dialog* (pp17-27). Newbury Park, CA: Sage Publications.
- Habermans, J. (1985). *The Theory of Communicative Action: Reason and the Rationalization of Society (The Theory of Communicative Action, Vol1)*.
- Hackos, J., and Redish, J. (1998). *User and Task analysis for Interface Design*. New York: John Wiley & Sons.
- Helander, M.(ed), Landnauer, T.(ed), Prabhu, P.(ed). (1997). *Handbook of Human- Computer Interaction*. Amsterdam: North-Holland.

- Holzblatt, K., and H. Beyer. (1998). Contextual Design. Defining Customer-Centered Systems. San Francisco: Morgan Kaufman.
- ISO/FDIS 13407:1988 (1999). Human Centered Design Processes for Interactive Systems. International Organization for Standardization. Geneva.
- Jacobson, I., Booch, G., Rumbaugh, J. (1999). The Unified Software Development Process. MA: Addison Wesley Longman.
- Jacobson, I., Cristerson, M., Jonsson, P., Övergaard, G. (1992). Object Oriented Software Engineering – A Use Case Driven Approach. MA: Addison-Wesley.
- Keinonen, T. (1998). One-Dimensional Usability – Influence of usability on consumers' product preference. Saarijärvi: Gummerus.
- Kulak, D., Guiney, E. (2000). Use cases: Requirements in Context. New York: ACM Press.
- Kolb, D. (1984). Experiential Learning: Experience as the source of Learning and Development. Englewood Cliffs, NJ: Prentice Hall.
- Landnauer, T. (1997). Behavioural Research Methods in Human-Computer Interaction. In Helander, M. (ed) Handbook of Human-Computer Interaction, pp. 203-228. Amsterdam: North-Holland.
- Landnauer, T. (1995). The trouble with computers – usefulness, usability and productivity. Cambridge, Mass.: MIT press.
- Lundequist, J., and Ullmark, P. (1993). Conceptual, constituent and consolidatory phases – New concepts for the design of industrial buildings. In Törnqvist, A., Ullmark, P. (eds.) Appropriate architecture: Workplace design in post-industrial society, pp. 85-90. IACTH 1993:1, Chalmers University of Technology, Sweden.
- Nielsen, J. (1993). Usability Engineering. Boston: Academic Press.

- Nonaka, I., and Takeuchi, H. (1995). *The Knowledge-creating company: How Japanise Companies Create the Dynamics of Innovation*. Oxford: Oxford University Press.
- Norman, D.A. (1998). *The Invisible Computer*. Cambridge, MA: The MIT Press.
- Norman, D.A. (1993). *Things That Make Us Smart*. Reading, MA: Addison-Wesley.
- Muller, M., Halswanter, J., and Dayton, T. (1997). Participatory Practices in Sowntware Lifecycle. In Helander, M.(ed) *Handbook of Human-Computer Interaction*, pp. 255-298. Amsterdam: North-Holland.
- Puttnam, H. (1988). *Representation as Reality*. Cambridge, MA:The MIT Press.
- Regnell, B. (1999). *Requirement Engineering with Use Cases – a Basis for software Development*. Lund: KF-Sigma.
- Salminen, P. (1999). *Totem Vantage Concept Description*. Nokia/RNT internal document.
- Salminen, P.(2000). *Increments in RUP phases*. Nokia/NET internal document
- Sheridan, T. (1997). *Task Analysis, Task Allocation and Supervisory Control*. In Helander, M.(ed) *Handbook of Human- Computer Interaction*, pp. 87-105. Amsterdam: North-Holland.
- Senge, P. (1990). *The Fift Discipline: The Art & Practice of The Learning Organization*. New York: Currency&Doubleday.
- Sommerville, I., and Kotonya, G. (1998). *Reguiremets Engineering: Processes and Techniques*. Chichester: John Wiley & Sons.
- Sommerville, I., and Sawyer, P. (1997). *Requirements Engineering: A good practice guide*. Chichester: John Wiley & Sons.

- Tuomi, I. (1999). *Corporate Knowledge: Theory and Practise of Intelligent Organizations*. Helsinki: Metaxis.
- Ulrich, K., and Eppinger, S., (1995). *Product Design and Depeloment*. Singapore: McGraw-Hill.
- Varela, F. J., Thompson, E. and Rosch, E. (1991). *The Embodied Mind: Cognitive Science and Human Experience*. Cambridge, MA: The MIT Press.
- Whiteside, J., Bennet, J, And Holzblatt, K. (1988). *Usability Engineering: Our experience and evolution*. In Helander, M. (ed) *Handbook of Human-Computer Interaction*, pp. 791-817. Amsterdam: Elsevier.
- Whiteside, J., and Wixton D. (1988). "contextuaism as world view for the reformation of metings" *proceedings of the Conference on computer supported cooperative work*, September 26-28. Portland, OR, p. 369.
- Winograd, T., and Flores, F. (1986). *Understanding computers and cognition – A new foundation of design*. Norrwood, NJ:Ablex.
- Wixton, D., and Wilson C. (1997). *The Usability Engineering Framework for Product Design and Evaluation*. In Helander, M.(ed) *Handbook of Human- Computer Interaction*, pp. 653-688. Amsterdam: North-Holland.
- Zachary, W., and Ryder, J. (1997). *Decision Support Systems: Integrating Decision Aiding And Decicion Training*. In Helander, M.(ed) *Handbook of Human- Computer Interaction*, pp. 1235-1259. Amsterdam: North-Holland.

Appendix A

The common software development methodologies are: Waterfall model, Technical and functional specifications, Ad hoc/ad in finum development, Rapid application development, Phased feature-based development, and Rational Unified Process (RUP).

Waterfall model

In waterfall model the software development life cycle is treated as a sequence. Each activity follows as a step. The output of one step follows as input for the next sequence. The output is frozen in the end of each step; this means that no modifications for a design phase are to be made after it is finished.

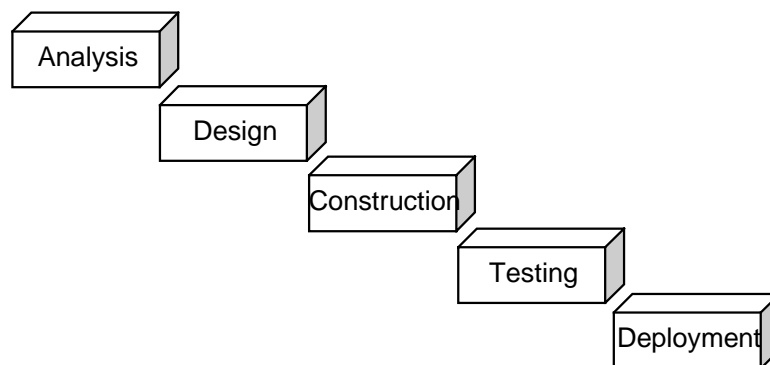


Figure 17 Steps in waterfall

In analysis step the analyst works very close to with experts in the given problem domain to capture the exact information (data) is required to be used, add how exactly the application should meet the customer needs. Design starts when some fixed set of requirements is given from the pervious set. In design phase the textual requirements are translated into specifics meaningful to both software environment and proposed system. Construction phase consist of the actual implementation, coding the design. Testing is broken down into "white box" and "black box" testing the modules need to be tested each separately and also the overall external logic need to be tested. Deployment includes the fixes found from the field and small enhancements for the application. This is the first phase the customer can verify any requests and check that product meets their needs

Usability experts have classically heavily criticised the waterfall model, because it's fundamental lack of customer co-operation and lack of design iterations. Other weaknesses include massive load of documentation and lack of dynamical resource planning.

Technical and functional Specifications

In this methodology the specifications are done separately and when completed handed to the development team. When the development team picks up the requirement specification the design continues the rest of the cycle using any number of methods.

The requirement phase can be done in other location than design, however the high risk of ill defined requirements as well as difficulty correcting ill defined or misunderstood requirements are disadvantages for this development methodology. Also lack of development team participation in requirement phase undermines commitment and motivation.

Ad hoc/ad infinum development

Coding without planning. The Management or customer sets needs for instant results. Therefore analysis and design phases are cut aside. The implementation is started immediately. Developers start making code (common to use some GUI based tools to quick results). This development method is mostly used in one or two person projects. Problem is that development goes on forever without reaching a conclusion. The design problems and milestones for the project are never explicitly set, and they are never met.

Rapid application development (rad)

The idea of Rapid application development is to quickly create an initial –often-fuzzy - requirement specification into simple prototype. This shallow implementation allows user to see some functioning code and get feel of the dynamics of the application. Customer is typically heavily involved with frequent review sessions. Evaluation of prototype often leads changes in requirements. Rapid development is used to explore critical user interface designs, complex algorithms and use of third party tools, application architecture. However critical is that the customer understands the idea of a prototype – the method produces fast models of actual product and the work amount of constructing the underlying functionality if not understood by customer causes unrealistic expectations.

Phased feature Based development

This software development method has two phases. First in requirement definition phase the requirements are gathered in sessions with client. The client's business, the processes and their workflow are used to define requirements. Some Rapid prototyping can be used to help defining requirements. In the Requirement phase the reasonable detailed set of requirements is designed to answer the following questions: System purpose, System features, Usecases, Initial object model, Sequence diagrams, sample GUI

During the application development phase the incremental approach is taken. The application functionality is continually built up in discrete. Within each increment development iterates set of discrete features.

The Rational Unified Process

Use case driven - Software is brought into existence to serve its users. To build a successful system, the user needs need to be captured. Use cases are capturing the intended interaction between a system and an external actor. In UML this external actor refers not only to human users but also to anything that is outside the designed system and interacts with it. Interaction of this sort is a use case. A use case is a piece of functionality in the system that gives a user a result of value, e.g. to accomplish a user's goal. Use cases capture the functional requirements. All the use cases together make up the use case model, which describes the whole functionality of the system. However, the use cases are not only a method to capture requirements, they also drive the development process. Based on the use case model the developers create design and implementation models that release the use cases. Also the design solutions are verified against use cases, to ensure that the proposed design is what was intended. Also the testers test that the implementation correctly follows the use cases. Use case driven means that the whole development processes goes through a series of workflows that derive from use cases. Use cases bind the workflows together.

Architecture centric - Architecture and concept provides the structure in which to guide the work in the iterations. Use cases are not selected in isolation. They are developed in parallel to the system architecture. Use cases drive the system architecture and the architecture influences the selection of use cases. Every product has a function and a form. The use cases describe the functionality of a system and the architecture is a form that is realising the system. Thus they have to be developed in parallel.

Iterative and incremental - Developing a commercial software product is a large task, requiring time and effort. It is practical to divide the work into smaller mini-projects, increments that can be controlled. Each mini-project is an iteration that results in one increment. It includes specifying/refining use cases, the analysis phase, design, implementation and testing. The iteration deals with a set of use cases that expand the services of a software product to satisfy user goals. Secondly, it manages the risks within each mini-project. In each iteration the developers identify and specify the relevant use cases, create the design, implement the design, and verify that the components satisfy the use cases.

Development cycles - The Rational Unified Process repeats itself in a series of cycles that each results a release to customers. A release is a product ready to be delivered to customers. It consists of the software, manuals and associated deliverables like training. Each cycle consist of four phases: Inception, Elaboration, Construction and Transition. The phases are further divided into iterations.

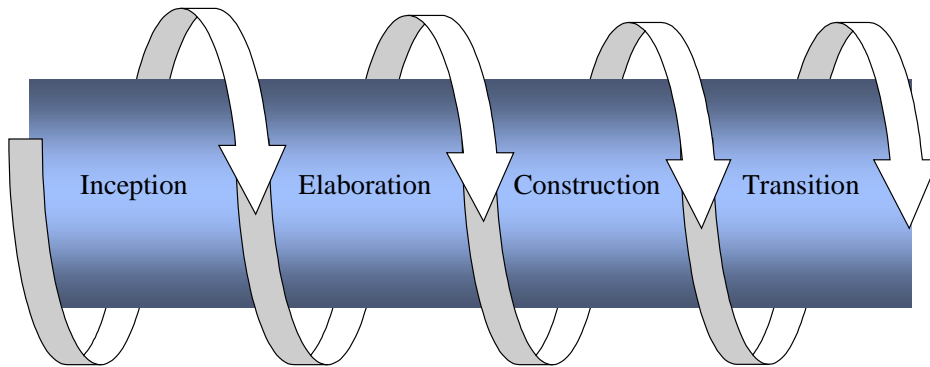


Figure 18 Inception, Elaboration, Construction and Transition the phases of Rational Unified Process

The inception phase develops the idea of a product into a shared vision of the product: It results in the plan of primary functionality, what the system is going to do to each of its major users, described in high level use cases. It includes the tentative architecture and the initial concept of a product. In this phase, the costs for developing the product are also estimated. In the elaboration phase the concept of the product exists. Most use cases are described in detail and the architecture is designed. Detailed project plans can be made. During this phase of development the most critical use cases are realised. During the construction phase the product is built. The architecture grows to become an almost functional system. The concept is realised. The vision evolves into a product and work practice that can be transferred to user population. At the end of this phase the product should include all the use cases agreed to be developed. In the transition phase a release is made for some selected pilot users. A small number of experienced users will have the product in use to evaluate the tool and report deficiencies. The transition phase includes the training of customer personnel and correcting bugs found in the delivery release.

Appendix B

Context of use – WCDMA Radio network planning

The user environment for and design context of use for observed software product is Wideband Code Division Multiple Access (WCDMA) radio network coverage and capacity planning. This appendix is aimed give reader the very basic knowledge what is radio network planning in WCDMA environment.

WCDMA system in brief

The 3rd generation mobile networks Universal Mobile Telecommunication System (UMTS) will be realised using Wideband Code Division Multiple Access (WCDMA) technology. This technology is referred as 3rd generation in contrast to GSM networks, which represent the 2nd generation. Radio network planning is in this document described only in taking into account the steps that are done with planning tool. Some information is also valid for 2nd generation systems, and it is likely that some planning methods will unify. The reason is that 2nd and 3rd generation systems will co-exist in many places of the world, and furthermore, both systems are providing of speech and data services.

In case of UMTS, the crucial item is the interference control. In the second-generation networks the interference control is related to frequency allocation. In third generation (3G) networks, the interference analysis capabilities and different options for the interference control are directly related to the capacity offered by 3G networks.

The main differences between WCDMA and narrow-band digital systems, such as Time Division Multiple Access (TDMA)/Frequency Division Multiple Access (FDMA), coverage prediction is that the interference estimation is crucial already in the first phase, when coverage is predicted. In addition to that, the different services (voice, data) have different processing gains and thus different receiver Signal-to-Noise ratio (SNR) requirements. In current coverage planning process the base station sensitivity is constant and the coverage threshold is the same for each base station. In case of WCDMA the coverage threshold is dependent on the number of users and used bit rates in all cells, thus it is cell and service specific.

Common features for coverage predictions exist. In all radio network systems, both of the links have to be analysed. In 2G both links tend to be in balance, whereas, in 3G the down-link (DL) can be higher loaded than the up-link (UL). This is simply because of asymmetric nature of traffic that 3G networks offer. Also propagation calculation is basically the same for all standards, with the exception that different tools can use different propagation models. One more common feature is the interference analysis: in WCDMA, this is needed for loading and sensitivity analysis, while in TDMA/FDMA it is needed for frequency planning.

3G networks will introduce a new concept of mobile users with data-capable terminals subscribed to different services. The distribution of such users in the networks will make a difference in radio network planning. This is especially true with high bit rate users, which will require a special attention during the planning phase. It should also be noted here that different services would call for different quality of service (QoS) requirements.

The 3G planning process in nutshell

In a cellular system where all the air interface connections operate on the same carrier, the number of simultaneous users directly influences the receivers' noise floor. Therefore, in UMTS, the planning phases cannot be separated into coverage and capacity planning as introduced in **Error! Reference source not found.** In WCDMA, the system is interference limited since every user shares the same bandwidth (frequency re-use is 1). The level of interference depends on transmit power of every mobile terminal in the network. Vice versa, the transmit power of every mobile terminal depends on the level of interference. Thus, higher the transmit power, higher the interference level, and vice versa. Therefore, the goal is find minimum possible transmit power, and yet maintain each connection

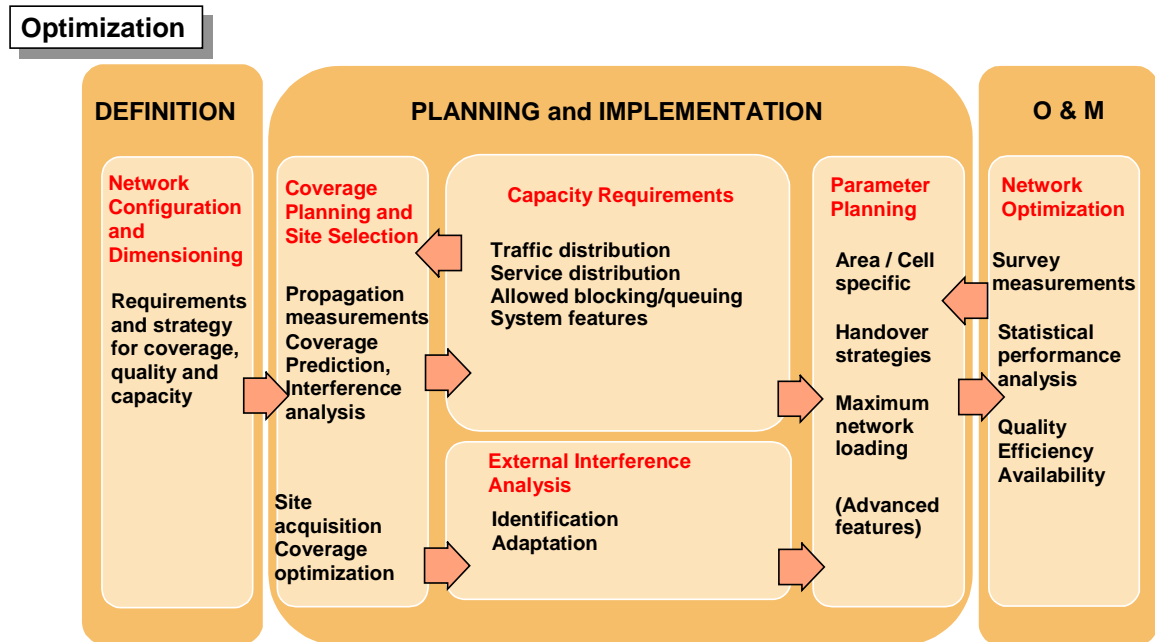


Figure 19 The UMTS planning process.

Since in WCDMA all users share the same resources they cannot be analysed independently. Each user influences the other one and causes its transmission power to change. These changes themselves again cause further changes and therefore the whole prediction process has to be iterative until the transmission powers stabilise or converge. Also the MS speeds, channel profiles and the bit rates and type of the used services play a more important role than in GSM systems. Furthermore in WCDMA there are new mechanisms such as fast power control, soft/softer handover (SHO) and orthogonal DL channels that affect heavily the system performance.

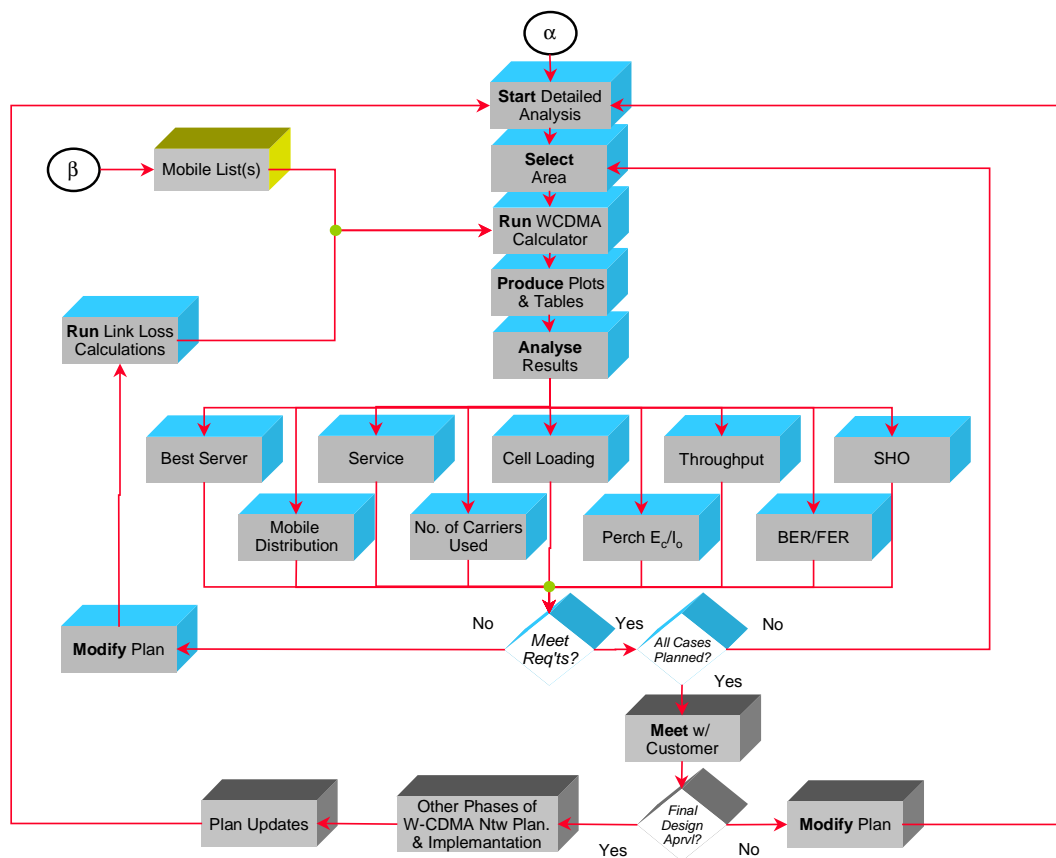
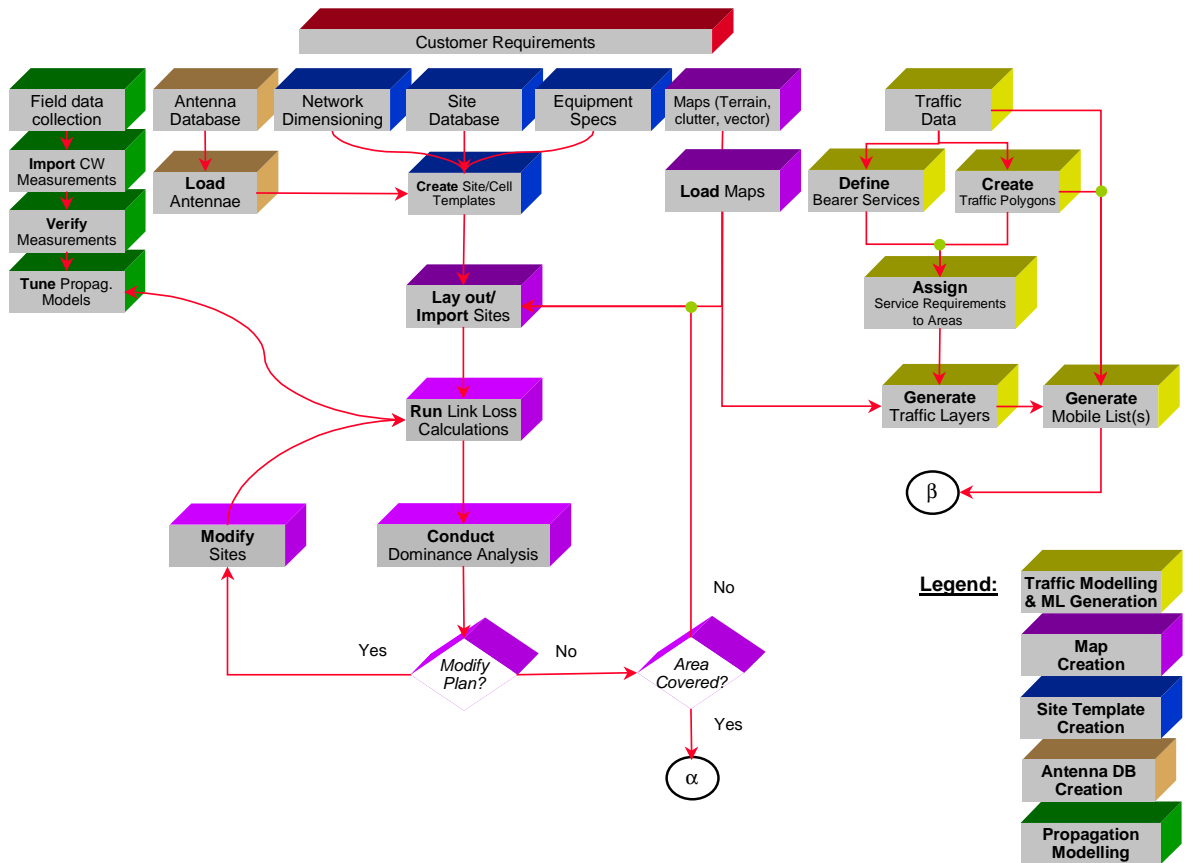
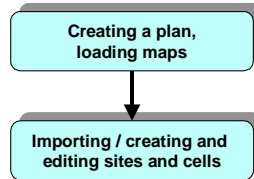


Figure 20 WCDMA Coverage and Capacity Planning Process.

Appendix C

Totem Vantage Overview



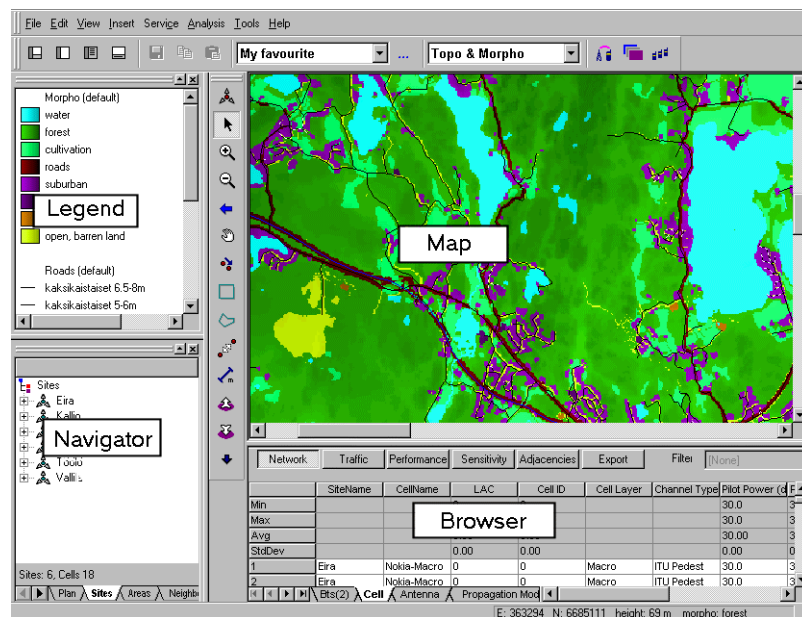
Totem Vantage supports the basic operations needed for radio network planning. It is possible to create BTS types, BTSs, sites, cells, antennas, and adjacencies. One can also edit BTS, site, cell and adjacent cell properties, and zoom and pan map views. Special attention has been paid to usability and special editing features have been introduced in order to maximise the efficiency and fluency of the workflow.

The main window menu bar and toolbars contain the commands for user operations. The toolbar provides a convenient way to access frequently required operations. Explicit tool tip assistance is supported for every toolbar icon.

The Totem Vantage main window is split up in three panes to optimise window management. The split windows are:

1. Navigator and legends
2. Map
3. Browser sheet

The size of the split windows can easily be adjusted in order to maximise the desired work area (map or Network Browser).



Navigator

In the Navigator, the network topology is shown as a hierarchical tree view. Based on this network composite structure, the user can easily access BTS, RNC, site, cell and antenna information, among others.

Site and cell objects in the tree view can be sorted, filtered and selected. In addition, a context sensitive pop-up menu is available for basic operations like listing to Browser, locating sites on map, and deleting and renaming.

Navigator has separate tabs for different types of data grouped in a logical way. For example, there is a tab for Site, Area, and Neighbouring cell handling.

Network Browser

The Network Browser offers an easy and effective interface to inquire, compare and edit cell parameters. Any combination of site related parameters can be defined to be presented in the browser view. A new set of parameters can easily be activated by using tabs. Mass editing of parameters in all or selected cells saves time by reducing the amount of repetitive work. Information from the selected rows and columns can be exported to a file or printed.

In order to help the network planning process and planner's workflow, there are so-called view sets which are composed of sets of logically grouped browser views. For example, network configuration data (sites/cells etc) can be found in one view set, whereas a view containing performance figures from various calculations can be found from another view set. It is also possible to specify and modify new view sets for items that need to be viewed together.

Network Browser has also an integrated filtering functionality. This enables very efficient site and cell selections, for example. A direct consequence of this is that smaller planning areas can be controlled easily without touching the rest of the data and network elements. Once defined, any filter can be stored for later use.

Map

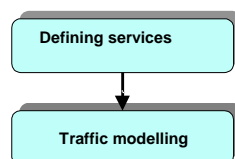
Totem Vantage digital map component utilises the following data:

topographic data in raster format (terrain height), morphographic data in raster format (terrain type), building height data in raster format, vectorised building data

A raster unit is usually in the range of 1 to 200m. This data is used in propagation predictions. Morphographic data can also be utilised in traffic density estimation.

Co-ordinates in Totem Vantage are in orthogonal (metric) format. Consequently, co-ordinates in various interfaces to external software (site import, measurement file interface, etc.) need to be in orthogonal format.

Bearer service and traffic modelling tools



Bearer service and traffic modelling features of Totem Vantage allow flexible traffic forecast definitions. Traffic distributions are the basis of the plan – they are needed for evaluating interaction of coverage and capacity. The more accurate the traffic estimate the more realistic the achieved results.

In the service definition phase, bit rate and bearer service type are assigned for each bearer service. For non real time traffic it is also possible to define the average packet call size and retransmission rate, i.e. it is also possible to model packet data services and calculate average throughput for both UL/DL and delays.

In the traffic modelling phase, the user can select between different ways to create traffic forecasts. The busy hour traffic can be given as input figures, or measured traffic data from commercially available measurement tools can be exploited. For example, knowledge of hot spot locations in the current network can be useful.

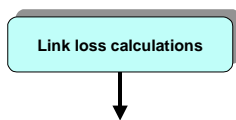
Different weighting methods can be applied when assigning traffic amounts to areas. For example, equal distribution or weighting based on morpho or road types can be used.

Traffic density is modelled separately for each service. Furthermore, traffic density of different services can be combined and integrated to be handled concurrently. Traffic can also be modelled in a mixed bearer service situation, where there is both real time and non-real time traffic.

Based on the traffic forecast, the Totem Vantage creates a realisation of the statistics by producing a snapshot of simultaneously active mobiles in the network. In the same context also a speed target based on service and morpho information is assigned to each mobile. Mobile parameters, e.g. minimum and maximum powers and speed, can be also modelled and specified. The WCDMA Calculator (described in section 0) uses this information for transmission power allocation. It is also possible to create multiple mobile lists, which enables the analysis of the effect of varying mobile lists to network performance under unchanging traffic conditions.

Traffic data can be shown in a 2D view and the different traffic scenarios can be saved as mobile lists for later use.

Link loss calculations



Link loss calculations are based on the propagation loss and gains from the receiving and transmitting antennas, and various losses on both ends of the radio path (such as cable, divider/combiner, duplexer and body losses). The gain of an antenna depends on the direction of the antenna, and for this reason the bearing, tilt, and pattern of the antenna need to be provided when calculating the gain.

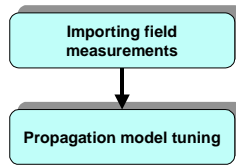
For radio signal propagation modelling Totem Vantage provides the traditional Okumura-Hata and Walfisch-Ikegami, which have been specially tailored in accordance with latest Nokia research work. It is also possible to attach external propagation models to the basic software.

The Okumura-Hata model is best suited for macro cells and for small cells in which the antenna is located above the surrounding rooftop level. A valid range for Okumura-Hata model is 1–20 km. The Okumura-Hata model includes optional line-of-sight checking, morphographic correction, and multi-obstacle diffraction loss calculation using the Deygout method which computes the diffraction loss of up to three obstacles. The line-of-sight checking uses a separate path loss function for determining line-of-sight paths.

The Walfisch-Ikegami model is intended for small cell planning where the maximum cell radius is 3–5 km. The model is based on Nokia's own measurements and on the model proposals presented in the study group COST231. The Walfisch-Ikegami model also includes an optional line-of-sight checking, morphographic correction, and multi-obstacle diffraction loss calculation using the Deygout method.

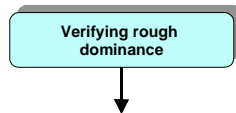
Link loss calculations can be speeded up by distributing them into several processors. In this way the calculation times can be reduced dramatically compared to a single-processor standalone workstation situation. When this option has been selected, Totem Vantage will automatically use the available computing power that can be found from other Totem Vantage workstations belonging to the same LAN.

Propagation model tuning



All propagation models can be tuned on a per cell basis. Field strength measurement data can be imported to Totem Vantage in order to tune propagation model parameters, mainly morpho class correction factors, which need to be tuned to match local conditions. Being able to tune the model based on measurement data brings substantial improvements in the accuracy of radio path propagation predictions. Several measurement systems are supported including ESVD and TOM measurement data formats.

Dominance analysis



Dominance analysis shows the dominant cell in the area based on pure link loss data and common pilot channel power settings. This is especially useful for planning dominance areas that join smoothly together.

Dominance areas can be shown with user-specified cell or RNC colour. To speed up the planning process the planner can first make rough link loss calculations and after that detailed predictions for accurate results.

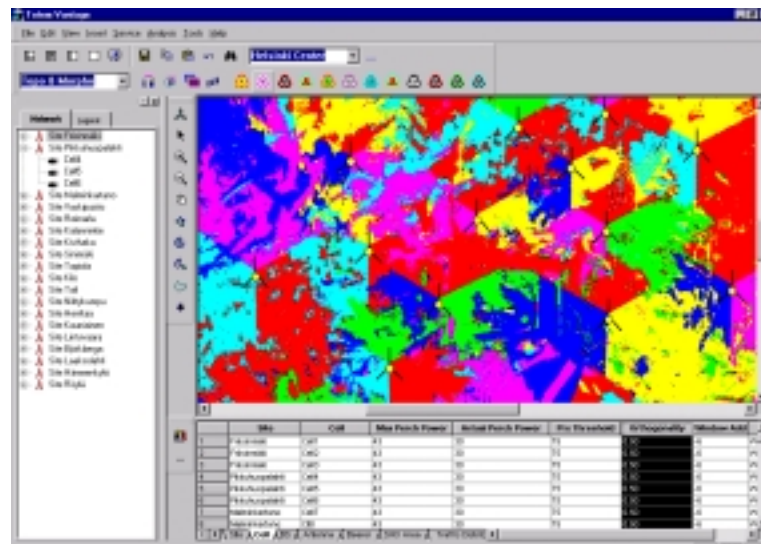
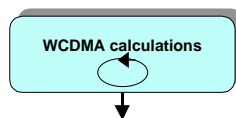


Figure 21. Example of dominance analysis.

WCDMA Calculator



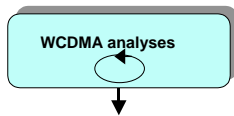
Calculations in a WCDMA system depend mainly on two factors: link loss and interference. The theoretical maximum cell range depends on both the maximum allowed link loss between the BTS and the mobile station and on the Common Pilot Channel power of the cell. WCDMA networks are unique in that a number of transmissions can use the same carrier frequency within the cell and in the neighbouring cells. While the desired signal is detected, the other signals cause interference that is interpreted as noise in the input of a receiver. Interference from other transmissions is the main limiting factor of a WCDMA cell. Therefore, transmission powers are minimised during normal network operations in order to maximise the number of MSs that can be concurrently connected to the network.

The objective of the WCDMA calculator is to predict interference in the network due to the estimated traffic or capacity need. This is done using a unique iterative algorithm, which emulates both fast power control and Soft Handover (SHO) process in the uplink and the downlink direction. Soft and softer HOs are separated and taken into account in all analyses.

The MSs do not move during the iterations, but their speed can be separately defined for each service type. It is also possible to create multiple mobile lists the data of which can be statistically combined to provide information about network performance under unchanging traffic situation. Sensitivity analysis is made based on this data. (See also section 0.)

Uplink load is defined to be a function of interference power. In WCDMA networks cells can be filled up to the load limit (parameter set by the planner) and the purpose of the limit is to prevent the network from becoming unstable due to excessive interference. This load limit is applied during the iterations and thus some MSs might not be served due to high interference. In DL the limited resource is transmission power. If the total BTS transmission power is reached then some of the MSs have to be left without service. Also in DL the planner can choose between the selection methods applied to reduce the overload. These methods (both in UL and DL) emulate the radio resource management algorithms running in Nokia RNC.

WCDMA analysis



The user can choose between fast and more accurate WCDMA analyses depending on the needs. It is also important to be able to take a quick look at the behaviour of a single mobile snap-shot and perform network tuning if needed. After this multiple iterations can be run in order to get statistically reliable network performance figures. With Totem Vantage Sensitivity Analysis, the user can investigate either a single iteration, or let the tool automatically generate a specified amount of iterations and then combine the performance figures from all the iteration runs automatically in a meaningful way.

Served and not served MSs

The served MSs are established as a result of the iterative WCDMA calculations described above in section. The served MSs create a load level in the network, and all other calculations are based on that load. For example, the coverage results are different in different load conditions since WCDMA is a system where interference is a limiting factor.

During the iterations the E_b/I_0 target of each MS is applied when tuning the transmission powers. Consequently the E_b/I_0 target of each served MS is reached, and hence, the MSs have a good enough quality.

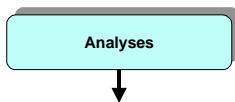
The served and not served MSs can be viewed on the map. In addition, not served MSs (outage mobiles) can be shown according to the reason why they are not served (e.g. UL overload, not enough MS TX power, and not enough BTS TX power). Furthermore, statistics are collected from the selected analysis area and displayed separately for each service type in Network Browser.

Based on these statistics the planner can decide whether the amount of not served MSs is low enough to fulfil the requirements.

Analysis tools

There are several analysis results available from the WCDMA Calculator. The results are shown in browser views, histograms and as rasters on the map. In this section, the most important analyses are described.

Load analysis



One of the outputs of the iterative interference calculations is the load in each cell. The load is an outcome of the MSs served (see section 0). The cell load analysis displays the UL load situation in each cell. Cell load is a function of interference coming from both the own and other cells.

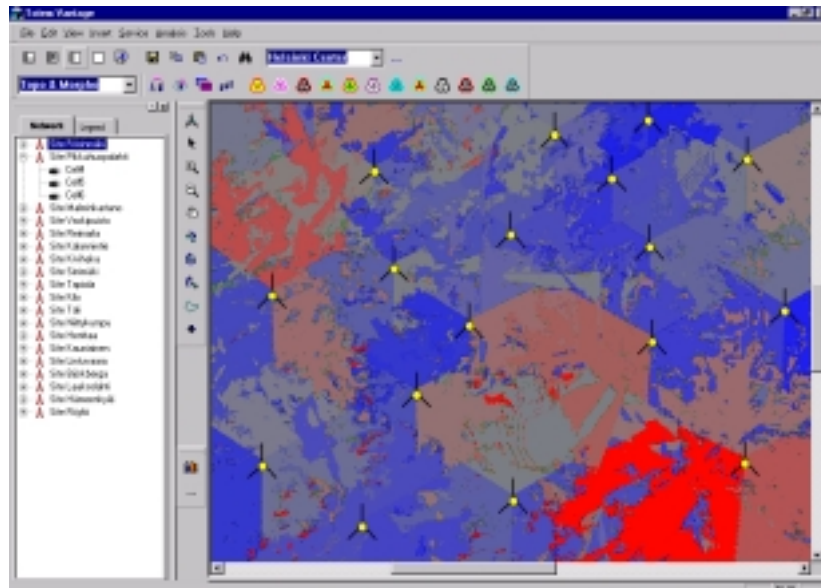
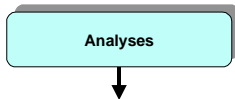


Figure 22. Example of load analysis in UL

Coverage and capacity analysis tools

Primary common pilot channel coverage analysis



A precondition for providing service is that the common control channels, for example, the pilot channel, cover the area with the required quality. Totem Vantage includes tools for analysing the common pilot channel coverage.

The analysis shows the areas where the E_c/I_0 of the pilot is predicted to be better or worse than the threshold value set by the planner.

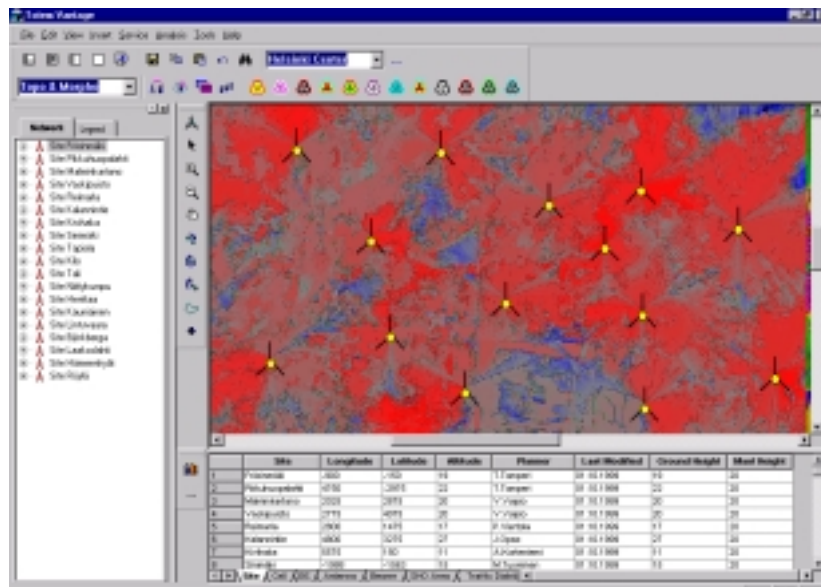
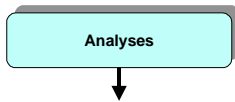


Figure 23. Example of pilot quality analysis

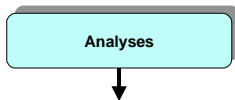
Best server analysis



The best server analysis for the uplink direction finds for each network location the cell, which a MS in that location can reach with the minimum transmission power. This analysis is performed after the iterations have been carried out with the WCDMA Calculator since the analysis takes into account the cell load.

The best server analysis for downlink direction finds for each network location the cell whose common pilot channel power is received with the highest level.

Uplink coverage analysis



The output of the coverage analysis is the service probability of an incremental MS with the selected service type/speed combination in the current load situation and in the defined area. The results presented in graphical and numeric format can be compared against the requirements. Coverage analysis can be performed for one, user-defined, speed and service type at a time, or alternatively, different services can be analysed simultaneously and a composite coverage plot is made for the selected services. The desired area to be analysed can be defined on the map using polygons.

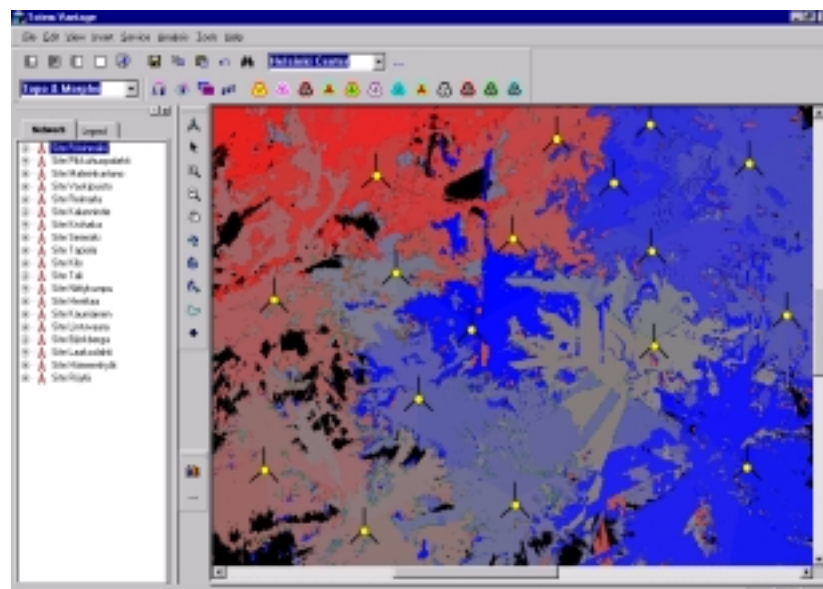
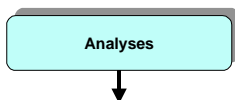


Figure 24. Example of coverage analysis for 384 kbit/s data rate. Black areas correspond to no coverage.

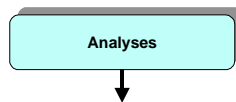
Throughput analysis



This throughput analysis shows the sum of the information bit rates of the served MSs in each cell (see also section 0). The SHO effect is removed from the results, which means that each MS is included in the results only for the cell that serves it best. The throughput varies between the cells depending on the traffic and interference conditions. The throughput and the number of links that have the cell as the best server are shown on the map and in histograms.

For packet data (non real time services) also the average throughput and delay is calculated.

Soft handover analysis tools



The soft handover (SHO) area analysis displays the geographical areas where SHOs are predicted to occur and the corresponding active set sizes. The SHO overhead analysis calculates and displays the extra resources needed due to SHOs. The SHO analysis results (soft + softer) can be viewed in histograms and on map displays.

SHOs are an essential part of WCDMA system but too big a number of MSs in SHO or too big active set sizes waste resources. These tools help the planner to optimise the SHO parameters in each cell.

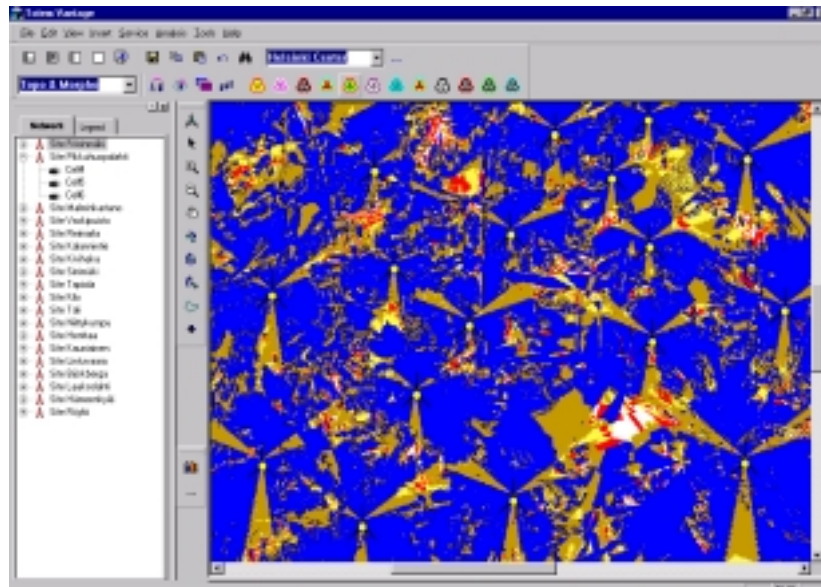
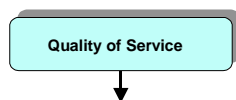


Figure 25. Example of soft handover area analysis.

Quality of Service

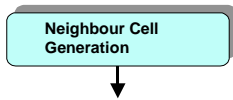


Quality of Service analysis is performed for obtaining the delay and throughput figures for packet data services. Main principle in the algorithms is to estimate the queuing delays caused by the interference limits over the air interface. Calculation method is also based on Erlang B and C formulas. By these means Totem Vantage is able to estimate the QoS which the end user of MS would be experiencing in the deployed radio network.

Modelling of cell performance in a mixed bearer service situation is supported, i.e. a situation where there is a certain proportion of real time (RT) traffic mixed with non-real time (NRT) traffic. Traffic modelling includes parameters for defining the amount of average NRT traffic both for uplink and downlink direction. The parameters include, average packet call size and packet retransmission activity, among others.

WCDMA analyses take the characteristics of RT and NRT bearer services into account. As in the real networks, WCDMA Calculator gives precedence to RT bearer services over NRT bearer services. The analysis results for modelled packet data traffic, such as cell throughput for UL/DL and average delay per bearer service can be seen from the resulting performance figures and graphical outputs.

Neighbour Cell Generation



A neighbour cell list contains neighbours for all cells in the radio access network. Neighbours are defined on a per cell basis. When the neighbour cell list has been created, it is possible to view the list of generated neighbours on the list and modify the neighbour related parameters. When the result is satisfactory, the list is saved and transferred to Nokia NetAct via generic configuration data transfer interface.

Before performing neighbour cell list generation it is essential to have right configuration and parameter settings. Therefore neighbours are usually generated only after all other analyses have been successfully performed and optimum configuration already achieved.

Neighbour cell lists contain the adjacent cells (neighbours) of each cell in the RAN (Radio Access Network). Neighbour cells are defined on a per cell basis, i.e. each cell can have a unique adjacency list.

When neighbour cell lists have been created, it is possible to view neighbouring cell relations on the map, and study and modify the related parameters. Neighbour cell lists can be exported into NetAct via generic configuration data transfer interface. This enables fluent workflow from network plan to operating network.

Reporting



The Network Browser data can be exported in ASCII files and imported to Microsoft Excel, for example. This data and the tables and graphs from the analyses described above can be printed out with a Windows compatible printer.