USER PARTICIPATION
in Systems Development

Bachelor of Science Thesis
Department of Informatics
School of Economics and Commercial Law
Göteborg University
Autumn, 2001

Advisor: Christian Maloney
Author: Raye Walter

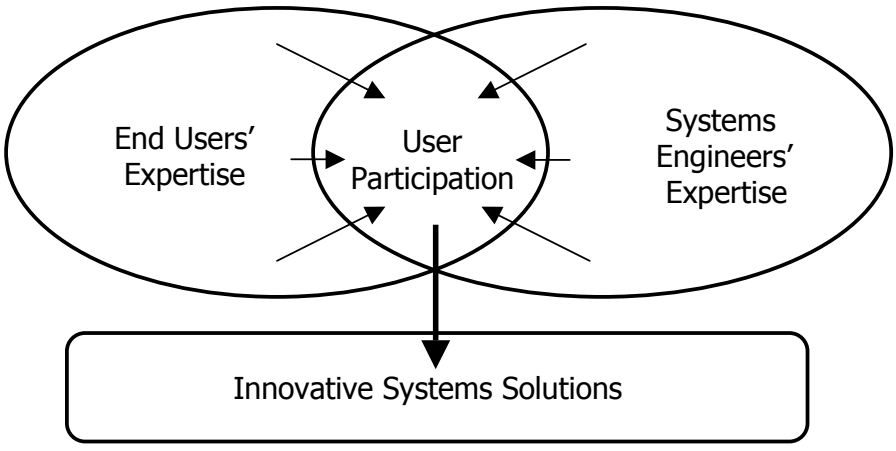
TABLE OF CONTENTS

1.1 ABSTRACT

This study offers a glimpse into a number of subtratal concepts, philosophies, and methodologies, which undergird the evolution of theories and concepts supporting User Participation in systems design and development. It takes a comparative look at how this concept has been structured into the frameworks of several leading system development methodologies. Of particular interest is an approach, which has its origins in Scandinavia, as the author of this study is a student at a Scandinavian university. The intent of this study is to outline and highlight the potential for end user influence in systems design. It is hoped that this study may help foster further interest and dialog relating to the effective utilization of modern methodologies in broadening and strengthening the expert domain competencies of both end users and systems designers. The study concludes with a brief discussion regarding several of the merits of user involvement in systems development.

As we progress into the 21st century, mounting pressure is being placed upon meeting the express and explicit needs of a seemingly boundless IT technology expansion. Computers and computer systems are laying claim to an ever-increasing segment of modern society, and placing greater and greater demands on the engineering of well-integrated computer systems. To meet this challenge, many philosophies, methods and methodologies have been developing and evolving, throughout the mid-to-late 1900's. Some of them deal with the logistical and technical aspects of workflow, others focus on functions and tasks in the distribution of information, while yet others place emphasis on science and research. There is, however, yet another concept of systems development that has been steadily making inroads into the field of systems development; that of, User Participation. My aim is to examine this 'common sense' approach to systems design and development, for even more successful computer systems applications, by taking a comparative look at a number of the leading systems development methodologies available to date. This study concentrates on a specific area of interest; namely, how the concept of User Participation has evolved and how it influences various systems development methodologies.

User Participation is a collaborative process. The idea is to join end users, who are experts in their occupational domains, with systems designers, the experts in computer technology, to create new and innovative systems solutions; the goal being, to improve product usability and the overall quality of the work experience and environment.



1.2 PURPOSE

The purpose of this study was to:

- Highlight the various positive aspects of the User Participation concept
:
- Explore how the convergence of computer systems development and behavioral sciences (psychology, sociology, philosophy) acts as a catalyst for further innovative advances in computer systems development

- Examine how the User Participation concept is integrated into computer systems development processes

Other goals of this research were to:

- Underscore some of the tangible benefits of a systems engineering approach that encompasses and utilizes the inherent potential in merging end user experience with systems design expertise

- Foster further discussions concerning the functionality of user participation in systems development projects

1.3 FOCUS

The central focus of this study was placed on presenting an informative overview of the correlations between user-centered concepts, methods, and techniques, and a number of systems development methodologies. Answers to the following questions were sought.

- How have we arrived at today's concept of User Participation?

- Is User Participation a new concept? Under what circumstances has it emerged?

- What are the practical effects of User Participation on systems development?

- How diverse are methodologies, which espouse user participation?

1.4 SCOPE AND LIMITATIONS

Primary emphasis was placed on highlighting those aspects of each of studied methodology, which dealt with user participation. Developing more of a perspective on the underlying thought processes in the integration of user involvement in each

methodology was another important element, which guided the scope of the research. This study focused on the theoretical aspects of the subject matter.

Time and level-of-study restraints also played a decisive role, when considering the scope of the research. These were two factors of particular importance, when setting parameters, in regards to the breadth and depth of the work to be presented. Therefore, this study did not attempt to present all the researched material in detail, but rather to provide a succinct, yet thorough, overview of the chosen material.

While there are several variations on the theme of user involvement in systems design, this study restricted itself to discussing a select number of the more prominent systems development methodologies in its inquiry into which, if any, best reflect the aims of User Participation.

The discussion of opposing points of views, or the analysis of practical considerations, such as logistics or economics, was not taken up in this study.

1.5 STUDY METHODOLOGY

This dissertation was based on a literary study of various books, course literature, published articles and other research reports, as the focus of this study was theoretic in nature.

Access to hard copy materials (books and course literature) proved to be limited. Therefore, a substantial amount of reference material was obtained via the Internet.

Personal interviews were not used in this study, as it was deemed that the availability of printed resources was adequate to provide the necessary reference base for the intended analysis.

A combination of positivism, and an inductive evaluation approach was used in the research process to assess the validity and reliability of the research material used in this study.

A 3-tier presentation:

- Scandinavian Approach
- Contributing Elemental Concepts and Research
- Leading Systems Development Methodologies

was made to provide the basis for a methodic evaluation of the findings in the research material and the subsequent discussion regarding User Participation in computer systems design and development.

A DISTINCTION BETWEEN THE TERMS METHOD AND METHODOLOGY

The terms *method* and *methodology* were found being used in different contexts in the research material. This raised the question of definition. Clarification of these terms offered by Brown (1997) is offered below:

A *methodology* in the context of systems development can be described as a description of the developmental process. Methodologies can vary in both breadth and depth; as to how much of the development process is covered by said methodology, and as to how much detail its methods present.

The individual steps or stages that a methodology is comprised of, in which techniques for implementing that particular step or stage are presented, are the *methods*. It is noteworthy to mention, however, that methods can also stand alone, not being a part of a methodology

1.6 DISPOSITION

The remainder of this study was divided into the following:

CHAPTER 2

SCANDINAVIAN APPROACH

CHAPTER 2 PARTICIPATORY DESIGN

As a student of one of Scandinavia's largest universities, it seemed only fitting to begin the research with an exploration of the Nordic contribution to the emergence of User Participation:

2.1 What is Participatory Design (PD)?

2.1.1 Definition

Participatory Design (PD) is a methodology in which representative end users provide continual feedback to computer systems designers during the development of system prototypes. This collaborative team of people represents the major stakeholders in a product or system design effort. By bringing these "domain experts" together, a vital link is established where users can interact directly with designers in the development process, with their suggestions for product improvements before those suggestions are codified into a program. The intent is to create designs that reflect the way the end-users actually use the product in their work. (PDC, 1998)

2.1.2 Philosophy

Reich (1997) writes:

> Design can be interpreted as a product or a process. As a product, it is an object that was conceived and realized in the same way. As a process, it is the sequence of events from conception to realization of the design object.
>
> **Premises:**
>
> - We are all designers and customers – producers and consumers of design.
> - Design is a social process.
>
> **Conclusions:**
>
> - In almost every activity there is a design aspect.
> - Social processes permeate our activities.
>
> Participatory Design is the antithesis to traditional design. Design knowledge exists in all those potentially affected by a design, and they can all contribute to design a better product. This is carried out in a social process of communicating, sharing, reconciling, and acting.

Magnusson (2001) quotes Löwgren and Stolterman's *Design av informationsteknik* (The Design of Information Techniques):

> Participatory Design is a process of mutual instruction, where designers and end users learn from each other. The more one shares a social and cultural background [environment], the more one shares a language, the more one participates in the design process. Participatory Design demands not only that end users share in the design process, but also that the designer shares in [work situations]. (English translation)

2.1.3 History

2.1.3.1 Origins

Participatory Design has its origins in Scandinavian trade unions' initiatives toward democratization in the workplace. The objective was to include the perspective of the worker, concerning the introduction and development of new technologies. The aim was to strengthen the workers' position in regards to the introduction and use of computer technology. The original concept was one of "work-oriented" systems design. Pelle Ehn (1992), of Denmark's Aarhus University, writes, "Democratic participation and skill enhancement – not only productivity and product quality – themselves [were] considered ends for the design." One concept, The Collective Resource Approach, was fostered in Norway, Sweden and Denmark. This approach to systems design promoted the notion of collective cooperation between two different areas of expertise (systems technology and end user experience) in the systems design process. By so doing, the most favorable conditions would be created for understanding the demands and requirements that a particular computer system would need to address.

Two Norwegians, Kristen Nygaard and Olav Terje Bergo, are credited with developing this new mindset in Norway, in the 1970's. Norway's strides toward democratic representation between trade unions and business organizations, after World War II, served as catalyst for this new thinking. An even earlier collaboration between British and Australian researchers and Norwegian Einar Thorsrud, in the 1960's, lead to the development of a programming language, SIMULA, in 1965. This language is used in object-oriented programming, and in a process known as, "Business Process Control".

Nygaard and Bergo's Collective Resource Approach greatly influenced many projects throughout Scandinavia concerning the integration of unions and end users into the systems design and development process: Norway's Norsk Jern- och Metallarbeiterforbund (NJMF), Sweden's Demokratiske Styrnings-systemer (DEMOS), and Denmark's Demokrati, Utvikling og Edb (DUE), just to name a few. UTOPIA and UNITE are two other, worthy of mention.

2.1.3.2 Contributing Factors

That this approach to democratization in the workplace had its roots in Scandinavia, and not, say, in England or Germany, is interesting to assess. Such contributing factors as being a somewhat geographically isolated people, accustomed to self-determination (Bentsson, 1995), certainly played a decisive role. This all lent itself to the emergence of a Nordic culture that could indulge itself in pursuits other than war and reconstruction (although, of course, both Norway and Denmark were occupied by the Nazis, during WWII).

Scandinavia has a reputation for its distinctive industrial relations. With a highly educated workforce; a high level of unionization by strong national

trade unions, with links to ruling social-democratic parties; and a positive interest in new technologies (Ehn), it stands to reason that such a homogeneous environment would provide suitable conditions in which to test and champion various concepts concerning the workplace, the overall quality of life, and the promotion of democratic ideals. Undisturbed, and with a long period of economic and political stability, this region of the world could develop a cultural and political tradition with strong emphasis on the rights and interests of the individual citizen, and cooperation between different social groups. These factors helped foster a pragmatic attitude towards technical development, which in turn promoted a tendency toward long-term planning.

The emergence of the concept of Participatory Design seems to have been a natural extension of an evolutionary process. Because, in order for the Scandinavian ideals of democratization process to avoid stagnation, it stands to follow that new technologies would need to be adapted to the needs of the people, and not vice versa.

2.1.4 PD Today

Today, the heavy focus on ”work-orientation” and ”democracy in the workplace” has given way to more socio-technical aspects of user participation. (PDC Workshop, 1996)

The influence and use of Participatory Design reaches far beyond the computer systems development arena to include such broad and diverse fields of product development as community housing and children’s tutoring aides. (PDC, 1998)

2.1.5 Examples of Practical Applications

There is a myriad of variations on how to approach the practical application of the PD process. Here is a representative conceptual model of a practical application of the PD process, as a *methodology*, on a software development project, as described by Michael Good (1992) of Digital Equipment Corp., in New Hampshire, USA:

1) Building relationships - We would spend enough time together to ensure a good fit between customer-participants and the P.... project. This included familiarizing our customers with p.... technology.

2) Contextual Inquiry - Contextual inquiry emphasizes interview methods conducted in the context of the participant’s work and building an understanding of work in context. The computer engineers needed to build an understanding of the customer’s work before we could collaborate as co-designers.

3) Brainstorming - Brainstorming sessions, where all ideas are recorded and criticism of ideas forbidden, would generate many ideas for how p.... technology could improve work.

4) Storyboarding - Customers and computer engineers would develop some of the most promising brainstorming ideas into illustrated scripts

of a ”day in the life” of a customer with p.... technology in the future.

5) Iterative Design - Using the storyboards as specifications, the computer engineers would build prototypes that would be tested by the customer-participants on a regular basis. All the previous steps would continue in an iterative fashion.

Gaffney (1999), of Information and Design Ltd, describes the use of PD, as a *method*, in a workshop setting:

> A Participatory Design workshop is one in which developers, business representatives and users work together to design a solution. PD workshops:
>
> - Give users a voice in the design process, thus increasing the probability of a usable design
> - Enable technical and non-technical participants to participate equally
> - Provide an opportunity for developers to meet, work with and understand their users
> - Provide a forum for identifying issues
> - Provide an opportunity to get or enhance user buy-in
> - Are highly productive
> - Use techniques that can be easily learned and applied in future activities.
>
> The ideal number of participants is probably 8 or 9.
>
> *Sample Agenda*
>
> Agendas will vary depending on the problem at hand, the attendees, and the amount of time available. The following is an example only:
>
> - Introductions - Participants introduce themselves. The facilitator can set the tone by being first to do so.
> - Usability presentation - This is an opportunity to get participants thinking about usability.
> - Objectives and Expectations - Be clear about the purpose of the workshop, and identify what each participant expects as an outcome.
> - Identify issues - The issues may be with a system to be replaced, or with the domain in general. Use affinity diagramming to extract and structure the issues.
> - Design goals - With the issues in mind, identify the usability goals that the system must meet.
> - Scenarios - Scenarios serve to center the discussion on the actual users. Have participants read and refine the scenarios.
> - Paper prototyping - Split the group in two and have each spend a short amount of time (no more than 20 minutes) working independently on solutions that address the selected scenario or scenarios.
> - Combine designs - Each group presents its design and the group discusses relative merits.
> - Further design work - Depending on the outcome of the first prototyping session, decide how to use the remaining time most effectively.
> - At the end of the workshop, review expectations and objectives to ensure they have been met.
> - Document the outcomes as soon as possible.
> - Be prepared to diverge from the prepared agenda if necessary.

2.1.6 Impact on Systems Development

Since 1990, PD conferences have brought together diverse international and interdisciplinary groups of designers, researchers, practitioners, users and managers to discuss, debate, and exchange ideas on different strategies and the further development of the PD process. In particular, there is the Participatory Design Conference, which meets bi-annually. (PDC, 1998)

CHAPTER 3

CONTRIBUTING ELEMENTAL CONCEPTS AND RESEARCH

Chapter 3.1 Usability

Usability First (2001) says that usability ”addresses the relationship between tools and their users”. The effectiveness of a tool is measured by assessing the ability of its users to execute tasks successfully. And of course, the same principle applies in systems development.

There are many important factors to consider when evaluating usability in a systems solution. Among them: the level of functionality in addressing user needs; the fluidity in the use of the application when executing tasks; the ability of the application to meet users' expectations.

Iterative design is the means by which to develop maximum usability. It is the repeated process of progressive refinements, aided by user testing and feedback, which facilitates the creation of highly usable systems solutions. (*Usability First*)

And to offer guidance in calculating the degree of success in maximizing the required or desired level of usability in a product, a definition of the term *usability* has been standardized as follows:

USABILITY STANDARDS (International Organization Of Standards 9241-11)

"System usability comprises the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use, where:

- **Effectiveness** measures the accuracy and completeness with which users achieve specified goals;
- **Efficiency** measures the resources expended in relation to the accuracy and completeness with which users achieve goals;
- **Satisfaction** measures the freedom from discomfort, and positive attitudes towards the use of the product.

Usability Attributes: Usability attributes outline the features and characteristics of the product that influence the learnability, effectiveness, efficiency and satisfaction with which users can achieve specified goals in a particular environment.

Context of use: The users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used.

Work system: A system, consisting of users, equipment, tasks and a physical and social environment, for the purpose of achieving particular goals. · User: The person who interacts with the product.

Goal: An intended outcome.

Task: The activities required to achieve a goal. These activities can be physical or cognitive. Job responsibilities can determine goals and tasks.

Product: The part of the equipment (hardware, software and materials) for which usability is to be specified or evaluated.

Measure (noun): The value resulting from measurement and the process used to obtain that value."

(NOTE: In a previous version of ISO 9241-11 "Learnability" was still defined as an attribute of usability. The "learnability" attribute is now included under the Usability section of ISO 9126 - Software Quality Characteristics)

Software quality is categorised into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability).

Here *usability* is defined as follows:

- The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
    - **Understandability**: The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
    - **Learnability**: The capability of the software product to enable the user to learn its application.
    - **Operability**: The capability of the software product to enable the user to operate and control it.

Chapter 3.2 Human - Computer Interaction (HCI)

As in the case of most evolving sciences, there appears to be no single agreed-upon definition of Human-Computer Interaction that covers all the topics that make up this discipline. The following definition and subsequent text attempts to clarify the general aspects of HCI.

Human-Computer Interaction (HCI) is a multi-disciplinary field of study, aimed at (in general terms) understanding how people interact with computers and to what extent computers are or are not developed for successful human interaction; and, particularly in the case of this study, how that manifests itself in regards to information technology. It deals with the design, evaluation and implementation of interactive computing systems. To do this, the human factors that determine effective and satisfying use are identified and the knowledge gained is applied to the design of more humanly acceptable technology. (Hewett et al., 1992)

Human-Computer Interaction is the study of how people design, implement, and use interactive computer systems, and how computers affect individuals, organizations, and society. HCI is a research area of increasingly central significance to computer science, other scientific and engineering disciplines, and an ever expanding array of application domains. This more prominent role follows from the widely perceived need to expand the focus of computer science research beyond traditional hardware and software issues, to attempt to better understand how technology can more effectively support people in accomplishing their goals. (Myers, Hollan, Cruz, 1996)

> HCI encompasses a large interdisciplinary domain, comprised of apsects from several disciplines, each having different emphasis:
>
> - computer science (application design and engineering of human interfaces)
> - psychology (the application of theories of cognitive processes and the empirical analysis of user behavior)
> - sociology and anthropology (interactions between technology, work, and organization)
> - and industrial design (interactive products)
>
> In correlation with the field of computer science, the other remaining disciplines serve as complementary disciplines. This brings to bear the realization of the fact that systems design problems exist in a context, and that the overly narrow optimization of one part of a design can be rendered invalid by the broader context of the problem. Consequently, even from a direct computer science perspective, it is advantageous to frame the problem of human-computer interaction broadly enough so as to help safeguard against the classic pitfall of systems design divorced from the context of the problem.

One important, and perhaps menacing, HCI factor is that different users invariably express different conceptions and form varying mental models about their interactions, and have different ways of learning and retaining knowledge and

skills. That is to say, people possess different “cognitive styles", as in, for example, "left-brained" or "right-brained" people.

In addition, there are the implications and influences of cultural diversity and national differences to be considered.

Another challenge in HCI design is that user interface technology changes rapidly, offering new interaction possibilities to which previous research findings may no longer apply. And also, user preferences change as they gradually master new interfaces.

It becomes, therefore, an intrinsic necessity to explore and understand how to come to a decision on the functionality a system will have, how to represent this to the user, how to build the system, and how to test the design. Thus, human-computer interaction includes science, engineering, and design aspects.

In the study of communication between man and machine, HCI elicits knowledge from both these domains (man and machine). Technical considerations include operating systems, programming languages, techniques in computer graphics, developmental environments, engineering and design methods. Revelant human aspects include communication theory, linquistics, graphic and industrial design disciplines, social sciences, cognitive psychology and human performance.

Concerns addressed by human-computer interaction include:

- the joint performance of tasks by humans and machines
- the structure of communication between human and machine
- human capabilities to use machines (including the learnability of interfaces)
- algorithms and programming of the interface itself
- engineering concerns that arise in designing and building interfaces
- the process of specification, design, and implementation of interfaces
- design trade-offs

Chapter 3.3 Cognitive Sciences

UCLA's Cognitive Sciences division (2001) defines Cognitive Science as a discipline that "is concerned with learning how aninmals (and machines) acquire knowledge, represent that knowledge, and how they manipulate those representations".

Cognitive Psychology derives from attempts to study sensation experimentally at the end of the 19th century. In the 1950's, an infusion of ideas from communications engineering, linguistics, and computer engineering led to an experimentally-oriented discipline concerned with human information processing and performance. Cognitive psychologists have concentrated on the learning of systems, the transfer of that learning, the mental representation of systems by humans, and human performance on such systems.

Cognitive Sciences is interdisciplinary. In order to understand the mind, the correlation between various different domains must be explored. This exploration brings together a host of practitioners from varying disciplines: psychologists, linguists, philosophers, computer scientists, electrical engineers, and mathematicians.

All of the disciplines contributing to Cognitive Sciences share in the goal of theories of cognition. And relative to this study, ULCA says, "for computer scientists and engineers, the goal is expressed in terms of the ultimate construction of robots [computers] capable of perception, coordinated motion, learning, language, and high-level reasoning. It is now abundantly clear that these goals are intimately intwined."

Examples from two disciplines:

Engineering

The European Institue of Cognitive Sciences and Engineering (EURSCO) (2001) is involved in several research projects dealing with cognition. Regarding their "Cognition in Design", they write:

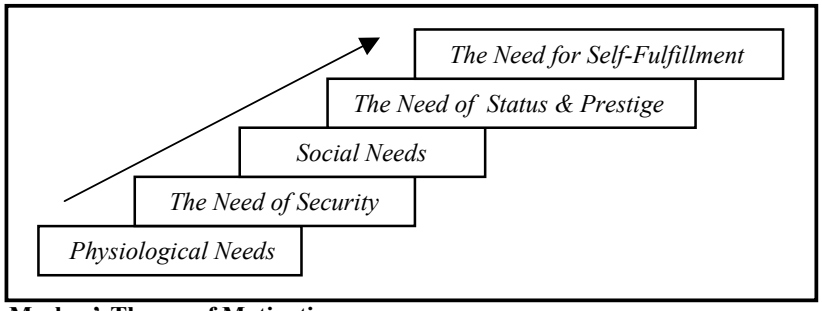
> The art of designing is to increase the number of constraints until a solution emerges. The most frequently used constraints are usually technical and economical. Constraints related to man-machine interavtion should be taken more inte account. .....It is better to implement an ergonomical design to better understand where constraints related to HMI [HCI] come from. ..... EURISCO is currently testing the Cognitive Function Analysis (CFA) method within this framwork. .....If a system is well designed for the user from the start, the number and the criticality of human errors can be significantly reduced, and human adaptation to the machines can be enhanced.

Psychology

A. H. Maslow's *A Theory of Motivation* (1943) states that all humans have five (5) primal needs. His theory is premised on a 5-tier hierarchy, each tier representing a particular need, which influences our behavior. Theory: The need allocated to a higher-ranked tier is not assessed or activated (and therefore does not influence behavior), until the need allocated to the tier beneath it is satisfied.

With 1) representing the lowest tier in the hierarchy, the five primal needs are:

1) Physiological Needs, 2) The Need of Security, 3) Social Needs,
4) The Need of Status & Prestige, and 5) The Need for Self-Fulfillment



Maslow'sTheory of Motivation

A discussion on Cognitive Processes and Motivation, in Jacobsen and Thorsvik's *Hur moderna organisationer fungerar* (*How Modern Organizations Function*) (1998), presents the Theory of Expectation. In citing V. H. Vroom (1964) et al, they write (English translation):

> The Theory of Expectation studies the reasons behind great achievements. It is assumed that one's behavior reflects one's choice of goal and subsequent behavior one believes will result in that goal being achieved. Motivation is regarded as a function of the expectation that a certain behavior will achieve a result which the individual values and desires.
>
> The main factor in the Expectation Theory is that humans are motivated to achieve a goal, if they: 1) value the goal, and 2) can ascertain that the goal is obtainable.

The value of the goal x the expectation that goal is obtainable = MOTIVATION

Motivation Formula

From these two examples alone, the significant and invaluable contributions of Cognitive Science to the concept of user participation in systems development are clearly illustrated.

CHAPTER 4

THREE LEADING SYSTEMS DEVELOPMENT METHODOLOGIES

CHAPTER 4 THREE LEADING SYSTEMS DEVELOPMENT METHODOLOGIES

4.1 What is the Rational Unified Process (RUP)? (Kruchten, 2001)

4.1.1 Definition

A web-based software engineering process which provides guidelines and support for software development organizations, RUP employs an iterative development process. It is component-based and uses an object-oriented approach to software engineering. RUP has a process framework which allows software development organizations to configure the process to suite their specific requirements, which makes it suitable for a wide range of projects and organizations. RUP provides specific guidance on how to apply the industry's best practices[1] to produce high-quality software that meets the needs of its end users, within a predictable schedule and budget.

4.1.2 History

Rational Unified Process was created by Grady Booch, James Rumbaugh and Ivar Jacobson (also the creators of UML), developed from Jacobson's Objectory (a.k.a. Object Oriented Software Engineering or OOSE) Method from the early 1990's (Jacobson, 1996). The cornerstone of the OOSE, and now the RUP, is the Use Case. A Use Case is a form of Hierarchical Task Analysis which has found favor in the software engineering community.

4.1.3 The Architecture of RUP (Ambler & Constantine,2000 & 2001)

Similar techniques used in software design are used in the structuring of RUP. In particular, the design is supported by an object-oriented model, using UML (Unified Modelling Language).

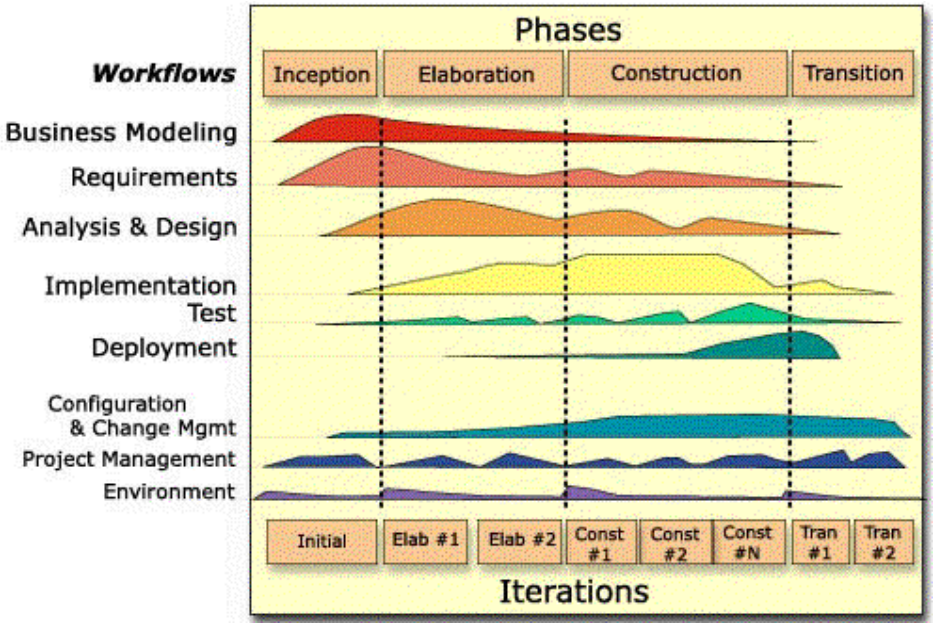
The RUP lifecycle process has two dimensions: one managerial and one developmental.

- The horizontal axis, depicts the project time frame and the four phases of the lifecycle (the managerial dimension).
- The vertical axis, lists the different processes used throughout the time frame and the correlation between them in correspondence to each phase (the developmental dimension).

Using the RUP, a software product is developed by incremental iterations. This allows for design refinements early in the lifecycle. This is reflected in the dynamic nature of the first (horizontal) dimension, which is referred to in terms of *cycles, phases, iterations*, and *milestones*. The second (vertical) dimension represents the static aspect of the development process, stated in terms of *process components*: activities, disciplines, artifacts, and roles.[1]

---

[1] See Appendix



The Two Dimensions of the RUP

The RUP is divided into four lifecycle phases. From the managerial standpoint, the goal is the development of a system, or a new version of a system.

I THE INCEPTION PHASE

Inception Tasks:

- Description of initial requirements
- Develop and justify business case for the system
- Determine scope of system
- Identify people, organizations, and external systems that will interact with the system
- Develop initial risk assessment, schedule, and estimate
- Configure initial system architecture to meet exact needs

II THE ELABORATION PHASE

Elaboration Tasks:

- Produce proven architectural baseline for system
- Evolvution of requirements model to "80% completion point"
- Draft coarse-grained project plan for entire Construction Phase
- Ensure that critical tools, processes, standards, and guidelines have been put in place for Construction phase
- Understand and eliminate high-priority risks of project

III THE CONSTRUCTION PHASE

Construction Tasks:
- Describe remaining requirements
- Flush out design of system
- Ensure that system meets needs of its users and fits into organization's overall system portfolio
- Complete component development and testing, including both the software product and its documentation
- Minimize development costs by optimizing resources
- Achieve adequate quality as rapidly as possible
- Develop useful versions of system

IV THE TRANSITION AND PRODUCTION PHASE

Transition Tasks:
- Test and validate complete system
- Intergrate with existing systems (operate system in parallel with any legacy systems to be replaced - if applicable)
- Convert legacy databases and systems to support new release
- Train the users of new system
- Deploy new system into production

Transition Artifacts to be produced:
- Final product baseline (also known as a production baseline) of the system
- Training materials for the system
- Documentation, including user manuals, support documentation, and operations documentation

The Transition portion of this phase is concluded with the Produce Release milestone. To pass this milestone, you must show that your users are satisfied with the system and that the actual expenditures, versus the planned expenditures, are still acceptable.

Production Tasks:
- Operate new system and support end-users working with it
- Monitor system, ensure continued operation
- Operate and maintain relevant jobs, logs, and supporting systems
- Respond to help requests, error reports, and feature requests by end-users
- Manage change control process so that defects and new features may be prioritized and assigned to future releases

Production Artifacts to be produced:
- Software Problem Reports (SPRs) summarizing potential defects or new features for future releases
- Problem resolution strategies to be followed by end-users requiring support
- Appropriate metrics summarizing system usage, system performance, and end-user satisfaction

The Production portion is concluded with the System Replacement milestone. To pass this milestone, you must achieve one of the following:

- A new version of the system is deployed into production
- The system is replaced by a new one
- The system is removed completely from production, an event called sunsetting
- Systems development organization ceases system operations

Key for the developmental aspect of the lifecycle is the goal of generating a progressively more well-defined version of the system under development, by means of a number of iterations.

The activities performed during these iterations are grouped into a set of Core Workflows. The task of each core workflow is to produce a description of some aspect of the system, be it a model of the system, or system documentation.

The Five Core Process Workflows (Ericsson, 2001) are:

Building Models - assessing the organization, its problems and needs, in order to ascertain the specific system requirements. It is here a business use case model and a business object model are developed. This workflow, however, is considered optional, and comes into use primarily if there are specific complexities in the organization that require exploration.

Requirements - focusing on usability, an evaluation is made of system requirements. Here the Use Case Model is produced; including actors, representing external entities (persons or things) which communicate with the system, and use cases, representing transaction sequences which yield value to the actors, that can be measured.

Analysis and Design - here the implementation environment is investigated. Its effect on system construction is also evaluated. An object model is built, which includes use case realizations that depict how the objects communicate within the use cases.

Implementation - here the system is implemented in the implementation environment. Source codes, executables, and files are produced.

Test - ensures that the system is as intended, and that the implementation is successful and complete. This produces system certification, deeming the system ready for delivery.

4.2 What is the Soft Systems Methodology (SSM)? (Avison & Fitzgerald, 1997; Lewis, 1994; Underwood, 1996)

4.2.1 Definition

A definition of soft systems methodology is proceeded by definitions of several key terms which form the basis for this methodology.

"Hard" problems are those problems, in systems design, which can be well-defined. The assumption is that there is a definite solution, and that a number of specific goals, that must be accomplished, can be defined. In essence, regarding hard problems, you can define the end product prior to commencing to implement the solution; "the 'WHAT' and the 'HOW' of a hard problem can be determined early in the methodology process"(Couprie et al,2001).

"Soft" problems, in contrast, are difficult to define. They contain a large social (organizational culture) and political (organizational power structure) component. These problems are not expressed as 'problems', as such, but as 'problem situations'. "When we think of soft problems, we don't think of problems, but of problem situations. We know that things are not working the way we want them to, and we want to find out why and see if there is anything we can do about it. It is the classic situation of it not being a 'problem' but an 'opportunity'"(Couprie).

Soft systems thinking attempts to understand the complex nature of IT systems.

Soft Systems Methodology (SSM) offers a specified approach to the analysis of complex problems. It is a methodology that investigates the different viewpoints and perceptions that people, in an organization, can have on a problem. SSM is based on the modelling of viable systems, from these various vantage points (views). Models of these "human activity systems" are then used to structure debate and to form a concensus about the need or desire for organisational change, the outcome of which leads to the development of a software system that fits the express needs of that organization, both systemically and culturally. The emphasis in SSM is on the understanding of problem situations, rather than on developing solutions.

4.2.2 History

Soft Systems methodology was created and developed by Peter Checkland, a professor and researcher in Software Engineering, for the express purpose of dealing with soft problems. Checkland had been in industry for a number of years and had been working with several hard system methodologies. He saw how these methodologies were inadequate in dealing with extremely complex problems which had a large social component, so in the 1960's, he turned to the University of Lancaster, in the UK, in an attempt to research this area and deal with these so-called "soft" problems.

Checkland's "Soft Systems Methodology" was created through a number of research projects in industry, and its application and refinement evolved over a number of years. The methodology (which is pretty much how we know it today) was published in 1981. (An alternative version was proposed in 1990. It was based on the results

of further action research. This later version promotes a looser interpretation of the methodology.) (Avison & Fitzgerald, 1997)

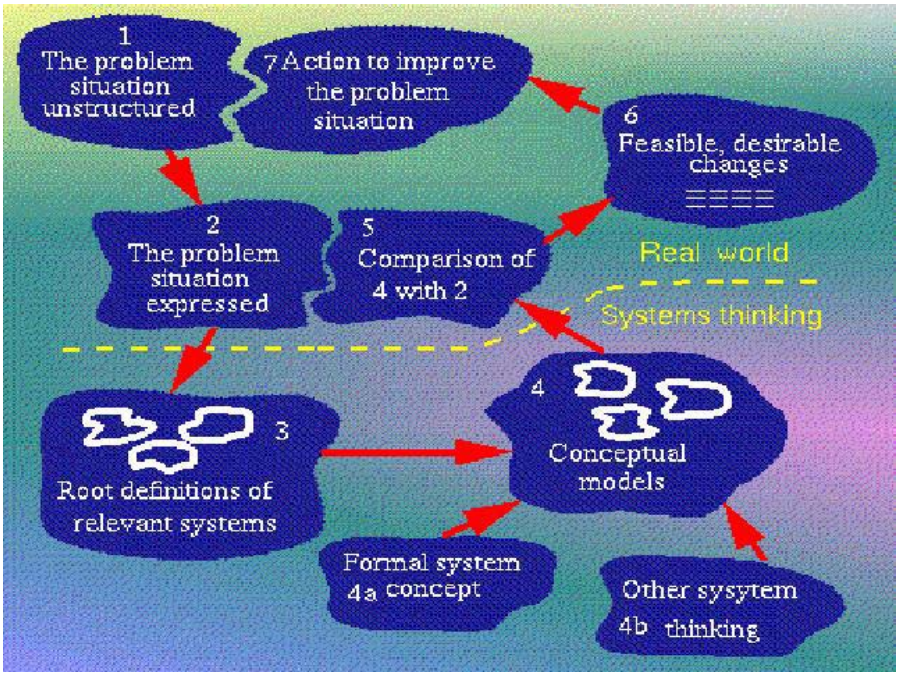
4.2.3 The Architecture of SSM

The core of SSM is the modelling of different perspectives on a problem, with the aid of *Rich Pictures*, *Root Definitions*, and *Conceptual Models*.

SSM is divided (Couprie) into seven distinct stages. These are:

1. Exploring and formulating the problem situation
2. Expressing the problem situation through Rich Pictures
3. Selecting how to view the situation and producing Root Definitions
4. Building conceptual models of what the system must do for each root definition
5. Comparison of the Conceptual Models with the real world. (Comparing the results from Stages 4 and 2 to see where they differ and/or are similar.)
6. Identification of desirable and feasible changes
7. Recommendations for taking action to improve the problem situation. (How to implement those changes identified in Stage 6)

SSM employs an iterative approach. The process is likely to be iterative both around and between stages. Circumstances may dictate several iterations of these seven stages in order to produce satisfactory results. With SSM, the process is as important as the outcome. It is during this process that the organization itself may experience change, as a result of the formulation and expression of various views.



Model of SSM

Stage 1 Statement of Problem Situation

Stage 1 is basically that people in the organization think there might be a problem, or room for improvement, and initiate an analysis or review. The initial stage consists of managers and/or employees (problem-owners) perceiving a potential need that a decision regarding a review or change of tasks - and the way they are performed - is required, and an analyst (problem-solver) is then called in. That is to say, the people within the organization, themselves, decide there might be a problem, or room for improvement, and initiate the analysis or review.

Stage 2 Analysis of Problem Situation

In Stage 2, the analyst collects and organizes organizational data in order to provide a description of the problem situation. The three primary types of data being sought are:

- the physical structure of the organization: those factors that do not change easily (i.e.buildings, locations, environments)
- processes or transformations which occur within the situation (many involve constant change)
- issues that are expressed or felt by organizational members (complaints, criticisms, suggestions, endorsements)

The analyst has a wide range of strategies and techniques at his/her disposal when collecting data:

WORK OBSERVATION

- identify tasks performed
- identify tools employed
- establish interactions between people/systems
- produce logs
- "day-in-the-life-of" descriptions
- make drawings of structures/layouts
- video recordings
- collect samples of tools used to handle information
- perform participant observation

INTERVIEWS

- unstructured (informal narratives)
- semi-structured (questionnaire w/ open-ended answers)
- highly structured (questionnaire w/ multiple-chose answers)
- critical incidents
- audio recording

WORKSHOPS AND DISCUSSION

- future workshops
- review workshops
- conflict resolutions workshops
- mock-ups; simulations; mind-games

Stage 1 and Stage 2 are known as the 'problem expression' phase, during which an attempt is made to build the richest possible picture, not of the 'problem' but of the 'situation' within which a problem is perceived.

When an analyst elicits information from the members of an organization, s/he communicates with them by means of natural language (i.e. English). "This poses a number of problems and potential pitfalls. The analyst should be prepared to accept that at this stage, the information elicited will be incomplete, and will contain contradictions and ambiguities. The system which we are looking at is a soft system, and therefore the information about the system is likely to be qualitative, rather than quantitative".

The Use of Rich Pictures

Rich pictures are diagrams used to provide a reference model of the system and to help the analyst gain an appreciation of the problem situation. It is important to note the difference between rich pictures and formal models. A rich picture does not attempt to present a model of the system in any precise way. Rich pictures should represent the structure, processes, and issues of the organization, which could be relevant to the problem definition. They should "try to give an impression of the organizational climate", which provides a representation of how we relate to and view the system. And that representation can be refined as the analyst's understanding of the system evolves and becomes clearer.

The analysis to be performed on a rich picture is comprised of the following:

- Roles of Intervention Analysis - identifies the issues that people involved in the situation regard as problematic

- Social Analysis - identifies the roles people play in the organization, the norms of behaviour those people display, and the values by which their behaviour is judged

- Power Analysis - is concerned with such issues as "What are the commodities of power in this situation?", "How is the commodity obtained?", and "How is the commodity passed on?"

"The problem-owner's help is the input of the process. The problem-solver will perform analysis on the soft system and end up with a rich picture as output of this transformation process. The analyst will use the rich picture to aid their communication with the problem-owner. In addition, he or she will notify the conflict he observes regarding personnel or function. The rich picture is used to identify problems and to inform the problem-owner of the situation, rather than provide possible solution."

Stage 3 Relevant Systems

In this stage, the analyst makes a selection of those systems, which s/he believes best shed light on the problem situation. This selection of systems is expressed in statements as the Root Definitions.

Root Definitions

A root definition is a well-defined statement concerning an area of activity and its components. It confirms both what is agreed upon and what has yet to be resolved. A root definition can be likened to a mission statement, though it is intended for internal organizational use only.

Important attention is to be paid to the development of root definitions. "Properly written root definitions provide a much simpler insight into building system models."

A root definition is expressed as a transformation process that takes an entity of some sort as input, changes or transforms it, producing a new form of the entity as output. "A prescription for developing transformation processes is shown in the following table, which shows examples of transformations which are typical of a golf course operation. As you may notice, these transformations will vary greatly, depending on the world view that is applied."

INPUT	OUTPUT	AS VIEWED FROM THE EYES OF:
Unused land	Land occupied by golf course.	Architect.
Need for tee times.	Need for tee times is met.	Club Management.
New golf balls.	Used, scuffed up golf balls.	Equipment industry.
Grass seed	Mature grass.	Greenskeepers
Uncooked food.	Quality meals.	Kitchen cook.
Registered golfer.	Golfer who completed round in X strokes.	Pro shop staff.
Golf lesson program.	Enhanced lesson program.	Club Professional.

One-to-One Transformations Involving Different World Views

Producing a root definition is a two-step process:

1 An issue or task is selected from a rich picture

2 A system is defined to address the issue or perform the task

"Each root definition involves two important things. The first being, we must involve a certain view of the world. Definition of the world view is not always trivial. Also, not all world views may be desirable to the definer. Remember that each rich picture will involve a variety of world views. The 'eyes' may come from sources such as government officials, company executive, project managers, employees, customers, competitors, and news media. Each of these world views will be linked to one or more distinct root definitions.

[Secondly], it is important to pay attention to the cardinality of the transformation process. Each root definition involves a transformation of one input to one output. Suppose we define a transformation as: 'Golf equipment' *plus* 'golf course' *plus* 'man-power' (three inputs) *yields* 'golf needs met' *plus* 'golf market served' (two outputs). This 'three to two' transformation is ambiguous, but can be resolved into many one-to-one transformations that look much clearer (i.e. golf equipment is transformed into used golf equipment)."

CATWOE

Root definitions are written as statements that elaborate on a transformation. Well-defined root definitions are crucial to the creation of the conceptual model, in Stage 4. Subsequently, root definitions are tested against a group of six elements, which are summed up in the mnemonic: CATWOE.

- **C**ustomer - everyone who stands to benefit from a system is considered to be a customer of the system. If the system involves sacrifices such as lay-offs, then those victims must also be counted as customers.
- **A**ctor - the actors perform the activities defined in the system.

- **T**ransformation process - this is shown as the conversion of input to output.
- **W**eltanschauung - the German expression for world view. This world view places the transformation process in the broader context external influence, questions about adaptability, etc.
- **O**wner - every system has a proprietor who has the power to start up and shut down the system.
- **E**nvironmental Constraints - inherent elements which exist outside the system. These constraints include organizational policies, as well as legal, political, and ethical matters.

CATWOE is used mainly for the analysis of root definition statements, but may be used as a building block (during Stages 1 and 2) in deriving the root definition statements, themselves, if the CATWOE elements are known.

Stage 4 Conceptual Models

Having arrived at a root definition of a system, a conceptual model of the proposed system can then be drawn. A conceptual model is a representative "human activity model" that strictly conforms to the root definition, while holding the use of activity sets to a minimum. Systems thinking is applied in this development.

Systems Thinking

Systems thinking is an iterative process that combines three concepts:

- The Perceived World: Each one of us has our own views of the world.
- Ideas: We perceive the world through the framework of ideas that are internal to us.
- Methodology: There are many of these for thinking about the world, of which SSM is one.

Originally, the model was constructed using the Formal Systems Model, prescribed by Checkland, which contained a lengthy list of criteria to be satisfied. However, Checkland has recently dismissed the concept of a formal model and now "prefers a looser, more flexible" 5-point criteria for creating the "ideal" system (Jarvis, 2001).

Checkland's Five E's for Selection Criteria

- Efficacy - Will it work at all?
- Efficiency - Will it work with minimum resources?
- Effectiveness - Does it contribute to the enterprise?
- Ethicality - Is it moral?
- Elegance - Is it beautiful?

Stage 5 Comparing Conceptual Models with Reality

In this stage, conceptual models built in Stage 4 are compared with the real world problem situation formulated in Stage 2. "The work at this stage may lead to the re-iteration of Stage 3 and Stage 4. Based on the rationale of this methodology, there are four ways of making comparison from number of experiences." (Couprie)

Four ways of making comparison can be summarized as follows:

1. Using Conceptual Model as a Basis for Ordered Questioning
This type of comparison may be appropriate when the real world situation differs greatly from the conceptual model. The system models are used to foster debate about change. The model is used as a source of questions to ask of the existing situation. The questions are written down and answered systematically. The answers can provide illumination of the perceived problem.

2. Comparing History with Model Prediction
Another method of comparison is made by re-constructing a sequence of past events and comparing what had transpired in producing the sequence with what would have happened if the relevant conceptual model had actually been implemented. Checkland warned that this method of comparison should be used carefully, so that it may reveal the inadequacies of the actual procedure and it can be interpreted as offensive recrimination concerning the past performance.

3. General Overall Comparison
Checkland suggested that in the illustration of the methodology as a whole, it is usual that model comparison is a general one, inquiring what features of the conceptual model are especially different from the existing system, and why. This comparison is also generally discussed with Checkland's "WHATs" and "HOWs". It is the distinction between 'whats' and 'hows' that makes the term *comparison* a somewhat crude description of what is happening in Stage 5. Checkland points out that at Stage 5, we have systems models available, which themselves are derived from the careful naming (in root definitions) of human activity systems, which we hope are relevant to the problem situation and to its improvement. In Stage 5, we examine the models alongside the expression of the problem situation assembled in Stage 2. The comparison between the two is the formal structure of a discussion about possible changes, a discussion held with concerned people in the problem situation. In order to make the discussion rich and wide-ranging, we wish to question whether or not various activities in the models are discernible in the real world, as well as - if they are present - how well they are being carried out. We also wish to discuss possible alternatives to the real world activities.

4. Model Overlay
The fourth method of doing Stage 5 is referred to as "model overlay" by Checkland. For the comparison, after completing conceptualization based on the chosen root definition, a second model is constructed – one depicting

the existing system. The second model is to replicate the conceptual model, as near as possible; the aim being to re-draw that model, altering it only where reality differs from the conceptual model. The direct overlay of one model upon the other then reveals the mismatch that becomes a catalyst for the discussion of change.

All four methods can aid in ensuring that the comparisons performed in Stage 5 are conscious, coherent and defensible. Depending on the perceived problems, one, or a combination, of these methods can be utilized. In the case of constructing completely new systems, previous experience implies that incrementation and trial-and-error is the best approach.

Stages 6 Identification of Feasible and Desirable Changes

In Stage 6, feasible and desirable changes are identified and discussed. The purpose of the proceeding comparison stage is to generate debate about possible changes that might be made within the perceived problem situation. This can be seen clearly with the second method of comparison, as discussed above.

The result of Stage 6 and 7 for both hard and soft systems is the creation and implementation of a system. Generally, in more nebulous problem situations, the eventual action is likely to be less than the implementation of a system, it is more likely to to be the introduction of a more modest change.

Normally, there are three kinds of changes:

- changes in structure - changes made to those parts of reality which in the short term (in the on-going running of things) do not change
- changes in procedure - changes in the dynamic elements
- changes in attitude - behaviour appropriate to various roles, as well as changes in the readiness to rate certain kinds of behaviour "good" or "bad", in relationship to others

Changes in structure and procedure are easy to specify and relatively easy to implement. Such changes can be made by those having authority or influence. However, it is relatively difficult to change *attitudes*, though it is possible, in principle, to try and bring about changes of this kind. Whether or not this is attempted, the main issue is to continuously monitor attitude, if changes are to be made in situations perceived as problems, so that affected persons in the situation agree that improvement has been achieved. One of the important features in SSM is it emphasis on *change*.

Another important feature of SSM is that it is "goal-driven"; it focuses on a desirable system and how to achieve it. Checkland indicated that the changes must be systemically desirable as the result of the insight gained from the selection of root definitions and conceptual model building, and that they must also be culturally feasible, given the characteristics of the situation, the people in it, their shared experiences, and their prejudices. It is difficult to find any changes that do not meet both criteria. Checkland found, in one of his case studies, that it is

important to move quickly and lightly through all the methodological stages, several times, if necessary, in order to engineer a "bridge" between 'what is' and 'what might be'. He also suggested that we may have to incorporate "root constraints", in order to find compromise in a situation where proposed changes have to be revised, due to the power influence.

Stage 7 Action/Implementation

The task in Stage 7 is to implement changes and put them into action. When action is taken, it might be straightforward. However, other situations may be encountered. The introduction of the action may change the situation so that, although the originally perceived problem has been eliminated, new problems emerge. It is often recommended that a temporary system be used to carry out the task, under the supervision of the analyst, followed by a transition to the operation of the new system.

4.2.4 Areas of Application

The approach has been widely used in both industry and the public sector. For example, according to the Lancaster University Management School (2001), SSM has received substantial application in the United Kingdom's National Health Service, especially in an evaluation methodology of the Resource Management Initiative, and in developing information strategies for both District Health Authority 'purchasers' and hospital 'providers' under the new NHS structures.

"Research in this area has been focused on the development of the ideas, concepts and philosophy underpinning SSM, in addition to the study of its applications in specific domains. In addition to the health service area, other application domains include the general area of project planning, and extensive application in the area of computer-based information systems."

4.3 What is the Dynamic System Development Method (DSDM)? (Stapleton, 1997; DSDM Consortium, 2001)

4.3.1 Definition

DSDM is perhaps not a method, in the stricter meaning of the term, but rather a framework of controls and supplementary usage guidelines for Rapid Application Development (RAD). In the broader sense, DSDM is a method, as it defines a process and a set of products. It is, however, a method with a flexibility that allows it to be tailored to meet any technical or business environment. The method prescribes no techniques, but rather suggested paths of implementation for both structured and object-oriented development approaches.

4.3.2 History

In the early 1980's, a new process called Rapid Application Development (RAD) was introduced into the information technology industry. RAD was intended as an alternative to the standard "waterfall"-model methods approach to system development. A concensus had arisen among both systems users and developers that there was need for a method approach that addressed the desire and need for more flexibility in the development cycle and opportunity for swifter product delivery.

The popularity for this new iterative design process grew throughout the decade, but there was no commonly agreed-upon definition of what exactly this process encompassed. This understandably resulted in a myriad of interpretations and approaches in use.

In urgent need of a standard structure for the RAD method, the DSDM Consortium was born. In January 1994, the founding members of the DSDM Consortium met, and over the next three months jointly developed and approved a public domain RAD method. While the DSDM framework of the method has been further developed and refined over the years, the basic concept has remained the same.

4.3.3 The Architecture of DSDM

What DSDM offers is a generic process, which enables and allows an organization to tailor it to its specific business requirements and technical restraints. The life cycle is iterative and incremental: the fundamental premise being that a usable and to 80% completed system can be developed in 20% of the time it would normally take to produce a total system, using a waterfall approach. Iteration allows for correction of inaccuracies and flaws in the design process or new changes that may appear during the process. Time and resources can be utilized more effectively.

The DSDM development lifecycle has five phases:

- The Feasibility Study - an assessment of whether or not the DSDM approach is the appropriate one for the project: DSDM is used for rapid

system development. Besides the feasibility report, an outline for development, and, in some cases, an optional fast prototype, is prepared. The feasibility study typically lasts a matter of weeks (rather than months).

- The Business Study - a short study of business and technical constraints, processes being automated and their information needs, providing the foundation for all subsequent work that is to follow. By means of workshops, the Business Area Definition (which includes identification the classes of users of the new system) is produced. The high-level functions identified in the Business Area Definition has to be prioritized, agreeing on the essential system functionalities to be developed, while less essential parts can be added later, if necessary. The system architecture, is essence, is outlined here, as well as a refinement of the prototype developed in the previous study, called the Outline Prototyping Plan. As in the Feasibility Study, the Business Study is a short process; taking less than a month.

- The Functional Model Iteration - focus here is on the refinement of the business aspects of the system, using the findings of the Business Area Definition. Both the functional model iteration and the design and build iteration consist of cycles of the following activates:

    1. Identify what is to be done in the cycle
    2. Agree on how it is to be accomplished
    3. Do it
    4. Check that it was done correctly (by review of documents, demonstration of prototype, or test of the software)

    Other outcomes resulting from this iteration stage are:

    - Prioritized Functions - definition of core functionality guaranteed to be delivered to the users at the end of current increment
    - Functional Prototyping Review Documents - comments by users, after iterative reviews of system
    - Non-Functional Requirements - majority of which are dealt with during the design and build iteration
    - Risk Analysis of Future Development - a timely key document for identifying risks and countermeasures

    (The bulk of the development work occurs in the two iteration phases, where prototypes are incrementally evolving, aided by system testing.)

- The System Design and Build Iteration - this phase is to ensure that the system is of adequately high standard for safe delivery into the hands of the users. The primary product here is the tested system. The tested system may not necessarily satisfy all requirements discovered during development, but it will satisfy the core (minimum) requirements agreed upon, plus as many as time allowances permits.

- The Implementation - covers the transfer of the system from the developmental to the operational environment; including necessary documentation (including User Manual and Project Review Document), end-user training and system hand-over. The Project Review Document summary includes each four possible outcomes:

    1. All requirements satisfied; no further development needed
    2. A major business functionality was discovered during development, which could not be addressed due to due-date time constraints; a return to business study phase to further development.
    3. Lower prioritized functionality, omitted due to time constraints, can now be added on; a return to function model iteration phase for development.
    4. A non-functional requirement, omitted due to time constraints, can now be added on; a return to design and build iteration phase for development.

There are 9 cohesive underlying principles that form the foundational basis for DSDM:

Principle 1	Active user involvement is imperative - the user involvement in a DSDM project is pro-active. As opposed to traditional approaches, a group of knowledgeable users support and participate throughout the development process. This results in better and shorter communication lines, which in turn saves time and resources.
Principle 2	DSDM teams must be enpowered to make decisions - the ability to make quick decisions is paramount. The level of functionality, usability, detail alteration are but some of the issues that should be handled without authorization from higher-level management.
Principle 3	The focus is on frequent delivery of products - a product-based approach, as opposed to activity-based, facilitates higher affectivity within agreed upon timeframes. It also allows for verification of developmental progress by staff/management outside the team.
Principle 4	Fitness for business purpose is the essential criterion for acceptance of deliveries - keeping the focus on delivering those requirements agreed upon and on quality assurance.

Principle 5	Iterative and incremental development is necessary to converge on an accurate business solution - crucial to the evolutionary process in which continual and near-instant feedback from users help site flaws and errors early in the process.
Principle 6	All changes during development are reversible - DSDM facilitates backtracking and course correction, when necessary.
Principle 7	Requirements are baselined at a high level - freezing the requirements agreed upon in the business study phase at a level that keeps them in focus throughout the iterative process.
Principle 8	Testing is integrated throughout the lifecycle - testing is not relegated to the end of the development process; a ”test-as-you-go” philosophy is adopted. Incremental testing and reviews by both developers and users assures that product development stays on track.
Principle 9	A collaborative and cooperative approach between all stakeholders is essential - this includes not only system developers and users, but even peripheral entities in such areas as IT operations, procurements, and/or external suppliers.

Timeboxes

There are currently several definitions of timeboxes in use. One of the prime definitions is, the time between the start and end dates of a project. The end date is fixed and the system is to be delivered by that date. DSDM takes timeboxes to the next level and designates timeboxes as checkpoints within the overall timebox for a project. This provides for incremental deadlines by which some aspect of the project will be delivered; be it an analysis model or any other aspect, whatever moves the project nearer to its completion date.

In keeping with the idea of speedy implementation, DSDM recommends the length of timeboxes be set to between two to six weeks; the shorter, the better. However, this is to be viewed merely as a guideline. The inference is to get things done. This reflects the premise of Principle 3, which emphasizes focus on frequent delivery of product.

MoSCoW Rules

Not found in the DSDM manual, these rules have been adopted by many organizations using DSDM as a way of prioritizing requirements in a RAD project. They were derived by one of the early participants in the DSDM Consortium, Dai Clegg, of Oracle U.K.

MoSCoW is an acronym, denoting the priority assigned to certain requirements in the design process:

> '**Must have**' - for those requirements fundamental to the system, without which the system would be rendered inoperable.
>
> '**Should be**' - requirements which probably would be mandatory in a development project not under time constraints, but without which the system would still be usable and useful.
>
> '**Could be**' - signifying requirements that could more readily be omitted from the current increment.
>
> '**Want to have but will not have this time around**' - for those valuable requirements that can be put off until further development takes place.

Project Roles

A unique aspect of a DSDM team is that it is made up of systems developers and end-users, working together. Teams are to be kept small in order to facilitate shortened lines of communication between team members: a minimum of two persons, because at least one representative is needed from each domain; and a maximum of six persons, wherein it has been found that beyond that limit, difficulties arise in the RAD process. A typical DSDM project will have one or two teams, although a large project may require as many as six teams, all working parallel. Once again, the operative number is *six*, greater than which manageability become a problem.

There no distinction made between the different IT roles, except for the role of Technical Coordinator, whose job it is to define the system architecture; ensure the technical consistency of the project, including the effective use of technical controls, such as configuration management; and to ensure that all work produced is of sufficient technical quality.

Being a user-centered approach to system development, DSDM designates several roles for system users; both as part-time advisors, and as participants within the project team.

The key user role within the project team is that of Ambassador User. Ambassador Users operate in much the same way as diplomatic ambassadors. They are responsible for bringing the knowledge of the user community to the team, and for disseminating information from the team to the rest of the users. They are not just an information channel between developer and user, but rather, they themselves come from the community which will eventually use the system.

Another key user role is that of Visionary. The Visionary may not be the financier and top decision-maker - that is the role of the Executive Sponsor -

but s/he is probably the initiator of the project through their vision for IT development in their business domain. The Visionary is involved from an early stage in the process (during the feasibility and business studies), helping to make decisions on what's important to the system and what's not, and s/he is also a participant in key demonstrations and team meetings, to ensure that the team remains focused on the original objectives of the project.

Yet another user role is designated by DSDM: the role of Advisor User. The Advisor User is a complementary role to that of Ambassador User, whose function is to present those user views which perhaps were missed or overlooked by the Ambassador User, to the project team. The Advisor Users may be anyone with an interest in the final product; an IT staffer or the system administrator, just to name two. Advisor Users participate on an ad hoc basis, dependent on the project needs.

Common for all these roles is the need for representatives skillful in effective communication. They must be capable of competent expression of their own needs, as users, and the vision of the business.

CHAPTER 5 CONCLUSIONS

Not being particularly familiar with all the specific aspects of the methodologies selected for this study, the research revealed that PD, RUP, SSM and DSDM provide a broad representation of the spectrum of systems development approaches available to systems engineers, at present.

Results

Though the focus of this study concerned systems development, this study also showed that the implications of User Participation extend well beyond the confines of systems design. There are psychological, sociological, and even political, as well as scientific and technological aspects to be considered. And subsequently, the role of systems developers contributes and provides more (to society) than just systems solutions.

User Participation is not a new politically-correct trend or fad. It is a true science, with decades of research and development supporting it. And as further advances in modern technology are made, the socio-technical correlation between man and machine becomes increasingly intrinsic in developing products and applications to meet the demanding requirements of evolutionary human needs and desires. For example, such associatively social attributes as ”satisfaction” and ”attractiveness” were shown by the study to be included in the definition of the term *usability*. Add to that the technical attributes ”effectiveness” and ”efficiency”, and the decisive weight that *usability* places on technology is emphasized.

Design knowledge is not necessarily the private domain of systems designs. The more education and knowledge end users obtain, the greater the demand for influence over product design. An unavoidable interdependency ensues. Therein lies one of the many complexities in the application of User Participation

The study also illustrated that User Participation is not an ”all-or-nothing” proposition. There are varying degrees or levels of user involvement that can be, and are being, employed in the design and/or development process of software systems.

An Overview of the Methodologies

The study highlighted some of the complexities and diversities in applying the concept of User Participation to projects, thus, the following overview:

> **RUP** is, in essence, a standard systems development methodology, that is based on an ”iterative”, rather than ”waterfall”, lifecycle process. No specific reference as to the use of User Participation is made. There are only the standard references made to user/stakeholder involvement, and that is during:
>
> - stakeholder identification
> - system requirements evaluation, and
> - end user training
>
> RUP is a methodology, in the true sense of the term. It cover the entire development process; from the identification of the problem area to implementation of a system

solution, and beyond. Within in the methodology there are various methods (stages of development) and descriptions of ”how-to” techniques to be applied.

SSM is premised on the science of complex problem definition in organizations, as opposed to simply user involvement in the systems development process. That being said, SSM is, in its very nature, User Participation. It is only by user participation SSM accomplishes its goals. It addresses these complexities by utilizing Checkland’s theories of ”hard” and ”soft” problems and soft systems thinking. It is the result of decades of evolution-ary research. SSM places the emphasis on understanding complex problem situa-tions, rather than finding solutions.

The methodology itself is complex. There are methods within methods, within methods. At the offset, SSM calls for an assessment, by the system’s stakeholders, of ”need”; whether or not there actually exists a problem situation requiring change. Interestingly, there is an encouragement or a caution (dependent upon one’s inter-pretation) that the SSM process may in many cases foster change within the organ-ization itself, through the acquisition of new knowledge, and through debate. There is also the complexity of language and communication, which embodies the very nature of soft problem. This, too, is addressed in SSM.

The study showed how the complex methods within SSM are approached by uncom-plicated, near simple, processes:

- Rich Pictures – which uses simple drawings to assist the problem-owners and the problem-solvers to illustrate, explain, and understand the problem situation (soft systems thinking).

- Root Definitions – which express the soft problem as statements, represented in hard systems thinking terms: input-transformation-output.

- CATWOE – which uses six simple ”elements” to analyze root statements for validity. It can also be used in first stages, if all the elements are already known.

The selection of conceptual models, to represent human activity; the modes of comparison with the real world problem situation; even, the definitions for evalua-ting change: these are all facilitated by the use of algorithms that are approachable by both end user and designer.

Soft Systems Methodology is a user participation methodology from beginning to end.

SSM is a prime example of the complexity of differentiating between methods and methodologies. While the theory of soft thinking has given rise to this methodology, the methodology is self-restricting; it limits its focus to design planning.

The emphasis, re-stated, is on identifying and understanding a perceived problem situation by presenting stakeholders’ viewpoints, and not specifically on developing strategies for solutions. Placed in the context of an entire systems development process (RUP, for example), SSM then becomes but only one of many methods that

a systems solution could possibly be comprised of. So here we have a methodology that is relatively narrow in its scope, but that is deep-layered within its own domain.

DSDM represents a third approach to User Participation. Although it calls itself a ”method”, DSDM is a full-blown user participation methodology: Not only does it make full use of the concept of User Participation, it also utilizes the concepts of incrementation and iteration to the maximum. The basis for its emergence and development was the need, among end users and designer, alike, for more flexibility in the systems development process, and swifter product delivery.

DSDM is completely configurable, as in the case of the Rational Unified Process and similar to Soft Systems Methodology, DSDM analyzes for ”need”, and makes use of user-friendly algorithms to help prioritize business requirements and thus simplify and accelerate the development process. And as Participatory Design’s ”democracy in the workplace” philosophy advocates, DSDM has among its principles, the re-positioning of the power structure from vertical to lateral, both within and without the design team. It also strives for equal representation between end users and designers.

The study showed that DSDM methodically makes use of iterations and incremental phasing (Timeboxes) within the lifecycle; the purpose being, to retain fluidity throughout the process, in order to facilitate any necessary backtracking and/or course correction; and to carry out continual system testing by end users.

In comparison with the other methodologies covered by this study, DSDM has a unique and well thought out approach for safeguarding that all end user views are heard and taken into account by the systems designers. Its use of Project Roles

- Ambassador User – who is the communications link between end users and designers
- Visionary – who is probably the project initiator, and who helps, among other things, with decision-making and safeguarding the original objectives of the project
- Advisor User – who participates on an ad hoc basis and functions as a complement to the Ambassador User, to see that no end user view is missed or overlooked

provides extra precaution to make the collaboration, communication, and cooperation between end users and systems designers as complete and successful as possible.

A look at the premises on which the studied methodologies are based

It study revealed that the primary incentive for the emergence of user participation in Scandinavian systems development was not necessarily the enhancement of system usability, as such; but rather, the limitation, by trade unions, of authority and influence of organizations over their employees: so-called ”democracy in the workplace”. This is in stark contrast to the central focus placed upon useful systems, specifically espoused by SSM or DSDM.

The more palatable socio-technical methods and methodologies of HCI and Cognitive Sciences offer a more pragmatic and less antagonistic balance between the aspirations and concerns of both users and organizations. And with the introduction and establishment of usability standards, the systems user's position as a vital factor in systems development is all the more strengthened and secured.

Also, despite the fact that these studied methodologies emerged and evolved during practically the same period of time (the last two decades, in particular), there is a striking absence of any references to possible contact or collaboration between the major proponents, such as Checkland, Nygaard, Jacobson and the like, in the research materials.

As mentioned earlier in this text, the DSDM Consortium is unique in its premise and inception. It was in answer to a perceived need for standardization of the RAD process: a true example of user participation.

Evolutionary Considerations

Over the two last decades, the tenets of user participation have evolved to the extent that many systems designers regard them as decisive components of the systems development lifecycle. The process, framed within "evaluate/design/test/re-evaluate" iterations, is both result-oriented and pragmatic; which underscores ISO usability standard's specific goals of effectiveness, efficiency, and satisfaction; the criterion by which user participation practitioner measure success.

The study shows that Participatory Design has undergone somewhat of a transformation in philosophy. The overly political rhetoric of "industrial democracy", in the 1970's, has given way to a more socio-technical approach, in which the practical interaction between user and machine, rather than power struggles between certain stakeholders (primarily trade unions and organization owners, in the case of PD), takes precedence. There is a conference; the Participatory Design Conference, which meets bi-annually to discuss the latest developments in PD.

The emergence and evolution of RUP is credited to the Rational Corporation's acquisition of Ivar Jacobson's Objectory Process, and several other products and services, when it merged with Jacobson's Objectory organization several years ago. Being web-based, RUP has regular software upgrade releases approximately twice a year, the study reveals.

SSM (mode 1) was introduced in 1981. It was developed through testing systems ideas in client organizations by doing so-called "action research". However, an alternative version (mode 2) was proposed in 1990. It was based on the results of further action research. This later version promotes a looser interpretation of mode 1. It suggests the SSM be seen as a framework, rather than as a methodology. This is the extent of the evolution of SSM, to date.

While the framework of DSDM has been developed and refined over the life of the DSDM Consortium, the basic concepts of DSDM have remained in place, since the mid 1990's.

User-centered versus Design-centered

The study reveals that the inclusion of empirical data, which end users bring to the design effort, introduces an inherently dynamic catalyst into the mechanics of the design process. Not only does this facilitate end user awareness of the potential benefits and/or limitations in a specific solution design, but also, the developers gain the opportunity to develop heightened sensibility to the intricacies of user/product interaction. An environment is created where the channeling of user requests and suggestions, and the articulation of technical and practical application capabilities and constraints converge. And in such an environ-ment, the focus, pursuant to the structuring of a viable product, must invariably turn to the issues of effective communication, rather than to the mere strict adherence to specific development methods or techniques.

RUP is a classic example of design-centered systems development. The central goals of this methodology is 1) to identify a systems solution, and 2) once the solution has been decided upon and the design model has been fixed, all efforts turn to the development and implementation of that design model. True, RUP emphasizes addressing high-risk areas early on; it also supports the rapid development of an initial version of the system solution. But even within the early modeling stages themselves, end user input or participation may well be extremely limited, if not virtually non-existent. No direct contact med end users is sited in any of the RUP lifecycle phases. The evolution of the system is based on a fixed architecture, developed early on in the process.

Even SSM, with its rich pictures, root definitions, CATWOE and conceptual models, adopts a fundamentally design-centered approach, once the problem situation has been established. Stage 5 calls for the comparison of conceptual models with the ”real world” problem situation. This sounds like the very essence of user participation, indeed, but in actuality, the model evaluation is carried out against a static representation of the problem situation, much in the same manner as in RUP.

In contrast, PD and DSDM promote and retain focus on end users by placing them along-side system developers, in the design team.

Iterative versus Waterfall

At face value, the waterfall lifecycle method of software development appears to provide all necessary parameters and safeguards for successful systems development. RUP has modified this approach, by adding iterations within each of its phases. However, with present-day socio-technical emphasis on product usability and end user satisfaction, even the RUP approach exhibits several menacing inadequacies. Among them:

- user input is limited to initial stage of development process
- design development is based on a static model
- testing of system occurs first after much of product is completed
- revisions can be both timely and costly

All of the methodologies included in this study employ iterations in their lifecycle models, however, DSDM is the epitome of iterative systems development. It offers iteration not only within the different stages of development, but between stages, as well.

Notable Feature(s)

Participatory Design

The influence and use of Participatory Design reaches far beyond the computer systems development arena to include such broad and diverse fields of product development as community housing, and children's tutoring aides.

Rational Unified Process

RUP is configurable and can be tailored to fit a wide variety of organizations. It has also adapted many of the best practices in modern object-oriented software development into its methodology.

Soft Systems Methodology

SSM tackles ill-structured and poorly-defined problem situations that contain large social components. It brings structure to these types of problems, which allows them to be managed in an organized manner.

Dynamic System Development Method

DSDM, by means of Rapid Application Development (RAD) aims to deliver a usable and useful 80% of the proposed system, in 20% of the time needed by standard waterfall lifecycle processes to produce an entire system.

The final system is more likely to meet end users true business requirements.

System implementation is likely to go more smoothly, since all relative stakeholders are involved throughout the entire development.

CHAPTER 6 DISCUSSION

This study has illustrated how varied the approaches can be, when addressing the subject of User Participation in systems design and development. There is no one single condition that dictates its integration into the process.

The Scandinavian philosophy of “democracy in the workplace”, which facilitated the emergence of Participatory Design, called for decisive action to bring about change. A strategy was needed to re-distribute decision-making authority. Suggestion boxes were not enough to affect the type of dramatic change that was sought by trade unions. Something more radical was needed. It was judged that by physically ”injecting” workers, the ultimate end users of the tools and systems being developed, into the very development process itself, the trade unions could create the necessary leverage; the leverage needed to ensure that their visions of powersharing would be realized. The intent was to influence the development of user-friendly/usable systems solutions, thus securing employment security.

The greater strength of Scandinavia’s Participatory Design lies, perhaps, in the ingenuity of the philosophy of user participation, itself, rather than in a specifically structured methodology. This makes very versatile and applicable in some form in countless situations; in systems development, and beyond.

While the study uncovered little, if anything, about attitudes regarding user participation in the Rational Unified Process (RUP), it has highlighted another aspect at least equally as important when developing new systems solutions: Iteration. It appears that RUP is for all intent and purposes a good example of a standard design-oriented development methodology. It employs all the correct checks, balances, and controls to ensure a thorough and precise execution of a development project; a systems developer’s project ‘bible’, so to speak.

RUP boasts that it utilizes many of the ”best practices” on the systems development methodologies market today. And one might venture to say that the inclusion of the ”practice” of ”Iterative Development" is what makes the whole process work. The years of experience with methodologies using the waterfall model method of systems development has exposed the need for increased flexibility throughout the process, and iteration is exactly what the doctor ordered, one might say. The age-old ”trial-and-error” approach proves yet again to be a true winner. RUP recognizes this need and smartly enough offers systems designers the methods, and the flexibility, to get him/her where s/he needs to go.

Although User Participation is not expressly mentioned in the RUP process, there is nothing in the process that hinders a designer from incorporating it into various processes. Perhaps Jacobson and company have deliberately left the door open. And, as RUP boasts about its frequent update releases and its ready availability via the Internet, it is probably only a matter of time before User Participation become another piece of artillery in RUP’s ”best practices” arsenal.

This study has also included a methodology of particular complexity, not necessarily in the techniques used in it, but in the types of problems the methodology seeks to address. Soft Systems Methodology (SSM) tackles the demanding issues of complex problem definition in a unique manner. Checkland’s development of, and research into, his theory of ”soft” and ”hard” problems has lead to the emergence of a relatively scientific approach to the

area of systems design, that supplies designers the blueprint by which to build their systems. Every project begins with the designer's struggle to identify the "problem". Checkland experienced this in his own work as an industrial systems designer, and envisioned applying similar techniques to tackle difficult problem situations in new areas.

SSM is one methodology that attempts to guide the designer through the arduous task of sifting through the maze of varying, and perhaps often conflicting, opinions s/he is presented with, when analyzing an organization's systems needs. It addresses the sheer nature of complex problem recognition and definition. And as the study has shown, there is a myriad of integrated methods and techniques used to bring clarity to these problem situations and to penetrate difficult-to-define areas of conflicting views, which many times may seem impermeable. The complexity of measuring and judging the various stake-holders' conceptions, interpretations, and attitudes, and discovering and defining the actual system requirements of an organization, is often the more daunting of the tasks systems designers face. This has been the challenge, which Checkland recognized, and thanks to his insightfulness and years of diligent research, the Soft Systems Methodology provides a structure for taking on these types of complex problem situations.

The last methodology this study has researched, Dynamic Systems Development Method (DSDM), has been shown to contain "the best of all worlds", with its use of iterative lifecycles and user participation. DSDM arrived on the scene as an answer to cries from systems designers and end users alike. The cry was for a usability standard, when using the Rapid Application Process (RAD). Here was a situation where the systems designers themselves are the "end users", and they used User Participation in their forming of the DSDM Consortium (interesting correlation). And out of this exercise in user participation came standardized user participation structuring for RAD users. One would suppose that it is no co-incident that the DSDM methodology is structured as it is.

DSDM was created to meet the need for standardization in RAD, and that it has accomplished, but it has also provided systems designers the perfect blueprint for integrating full User Participation into the development process.

This study has shown that the concept of user participation in systems development is both sound and rational, and that the structuring of methodologies to include participation by end users in the development process is not only plausible, but also functional and adaptable. And with present-day technological expansions, the issues and elements of product usability are all the more fundamental to successful system development.

Traditionally, there has been a strong tendency, on the part of systems developers to view their technological tasks as strictly "scientific" – the perspective of positivism. Their own conceptions of problem situations have been central; the assumption has been that a model of the "real world" view could be adequately fashioned by scientific methods. And unfortunately, this has resulted in many systems that offer solutions for problems that, in actuality, are more of interest to the developers themselves, than to those who require the solution; falling well short of actual end users' needs.

Positivism teaches that humans have two, and only two, sources of knowledge: observation, and the applications of science; so-called *hard knowledge*. Conversely, hermeneutics says that knowledge comes as the result of interpretation of human actions, thoughts, intuitions, experiences; *soft knowledge*. It only stands to reason that the combining of these

strident theories raises the bar on problem definition and solution implementation in the field of systems development.

In conclusion, we are, as it seems, challenged by nature (according to such psychological theories as Maslow's Theory of Motivation, and others), to seek out ways of defining and fulfilling those instinctive needs of human nature. Therefore, as reliance on computer systems becomes an increasingly integral part of our personal lives, and of our societies, the application of User Participation will continue to grow in importance.

User Participation seems to have been a natural extension of an evolutionary process set in motion decades ago. And as this study has highlighted, there continues to be avid support from a myriad of various disciplines within the socio-technical science community, interested in further exploration of the effects of user participation in systems design.

It is hoped that this study has contributed to generating heightened interest, and further dialog, concerning the dynamics of User Participation in system development.

CHAPTER 7 REFERENCES

LITERATURE

Avison, D. E., Fitzgerald, G.(1997).*INFORMATION SYSTEMS DEVELOPMENT: Methodologies, Techniques and Tools*.Berkshire:McGraw-Hill

Bengtsson, B-A.(1995),*Zigma Samhällskunskap*.Uppsala:Almquist&Wiksell

Jacobsen, D. I., Thorvik, J.(1997).*Hur moderna organisationer fungerar*.Lund:Studentlitteratur

Lewis,P.(1994).*Information-Systems Development*.London: Pitman Publishing

Stapleton, J.(1997).*DSDM:Dynamic Systems Development System*.Essex: Addison-Wesley

INTERNET DOCUMENTS

Ambler, Scott W., Constantine, Larry L.(2000, 2001).*Volumes 1-4 of Completing the Unified Process Series*.[www document].URL http://www.AmbySoft.com/

Brown, J.(1997).*HCI and Requirements Engineering – Exploring Human-Computer Interaction and software Engineering Methodologies for the Creation of Interactive Software*.[www document].URL http://www.acm.org/sigchi/bulletin/1997.1/brown.html

Couprie, D., Goodbrand, A., Bin, L., Zhu, D.(2001).*Soft Systems Methodology*.[www document].URL http://sern.ucalgary.ca/courses/seng/613/97/grp4/ssmfinal.html

DSDM Consortium.(2001).[www document].URL http.//www.dsdm.org

Ehn, P.(1992).*Scandinavian Design: On Participation and Skill*.[www document]. URL http://www.ilt.columbia.edu/Publications/papers/Ehn.html

Eurisco's Research Topics.(2001).[www document].URL http://www-eurisco.onecert.fr/topics.html

Ericsson, M.(2001).*Developing Large-Scale Systems with the Rational Unified Process*. [www document].URL http://www.rational.com/products/whitepapers/sis.jsp

Gaffney,G.(1999).*Information & Design*.[www document]. URL http://www.infodesign.com.au/usability/participatorydesign.html

Good, M.(1992).*Participatory Design of a Portable Torque Feedback Device*.[www document].URL http://www.recordare.com/good/chi92.html

Hewet, Baecker, Card, Carey, Gasen, Mantel, Perlman, Strong, Verplank.(1996).*Human-Computer Interaction*.[www document].URL http://www.acm.org/sigchi/cdg2.html

REFERENCES (Internet Documents, continued)

i3news:FW:ECSCW 2001 Worhshop on ParticpatoryDesign.(2001).[www document}. URL http://turing.mip.ou.dk/mail/i3news/msg00624.html

Int. Org. For Standardization.(2001).*Usability*.[www document]. URL http://www.iso.ch/iso/en/ISOOnline.openerpage

Jacoben, I.(1996).*Ivar Jacobsen's Address at the OOPSLA Conference 1996*.[www document].URL http://www.rational.com/products/whitepspers/352.jsp

Jarvis, C.(2001).*Peter Checkland's Soft Systems Methodology and CATWOE*.[www document].URL http://sol.brunel.ac.uk/~jarvis/bola/information/ssm.html

Kruchten, P.(2001).*What Is the Rational Unified Process?*.[www document]. URL http://www.therationaledge.com/context/jan_01/f_rup_pk.html

Lancaster University Management School.(2001).[www document]. URL http://www.lums.lancs.ac.uk/mansci/PhD/ResearchAreas.htm

Levinger, D.(1998).*Participatory Design History*.[www document]. URL http://222.cpsr.org/conferences/pdc98/history.html

Magnussin, C.(2001).*Introduktion till designmetodik, Rehabiliteringsteknik och design vt 2001*.[www document]. URL http://www.certec.lth.se/fk/vt_01/forelasning/design_metodik.html

Participatory Design and the Changing Structures of the Workforce in the Information Society.(1996).[www document].URL http://orgwis.gmd.de/~mambrey/pdcws.html

Participatory Design of Collaborative Systems - New Challanges?.(2001).[www document].URL http://ecscw2001.gmd.de/w1.html

Reich, Y.(1997).*Participatory Design (PD): Increasing value for all involved in, and affected by, design*.[www document].URL http://or.eng.tau.ac.il:7777/topics/pd.html

UCLA Cognitive Science.(2001).*What is Cognitive Science?*.[www document]. URL http://www.cogsci.ucla.edu/Background/

Underwood, J.(1996).*Soft Systems Methodology*.[www document].URL http://www-staff.mcs.uts.edu.au/~jim/bpt/ssm.html

Usability First.(2001).[www document].URL http://www.uasbilityfirst.com/intro/index.txl

APPENDIX

[1] The Rational Unified Process emphasizes the adoption of six *best practices* of modern software development, in order to minimize inherent risk in developing new software. These best practices are:

- Iterative Development
- Requirements Management
- Use of Component-based Architecture
- Model Visualization
- Continuous Quality Verification
- Change Management

These practices have been assimilated into the RUP definitions of:

Activities - the definitions of the way artifacts are produced and evaluated

Artifacts - the work products produced, used, or modified in the performances of activities

Disciplines - the specific areas of the software engineering process
i.e. Requirements, Analysis and Design, Implementation, and Test

Roles - sets of activities performed and artifacts owned