PADRE: A PARTICIPATORY DESIGN REQUIREMENT ENGINEERING SYSTEM

MIHALY LENART
Department of Biomedical Engineering
University of Miami
1251 Memorial Dr. MCA 219A
Miami, Fl. 33146

AND

ANA PASZTOR*
School of Computer Science
Florida International University
University Park, Miami, Fl. 33199
Fax: +1 (305) 34-3549
{lenartm, pasztora}@cs.fiu.edu

* This research has been supported by NSF, grant no.: IRI-9630699

Padre: A Participatory Design Requirement Engineering System

Abstract All important features, such as usability, quality, or efficiency of a product, are determined early on in the design process by the extent to which design requirements are captured and understood. Traditionally, however, those most effected by design decisions, such as users, have been left out of the design process in general and the requirement gathering in particular.

Although restricted exclusively to computer system design, recently a new approach, called Participatory Design, has grown out of the realization that traditional system design fails "to do things that both managers and users expect them to." With Participatory Design, the field of Requirement Engineering encompassing acquiring, refining, and checking clients' needs for computer systems has come into focus.

Recognizing that the problems addressed by Participatory Design are problems shared with design at large, we set ourselves the goal of applying existing Requirement Engineering methods to and develop new ones for design in general. In this paper, we present PADRE, a new Participatory Design Requirement Engineering intelligent agent-based system. Its scope is requirement *elicitation*, *specification* and *validation* using recognized, as well as new cognitive communication tools. PADRE takes as input an initial, informal requirement specification for a design object, as well as an ontology of the application domain, and generates a *complete*, *unambiguous*, *consistent*, and *implementable* requirement specification, as well as a tree representing the design process. It does so by "mentally" walking the designer and the user agents through the design process while stepwise refining specifications. In order to assist and partially automate the walk-through, we have developed a domain independent elicitation tool, called WALT. A session with WALT results in the generation of a tree structure representing various specification refinement levels and processes, as well as a tree representing the various design decisions made by the designer agent during the session.

PADRE is a participatory design system that can be used in any application domain. This sets it apart from most of the existing participatory design methods and tools.

1. Introduction

The pivotal role of design in product development has been widely recognized by industry and government agencies. Although the cost of design is relatively low compared to the cost of manufacturing and marketing, the impact of design on the production costs, product quality, and marketability is enormous. Therefore spending on design related research has been on the rise in the last decade. Another reason for the increased spending is that although design tasks are inherently difficult to solve, some initial success coming out of AI- and Cognitive Science-related research raised high hopes for new and efficient ways of dealing with complex design tasks.

One of the major problems in design is how to define the scope, goals and parameters of a design task. The *specification* of the design object, the purpose of which is to describe desirable features, requirements, and limitations of the product, as well as of the design and manufacturing process, constitutes the foundation of any design process. Traditionally, however, those most affected by design decisions, particularly the users, have been left out of the specification and the design process in general. “Imagined users, model users, or surrogate users like paid subjects of focus groups and operability tests stand in for those who will actually” use the product ([Schuler and Namioka, 1993], pg.vii).

The actual context in which the design product will be used is replaced by “stereotypic scenarios or extrapolations from prevailing models of generic” design solutions. The results are disastrous. Here are some:

- According to a recent newspaper report, the nation's largest architectural firm, the Minneapolis based Ellerbe Bekkett Co., has been sued under ADA by the Justice Department for failing to provide adequate access to six sport arenas built in metropolitan areas around the country in the last three years. In particular, the arenas seem to lack adequate sight for fans in wheelchairs. Although not disclosed, the financial consequences must be enormous.
- We had recently a lively discussion with the Executive Director of the National Catholic Office for Persons with Disabilities, Mary Jane Owen, about assistive devices for the physically challenged. She has a number of disabilities herself. Some time ago a wheelchair manufacturer gave her a prohibitively expensive model to try. She was very pleased at first with the wheelchair, however, when she wanted to try it at home and in her office, it turned out that the controller, which was placed on the side of the arm support, prohibited her from getting through doors. The manufacturer refused to change the design, because of the tremendous costs involved.

- According to the director of the Physical Therapy Department at the second authors' home university, 80% of all prosthetics wind up in the closet, because the users refuse to use them.
- The Rakow Youth Center on Miami Beach was an esthetic hit when it was built. However, it is all glass, full of outside balconies and galleries, and the atrium, where kids play outdoors, is all concrete. This keeps parents and personnel quite nervous, in spite of all safety measures.

Although restricted exclusively to computer system design, recently a new approach, called *Participatory Design* has been developed, "in which the people destined to use the system play a critical role in *designing* it." Participatory Design has its origins in the Scandinavian experience and advocates "customer centered approaches in an effort to chart a clear path from customer to deliverable" ([Comm. of the ACM], pg.31). It grew out of the realization that traditional system design fails "to do things that both managers and users expect them to" ([Greenbaum, 1993], pg. 30). "The most basic motivation [for Participatory Design] is the idea of democracy. To be more concrete: People who are affected by a decision or event should have an opportunity to influence it. Participation is the key element in democracy. The other compelling idea is that quality can improve with strong and effective participation of the people involved" ([Schuler and Namioka, 1993], pp. xi-xii). As a consequence of the Participatory Design approach, attention has been focused on a new field called Requirement Engineering.

According to [IEEE Std. 830-1984], *requirement engineering* is the process of acquiring, refining, and checking clients' needs for a software system. The RESG (Requirements Engineering Specialist Group of the British Computer Society) defines it as "the elicitation, definition, modeling, analysis, specification, and validation of needs of a computer system." Requirements Engineering is a multi-disciplinary field using methods and tools from disciplines such as Software Engineering, Artificial Intelligence, Cognitive Science, and Social Sciences. However, all research work done in and published on Requirements Engineering so far, is concerned with computer system design. In contrast, our goal is to apply existing requirements engineering methods to and develop new methods for *design in general*. We are motivated by the recognition that

a) all important features, such as usability, quality, or efficiency of a product, are determined early on in the design process by the extent to which design requirements are captured and understood, and
b) requirement engineering is of general concern not restricted to computer science or software engineering in particular. However, requirement engineering methods for software systems can benefit from other disciplines and vice versa.

In this paper we present a new participatory design requirement engineering system, PADRE, the scope of which is requirement elicitation, specification and validation. Requirement *elicitation* is the process of gathering information on the design requirements, domain, and environment. The elicitation process involves the application of various elicitation methods and tools, such as observation, interviewing, text analysis and questionnaires. Requirement *specification* describes *what* the design object is supposed to do and what features it is supposed to possess. *Validation* analyses the outcome of the requirement engineering process to establish the extent to which it accurately embodies the original requirements.

2. PADRE: Scope and Architecture

Most of the problems addressed in Requirement Engineering originate in problems inherent in the informal specification of design. Informal specifications are the well-known and commonly used specifications in a natural language, such as English, or rather a subset of a natural language used by a given profession. Informal specifications are also codes, guidelines, and regulations as they can be found in the professional literature. Each specification starts out informally. At each stage of the specification process, goals, parameters, and constraints can be clarified further indefinitely by elicitation of more and more details. For obvious reasons, however, the elicitation process has to get to a halt at some stage.

Each specification has two parts: a hidden, common knowledge part and an explicit part describing relevant features and/or requirements. To what extent the explicit part spells out details, depends on the purpose of the specification. While less details leave room for creativity, at some point in the design process the specification has to become *determinate* in order for the designer to be able to recover the design object from the verbal specification. In fact, given sufficient information, the specification process might lead to *implementable specifications* or even to concrete solutions in which case we talk about “executable specification” (c.f. [Davis, 1982], [Partridge and Wilks, 1987], [Rattray *et al*., 1991], [Lenart *et al*., 1994]). Usually, however, there is a huge *gap* between informal specification and design implementation. Our *goal* is to *minimize* this gap and support the automation of this process, and, move over to do so in the spirit of Participatory Design [Lenart and Pasztor, 1995, 1997].

To this end, we are developing a domain independent, generic (intelligent) agent-based requirement engineering system for design at large, called PADRE (PArticipatory Design Requirement Engineering). Capturing design requirements is often done by post hoc knowledge acquisition, which in general is inadequate and difficult [Kellogg *et al*., 1989]. In the approach described herein, however, design specification is a *dynamic* process evolving through a *direct interaction between all participants of the design process*. Similarly to [Lancaster *et al*., 1989], the elicitation process is based on the structure of human cognition through the use of recognized cognitive categories as defined in [Cameron-Bandler and Lebeau, 1985] (see also [Pasztor, 1997]), and uses several cognitive elicitation techniques. PADRE is a participatory design system that can be used in any application domain. This sets it apart from most of the existing participatory design systems, which, as we have mentioned before, are all aimed at system design [Comm. of the ACM, 1995].

PADRE employs several (not necessarily computer-based) intelligent agents. It takes as input an initial, informal requirement specification for a design object, as well as an ontology of this object's application domain (see next section for definition), and generates a *complete, unambiguous, consistent*, and *implementable* requirement specification as well as a tree repesenting the design process of the desired object. It does so by walking the designer and the user agents∗ trough the design process. During this *walk-through*, requirement specifications are stepwise modified by both the user and the designer agents, respectively, and requirements specifications are stepwise refined to implementable ones. Implementable specifications correspond to the *atomic objects* of [Alexander *et al.*, 1988] or *surrogates* in data base terminology [Codd, 1979]. In order to assist and partially automate the walk-through, we have developed a generic, domain independent elicitation tool, called WALT (WALk-Through). A session with WALT results in the generation of a tree structure representing various specification refinement levels and processes, as well as a tree representing the various design decisions made by the designer agent during the session.

PADRE is *extensible* in both the number and the type of intelligent agents employed and in terms of the tools it interfaces. The terminal nodes of the specification tree represent implementable requirements (corresponding to the measurands in [Harris and Sydenham, 1994]).

2.1 Solving Problems inherent in Informal Specifications

Design specification is necessary for the following reasons:

- Without specification there is no goal or direction for the design process. Even though specification is often abstract and intentionally vague, it makes goals and parameters explicit in order to avoid misinterpretations.
- Without specification chances are great to get lost in details. There are no guidelines, no feedback as to which tasks have been accomplished and which comes next. One has no overview of the entire design process.

∗ Here we use the term "user" to include all participants like customers, clients, financing agents, etc. who are involved in the definition of the design object. We use the term "designer" to include all participants who are involved in the design and production of the design object.

- Without specification verification of the correctness of emerging solutions is impossible.
- Without specification the impact of each task on the entire process cannot be determined, therefore, it is difficult to develop efficient design strategies. Bottleneck problems or deadlocks cannot be eliminated.
- Without specification the coordination of teamwork is almost impossible. Specification provides means to clearly define, delegate, and create interface between subtasks, as well as avoid redundancies.

There are two kinds of design specifications: formal and informal. *Formal specifications* use formal languages for describing design goals, parameters and constraints. They were first introduced in software design but got eventually adopted in other design areas, e.g. in VLSI-design, as well. Formal specifications are derived from informal ones and are usually not easily accessible for professionals involved in the design process. According to [Fraser, 1994]: "… only minimal guidelines are provided for eliciting and structuring the requirements before they are expressed in the formal notation [Jones, 1990]."

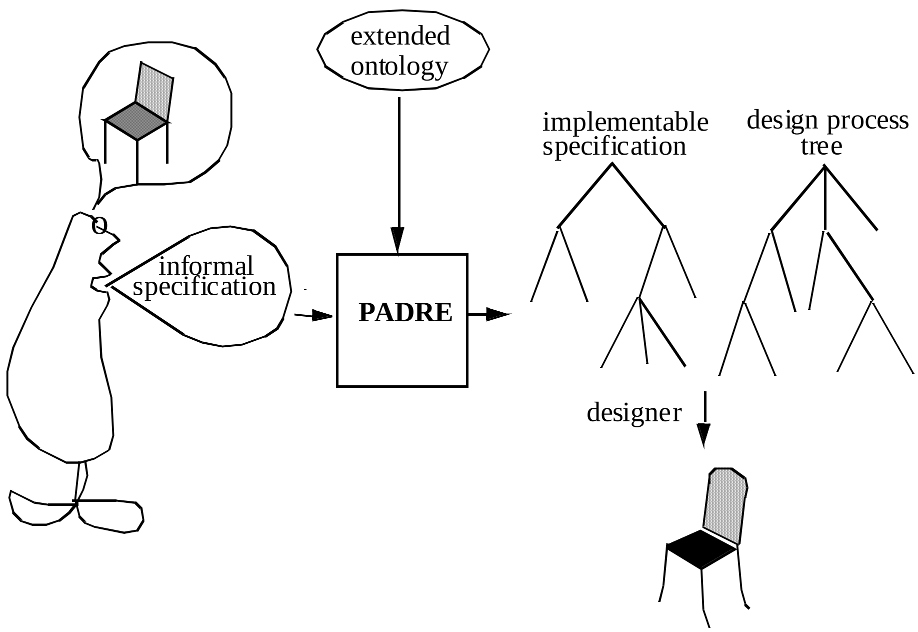

Figure 1: The concept of PADRE

An important part of the specification process is collecting and processing information. Information gathering or requirements elicitation is difficult because of the *gap between different mental images or models of the involved parties and of communication deficiencies*. This gap is often due to different modes of thinking associated in many cases with each kind of specification, namely formal mode and “narrative” mode, respectively [Kornell, 1988]. In narrative thinking, goals are less clearly defined than in formal thinking and models may be lifelike even if their elements are not likely. Even if the user has a clear mental image of the design object, there is still a gap due to the discrepancy between these mental images on one hand, and the vagueness of the verbal specification of the design object, on the other [Johnson-Laird, 1983] (pg.158). Informal specifications are inherently *incomplete, ambiguous* and *inconsistent*. The following is an individual description of each of these problems and how PADRE addresses them.

2.1.1 Incompleteness

When we “future pace,” i.e. we pack our luggage to travel and we want to make sure that nothing will be missing, then we mentally walk through a day or situations on the trip and see, hear or feel what we will wear, need or use. Sometimes we daydream about the trip and suddenly we remember to pack or buy something for the trip. Similarly, walking the user with the designer through the design process helps the user to *find requirements* which s/he did not think of initially. In PADRE, the designer walks the user through the design process soliciting missing requirements in this fashion. (Henceforth we will simply refer to the user agent as user and to the designer agent as designer.)

For example, let us assume that the user wants the designer to design a chair with the requirement that it be *easy to clean*. Walking through the design process with the user, the designer explores the possibility of an upholstered chair with nylon cover in order to satisfy the requirement *easy to clean*. Only at this point does the user remember to tell the designer that s/he is allergic to nylon.

The walk-through is carried out in a spiral fashion: First the designer mentally walks the user through the design process until they get to a point at which there are too many uncertainties to continue. Here the designer may offer the user solutions (e.g. early prototypes) satisfying existing requirements. Once the uncertainties are resolved, the walk-through resumes, etc.

In a case study, which we will refer to as “The Heart Project” and the aim of which was to develop a fluid mechanical model of the heart for testing artificial valves, a customer's requirement was to be able to change the stroke volume on the fly. The designer team satisfied this requirement by designing an adjustable pivot point on the crank arm connecting two parallel shafts — one attached to the motor and the other one to the piston. Walking with the user and the designers through the process, the authors, facilitating requirements elicitation functions, noticed that after changing the position of the pivot point, all information about the previous position was lost. At this point, the user realized that he does need this information for being able to keep track of and adjust the stroke volume. This new need (requirement) forced the designer to add a monitoring device for registering the position of the pivot point.

2.1.2 Ambiguity

Verbal specifications are often *vague*. The user might have a clear mental image of the design object, but be unable to *describe* it verbally with the same clarity. In addition, verbal descriptions often have *different meaning* for different people due to their individual experiences. This is particularly true if the specification is too coarse. During the walk-through, the *user* and the *designer* interact to *stepwise refine* these specifications to obtain unambiguous and therefore *implementable* requirements specifications. (A specification is called implementable if it completely and unambiguously describes the requirements for a design object such that it can be generated according to this description to the full satisfaction of the user.) The idea is to offer the users ways of refining their original large chunk, coarse requirements in terms of various elments/variables/ criteria involved in the design process.

PADRE helps to elicit more *determinate* representations of the design object with the goal of narrowing down the choice of images of this object in the designer's mind. Johnson-Laird [Johnson-Laird, 1983] (p.160) describes experiments proving that people tend to form a mental model from a determinate description (specification), but fail to do so from an indeterminate one. For indeterminate descriptions, people often come up with models consistent with the description, but which the specifier did not have in mind.

For example, going back to the chair design, let us assume that the specification calls for a *comfortable* chair. Clearly *comfortable* has different semantics for different people. The designer can not use this specification as it is now, unless s/he does some "mind reading," which actually is practiced a lot in design, as well as in ordinary, every day communication. Using an application domain ontology, the walk-through process prompts the user to refine his requirement *comfortable* in terms of variables like size, shape, material etc.

"A specification of a representation vocabulary for a shared domain of discourse — definition of classes, relations, and other objects — is called an *ontology* (italics ours)." [Gruber, 1993].

In the case of the Heart Project, almost all initial requirements, such as *durable, visible, easy to maintain, portable,* etc., had to be refined to unambiguous requirements, which were then almost trivial to validate. An example for problems due to ambiguity was the following: Certain mathematical formulas were applied by both the user and the designers for the same purpose in two different ways with different results due to different interpretations of the term: *static forces*.

At the end of the walk-through and as part of the *validation* process, the user establishes whether all aspects of, say, *easy maintenance* were exhausted in the process. If he thinks that this is not the case, then *missing aspects* will be *elicited* from the user and the domain design ontology will be extended. In the above example of the requirement *easy to maintain*, such a missing aspect could be a specification of the maximum number of parts needed to be removed for maintenance.

2.1.3 Inconsistency

Inconsistency may occur *within* the user's specification, but also *between* the user's specification and the *domain ontology/designer's knowledge base*. An example of the first kind is a specification containing the requirement of modern furniture and at the same time an antique desk. Another example of the first kind, which occurred in the Heart Project and is very common, is the inconsistency between the extent of the requirements and the time constraints set by the user. An example of the second kind is to require a wood floor in the kitchen while the code requires fire proof material.

Inconsistencies also occur between user requirements and designer's criteria. As a consequence of such inconsistencies, designers often ignore or override (not necessarily consciously) user requirements. For example, in the Heart Project, the designers chose to ignore the requirement of the *ability to change heart rate on the fly* because they didn't want to change the existing electronics. The walk-through process of PADRE is designed to discover and solve these inconsistencies (see next section).

2.2 The Concept and Architecture of PADRE

PADRE is a domain independent, generic agent based requirement elicitation system which assists in the creation of a complete, unambiguous and consistent specification of a desired design object. It uses intelligent agents to provide expert knowledge about

- the design process,
- the user requirements,
- the application domains, and
- cognitive elicitation techniques.

In the present phase of system development, the intelligent agents of PADRE are a user, a designer, a dialogue, and a requirement elicitation agent, respectively. The dialog agent is responsible for the transfer of information between the agents, as well as between the agents and the various knowledge bases, like e.g. domain ontology. The requirement elicitation agent is responsible for employing various cognitive elicitation techniques to elicit and help step-wise refine user requirements.

We do not intend to automate any of the user, designer or requirements elicitation agents completely, as that would violate the principles of Participatory Design. PADRE employs various software tools. Presently, the heart of PADRE consists of WALT — a generic, domain independent tool implementing the walk-through process.

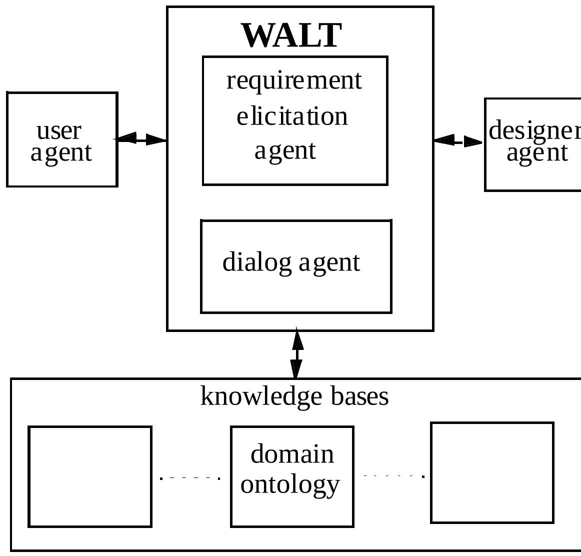

Figure 2: The PADRE architecture

2.2.1 WALT: A Domain Independent Walk-Through Tool

WALT provides a dialog structure for the walk-through. A session starts with the identification of the participants (in our case user and designer), and the design task, and with the selection of the appropriate domain ontology.

An initial informal user requirement specification, SPEC, initiates the construction of a specification tree, and becomes the primary specification. The dialog items are generic forms and browsers, as well as fields to be filled out during sessions. These provide the user interface between agents. In the following, we describe recursively a session with WALT (see **LOOP** below).

Next the designer enters the operation he thinks he has to perform to be able to design the desired object satisfying SPEC. This is stored in *OP*. As an example, let us suppose that the desired design object is the lay-out of a book. The corresponding domain ontology is used by the designer to choose the operations which he would need to perform to design a book lay-out satisfying the user's requirements specifications. Ontologies are represented in a frame-based language. Each class may correspond to a design operation and its subclasses to sub-operations. For example, the class "page" corresponds to the operation "design pages," while its subclasses "title page," "odd page," "even page," etc. correspond to the sub-operations "design title pages," "design odd pages," design even pages," etc. The designer is free to choose the operations and in which order he lists them, however, he is bound by meta-rules and constraints governing the mapping of the domain ontology to the design process. Moreover, the structure of the domain ontology encourages a top-down design style, where operations are decomposed into simpler sub-operations until "atomic" operations are reached.

(begin LOOP) In general, for each operation *OP* to be performed by the designer, after the designer has entered *OP* into the corresponding form, a sequence of forms follows to process all rules (constraints) and variables pertaining to the performance of *OP*.

Rules: First the designer lists all rules related to *OP* in his knowledge base or domain ontology. Next, the dialog agent prompts the user to check the *consistency* of the rules with SPEC. The user is prompted to change SPEC to remove all inconsistencies. When the user is done, the designer is prompted to backtrack as many steps in the operation tree as needed to accommodate the newly emerging specification, say SPEC'. This gives the dialog a *spiral* structure (see Figure 3). At all times the agents can access a database containing the operation tree (which has the design task as its root) developed so far by the designer. Access is also provided to the rules and variables of each operation.

Suppose the design task is to design a *zero scape landscaping of a yard* and the specification calls for *gardenias*. Since there is a designer rule "*Gardenia* $\rightarrow$ *Ten-gallons-of-water-per-day*," the user has to change the requirements.

In our *book lay-out* example, let *OP = choose characters for chapter titles*. The rules for *OP* contain "*chapter title* $\rightarrow$ *boldface characters*" and "*ZapfChancery font* $\rightarrow$ *no boldface characters.*" SPEC, however, contains the requirement that *chapter titles are written* in *ZapfChancery font.* The user now has to change this requirement because it is inconsistent with the designer's rules for *OP*.

Rules are represented in the designer's knowledge base, or the domain ontology. For example, the ontology may contain the following class definitions:

```
⟨chapter-title
    ⟨superclass title⟩
    ⟨characters (restrict (a characters with
                                ⟨weight boldface⟩))⟩⟩
⟨characters
    ⟨family (type string)
     (restrict (OR Areal, … ,Zurich))⟩
    ⟨size (type integer)
     (restrict (AND ≤1 ≥36))(unit pt)⟩
    ⟨weight (type string)
     (restrict (OR regular boldface))⟩
    …
    (if (characters:family ZapfChancery)
     then (weight (NOT boldface))⟩
```

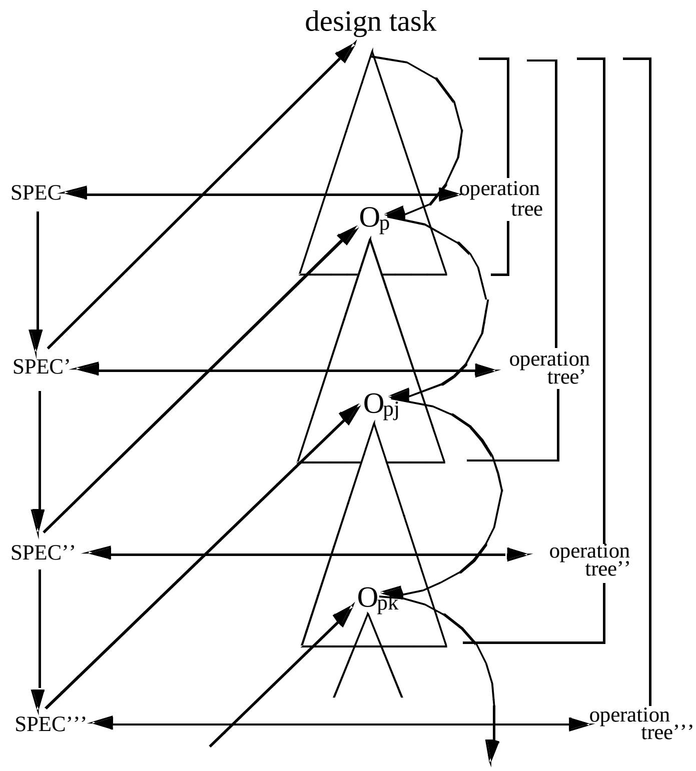

Figure 3: Spiral structure of a WALT session: the backtracking pro

Variables: Next, the designer lists all variables which need to be given values in order to perform *OP*.

As before, the designer is guided by the domain ontology. For example, in the case of *OP* = "*choose characters for chapter titles*," it follows strait from the *characters* class definition that the variables *family*, *size*, *weight*, etc. need instantiations. The designer may give these variables default values, which have to be consistent with their types and units, as well as the operation rules.

Next, the user is prompted to accept/reject the default values and instantiate the variable values. The user is provided with the allowed types and units of these values. In order to refine the non-implementable requirements, the user is also asked which of the variables help refine the requirements in SPEC. For example, SPEC may contain a requirement for *clearly legible print*. The user may find that the variables (*character*-) *family* and *size* of *OP* relate to this requirement, and he instantiates them accordingly. The user may, for example, find that Times Roman characters are more legible than Helvetica ones. So the requirement *clearly legible* is now refined in terms of a specific (character-) family and size (range). This is recorded in the specification database and becomes part of the final specification tree and will be used in the validation phase.

In spite of the fact that the user was provided the allowed type and units for the variable values, the values given by the user might still be ambiguous and/or inconsistent with the types, units, or operation rules. Therefore the designer checks for ambiguity and inconsistency and if any such is found, the user is prompted to change given values accordingly.

This is probably the most critical phase of the walk-through, as far as refinement of the requirements goes. Often the user is unable to refine his requirements appropriately to the requested values, and needs assistance. For example, to choose a character family, the user has to see print *samples* (*prototypes*). Or specifying the *hotness/mildness of a dish*, calibration may be in place.

Calibration means to match or adjust two value systems to a given scale. The concept of calibration is related to rating grids [Boose and Bradshow, 1988] and scaling/clustering [Cooke and McDonald, 1988] used for knowledge acquisition.

More generally, at this point appropriate values are elicited from the user by the requirements elicitation agent, using elicitation techniques like e.g. structured interviews ([Lerner, 1956], [Sjøberg and Nett, 1968], [Zeisel, 1981]), and [McGraw and Seale, 1988], protocol analysis [Ericcson and Simon, 1984], and hypermedia tools ([Wells, 1989], [Kopec et al., 1989)∗.

Suboperations: Now the designer lists all suboperations needed to be carried out in order to perform *OP*. **(end LOOP)** (see Figure 4.)

The **(begin LOOP)** through **(end LOOP)** is repeated for the next operation, until all operations are exhausted and all relevant variables have implementable values. As mentioned before, this process can be interrupted by phases of actual design, in which the present specification is being satisfied.

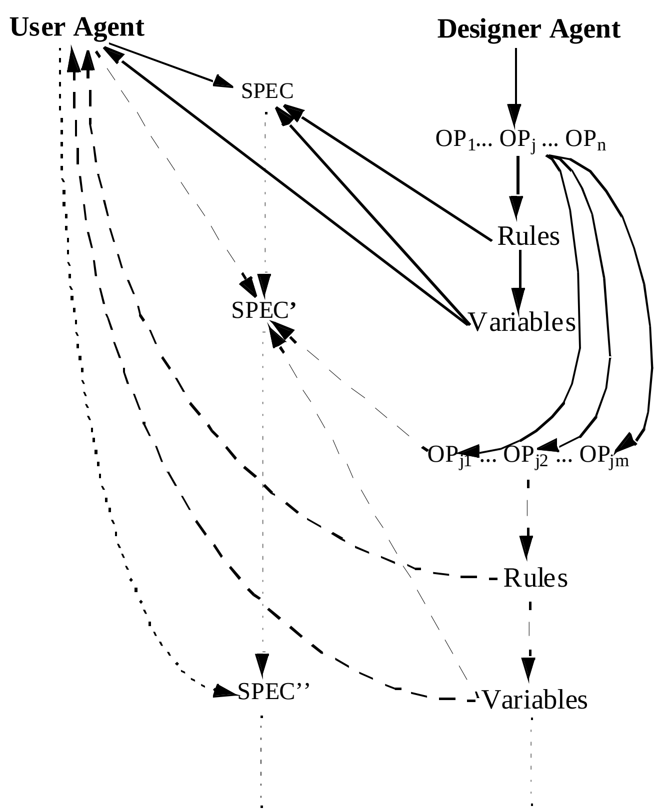

Figure 4: The structure of a WALT session

∗ For an excellent survey see [Byrd *et al*., 1993].

Validation: At the end of each WALT session, the user is presented with the generated specification tree. This tree represents the stepwise refinement of each non-implementable user requirement in terms of all variables instantiated during the session and their values. The user analyses the tree (possibly comparing it with an early prototype provided by the designer) and establishes the extent to which it accurately embodies the original needs. For example, let us assume that the requirement *clearly legible print* has been refined in terms of *character size*, *family*, *color*, as well as *margin sizes*, etc. The user may find that the paper color and the book binding method, which were not considered earlier, also affect the legibility of the print. The domain ontology is then extended accordingly, and a new WALT session may be started. The WALT sessions continue iteratively until the user is fully satisfied with the resulting specification tree (see Figure 5).

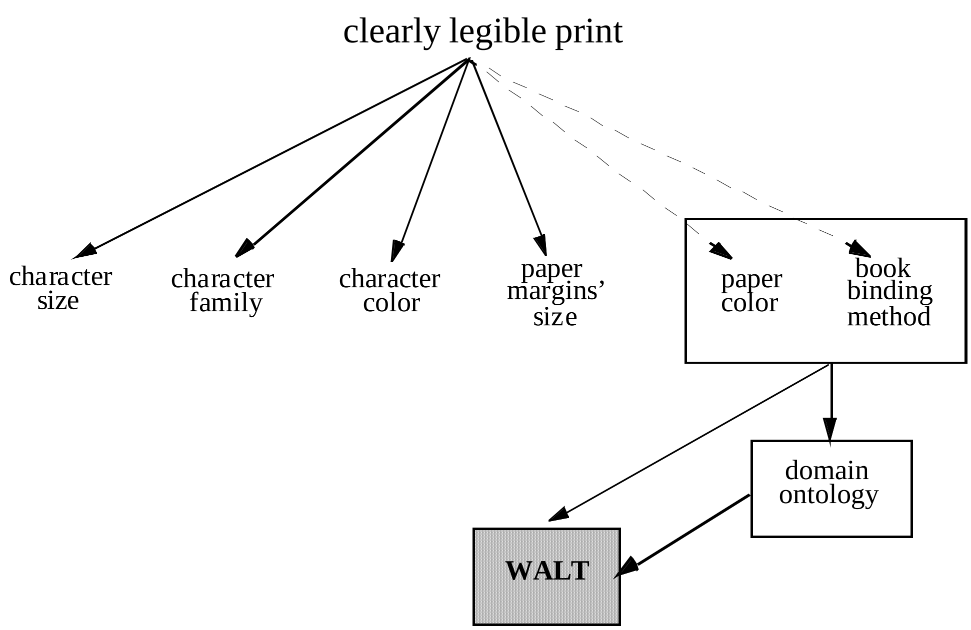

Figure 5: Validation

3. Relation of PADRE to Relevant Research and Future Work

The research described here closely relates to two major areas: Requirement Engineering, in particular Participatory Design for software systems, and Knowledge Acquisition (KA) for KB/Expert Systems.

1) Starting in the late 70-s and early 80-s, a number of Requirement Engineering tools and techniques have been developed. Some of them are: CORE (COntrolled Requirements Expression) [Mullery 1979] [Finkelstein *et al*.1992], ISAC (Information Systems Work and Analysis of Changes) [Lundeberg *et al*., 1981], SQD (Software Quality Development) [Sullivan 1986][Hauser and Clausing, 1988] [Akao, 1990], GIST [Davis, 1993], [Balzer et al., 1982.], JAD (Joint Application Development) [Adrews, 1991], PAISLey [Thayer and Thayer, 1990], REVS (Requirements Engineering Validation System) [Thayer and Thayer, 1990], RSL (Requirements Statement Language) [Boehm, 1984], SADT (Structured Analysis and Design Technique) [Thayer and Thayer, 1990], SREM (Software Requirement Engineering Methodology) [Alford, 1977, 1990][Bell et al., 1977], SSADM (Structured Systems And Design Methodology) [Asworth, 1989] [Ashworth and Goodland, 1990] [Malcolm, 1992] [Robinson, 1990].

Besides PADRE, the present authors are not aware of any other general, domain independent requirement engineering tool for Participatory Design. As mentioned earlier, Participatory Design for software systems has recently become the focus of attention due to an increasing gap between user requirements and software systems as these become more and more complex [Comm. of the ACM, 1995]. In particular, “The interest in participatory design, ethnographic techniques, and field research techniques grows out of the recognition that traditional interviews and surveying techniques are not adequate for today's applications.” [Beyer and Holzblatt, 1995]. The ideas behind the WALT as a tool to aid requirement elicitation, are closely related to the ideas behind Participatory Design-tools like Contextual Inquiry [Holtzblatt and Jones, 1993], Future Workshops, or Organizational Games (see [Boedker *et al.*, 1993]).

The concept of refining specifications interactively in a step-wise manner has been long pursued in software engineering, and automating this process is an important research topic in AI (c.f. [Harandi and Lubars, 1991]).

2) Knowledge acquisition is a fundamental part of numerous problem solving processes. With the advancement of KB/Expert Systems, it became one of the major research topics in AI. KA is not only fundamental, but also probably the most difficult and time consuming part of the problem solving process [Gruber, 1989]. It is also one of the major bottlenecks in the development of KB/Expert Systems. This bottleneck problem has been first tackled by domain specific tools such as SALT [Marcus, 1988], [Marcus and McDermott, 1989], KNACK [Klinker, 1988], MORE [Kahn *et al.*, 1985], ROGET [Bennett, 1985], MOLE [Eshelman, 1988], and SIZZLE [Offutt, 1988]. These are based on modeling of problem solving methods and application domains.

Other KA-tool designs took this approach a step further and tried to integrate a variety of problem solving methodologies or utilities into a single workbench ([Abrett and Burstein, 1987], [Boose and Bradshaw, 1988], [Diedrich *et al*., 1987]). However, these tools are limited to particular application domains and are difficult to modify for new applications.

In order to overcome such limitations, new methods and a new generation of *generic* meta-level tools, such as DIDS [Runkel and Birmingham, 1993], Spark [Eriksson and Musen, 1993], KADS [Wielinga *et al*., 1992], [Puerta, 1992], DASH [Eriksson *et al*., 1994], GAS [Chien *et al*., 1995], SBF [Yost *et al*., 1994] and PRO-MINDS [Harris and Sydenham, 1994] have been developed. These tools can either help to generate new domain-specific KA-tools or acquire knowledge for a larger domain by applying multiple and alternative problem solving methods.

Many features of these KA-tools are incorporated into the PADRE concept. In particular, ontologies are used as a basis for the requirement engineering process similar to the approach proposed in DASH. An important model for the development of PADRE was the PRO-MINDS project the aim of which is to support the design process by a multi-sensor system that can elicit knowledge about observed objects or events.

Besides the above mentioned similarities to other KA-tools and methods, PADRE has the following unique features:
a) PADRE provides a generic requirement engineering method and appropriate tools. Its application is not restricted to a particular design domain.
b) Sessions with WALT *change dynamically* the domain ontology as well as the KB of the agents involved.
c) WALT is the only requirement elicitation tool which systematically refines all elicited requirements to implementable ones. It is able to do this because it employs effective elicitation methods based on the structure of human cognition through the use of recognized cognitive categories.

Although currently PADRE presupposes the use of single agents communicating sequentially, future implementations will accommodate the concept of concurrent design with geographically distributed teams. Future work on PADRE will also address issues, such as:

- The problem of identifying all stakeholders which should have a voice in the requirement elicitation process.
- The problem of “contamination” or overlapping of participants’ roles, i.e. participants themselves are often not aware which role they play in the requirement elicitation process. Identifying the exact role of each participants in a given situation, eliminates numerous ambiguities and inconsistencies.

- Different participants have different views and interests in the elicitation process. Representing multiple views is one of the central issues in Requirement Engineering [Nuseibeh *et al*., 1993] [Kotonya and Sommerville, 1992].
- The problem of knowledge representation related to the design ontology and the designer's strategy.
- The problems of identifying and hiding design decisions irrelevant to the user, as well as that of requiring domain expertise from the user, like in the case when the user has to refine a requirement in terms of designer-defined variables.
- Problems related to agent communication.

References

[Abrett and Burstein, 1987] G. Abrett and M. Burstein, The KREME knowledge editing environment, *International Journal of Man-Machine Studies*, 27, (2):103-126.

[Akao, 1990] Y. Akao, Quality Function Deployment: Integrating Customer Requirements into Product Design, Productivity Press: Cambridge.

[Alexander *et al*., 1988] J.H. Alexander, M.J. Freiling, S.J. Shulman, S. Rehfuss, and S.L. Messick. Knowledge Acquisition Tools for Expert Systems, volume 2 of *Knowledge Based-Systems*, chapter Ontological Analysis: An Ongoing Experiment, pages 25-37. Academic Press.

[Alford, 1977] M. Alford, A Requirements Engineering Methodology for Real-Time Processing Requirements, *IEEE Transactions on Software Engineering*, vol. SE-3, no. 1, 60-69.

[Alford, 1990] M. Alford, Software Requirements Engineering Methodology (SREM) At the Age of Eleven – Requirements Driven Design, In: Ng, P.A.; Yeh, R.T. (eds.): *Modern Software Engineering: Foundations and Current Perspectives*, Van Nostrand Reinhold: New York. 351-377.

[Adrews, 1991] D.C. Adrews, "JAD: A Crucial Dimension For Rapid Application Development," Journal of Systems Management, vol. March, pp. 23-31.

[Ashworth, 1989] Ashworth, C.M. "Using SSADM to Specify Requirements". In: Professional Software Engineering Group C1 (ed.): *Colloquium on Requirements Capture and Specification for Critical Systems*, The Institution of Electrical Engineers: London.

[Ashworth and Goodland, 1990] C. Ashworth and M. Goodland, SSADM: A Practical Approach, McGraw-Hill: London.

[Balzer *et al*., 1982] R. M. Balzer, N. M. Goldman, and D. S. Wile, "Operational Specification as the Basis for Rapid Prototyping, " ACM Soft. Eng. Notes., vol. 7 No 5, pp. 3-16.

[Bell et al. 1977] T.E. Bell, D.C. Bixler, M.E. Dyer, An Extendable Approach to Computer-Aided Software Requirements Engineering, IEEE Transactions on Software Engineering, vol. SE-3, no. 1, 49-60.

[Bennett, 1985] J.S. Bennett, ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system, *Journal of Automated Reasoning*, 1:49-74.

[Beyer and Holzblatt, 1995]Hugh R. Beyer and Karen Holtzblatt, Apprenticing with the Customer, *Communication of the AC*, 38(5): 45-52, May, 1995.

[Boedker *et al*., 1993] S. Boedker, K. Groenbaek, and M. Kyng. *Participatory Design*, eds. D. Schuler and A. Namioka, chapter Cooperative Design: Techniques and Experiences from the Scandinavian Scene. Lawrence Erlbaum Assoc.

[Boehm, 1984] B. W. Boehm, Verifying and Validating Software Requirements and Design Specifications, *IEEE Software*, vol. 1 NO 1, pp. 75-88.

[Boose and Bradshaw, 1988] J.H. Boose and J.M. Bradshaw. *Knowledge Acquisition Tools for Expert Systems*, volume 2 of Knowledge-Based Systems, chapter Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems, pages 39-64. Academic Press.

[Byrd *et al*., 1993] Terry Anthony Byrd, Kathy L. Cossick, and Robert W. Zmund. A synthesis of research on requirrement analysis and knowledge acquisition techniques. *Management Information Systems Quarterly*, pages 117-138, March 1993.

[Cameron-Bandler and Lebeau, 1985] L. Cameron-Bandler and D. Gordon Lebeau, *The Emprint Method. A Guide to Reproducing Competence*, Real People Press, Moab, Utah.

[Chien *et al*., 1995] ChiH-Cheng Chien, Cheng-Seen Ho, and Hahn-Ming Lee. mwkat: A meta-tool-generated knowledge acquisition tool for multimedia workstations development. *Applied Intelligence*, 5:51-72.

[Codd, 1979] E.F. Codd. Extending the database relational model to capture more meaning, *ACM TODS*, 4(4):397-434, December 1979.

[Comm. of the ACM, 1995] *Comm. of the ACM*, May. Requirement gathering: The human factor, May 1995.

[Cooke and McDonald, 1988] N.M. Cooke and J.E. McDonald. *Knowledge Acquisition Tools for Expert Systems*, volume 2 of Knowledge Based-Systems, chapter The Application of Psychological Scaling Techniques to Knowledge Elicitation for Knowledge-Based Systems, pages 65-82. Academic Press.

[Davis, 1993] A. M. Davis, Software Requirements Analysis & Specification, Second ed. USA: Prentice-Hall.

[Davis, 1982] R. Davis. *Knowledge-Based Systems in Artificial Intelligence*, chapter Teiresias: Applications of meta-level knowledge, pages 227-490. McGraw-Hill, New York.

[Diedrich *et al*., 1987] J. Diedrich and I. Ruhmann and M. May, KRITON: A knowledge acquisition tool for expert systems, *International Journal of Man-Machine Studies*,26,(1): 29-40.

[Ericcson and Simon, 1984] K.A. Ericcson and H.A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, Cambridge.

[Eriksson and Musen, 1993] H. Eriksson and M. Musen. Metatools for knowledge aquisition. *IEEE Software*, pages 23-29, May 1993.

[Eriksson et al., 1994] H. Eriksson, A.R. Puerta, and M.A. Musen. Generation of knowledge-acquisition tools from domain ontologies. *International Journal Human Computer Studies*, 41:425-453.

[Eshelman, 1988], L. Eshelman,ed. Marcus, *Automating Knowledge Aquisition for Expert Systems*, chapter MOLE: A knowledge acquisition tool for cover-and-differentiate system, pages 37-80, Kluwer Academic Publishers, MA.

[Finkelstein et al., 1992] Finkelstein, A.; Kramer, J.; Nuseibeh, B.; Finkelstein, L.; Goedicke, M. Viewpoints: A Framework for Integration Multiple Perspectives in System Development". International Journal of Software Engineering and Knowledge Engineering, vol. 1, no. 2, 31-58.

[Fraser, 1994] Fraser, Martin D., Kumar, Kuldeep, Vaishnavi, Vijay K.: Startegies for Incorporating Formal Specifications, Communications of the ACM, Oct., pp. 74-84.

[Greenbaum, 1993] J. Greenbaum, A design of one's own: towards participatory design in the United States, in D. Schuler and A. Namioka, eds., *Participatory Design*, eds., Lawrence Erlbaum Assoc.

[Gruber, 1989] T.R. Gruber. *The Acquisition of Strategic Knowledge*. Academic Press, San Diego, CA.

[Gruber, 1993] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199-220.

[Harandi and Lubars, 1991] M.T. Harandi and M.D. Lubars. Artificial Intelligence & Software Engineering, chapter Automating Software Specification and Design, pages 43-59. Ablex Publishing, Norwood, New Jersey.

[Harris and Sydenham, 1994] D.D. Harris and P.H. Sydenham. Pro-minds: Development of a software tool to support the measurement system designer. *Measurements*, (15):I-14.

[Hauser and Clausing, 1988] , J.R. Hauser and D. Clausing, The House of Quality, *Harvard Business Review*, vol. 66, no. 3, 63-73.

[Holtzblatt and Jones, 1993] K. Holtzblatt and S. Jones. *Participatory Design*, eds. D. Schuler and A. Namioka, chapter Contextual Inquiry: A Participatory Technique for System. Lawrence Erlbaum Assoc.

[IEEE Std.830-1984] IEEE Guide to Software Requirements Specification, The Institute of Electrical and Electronics Engineers: New York.

[Jackson, 1983] M.A. Jackson, Systems Development. *Prentice Hall*: Englewood Cliffs, NJ.

[Johnson-Laird, 1983] P.N. Johnson-Laird. Mental Models. Harvard University Press, 1983.

[Jones, 1990] C.B.Jones, Systematic Software Development Using VDM, 2nd ed., Prentice-Hall, Englewood Cliff, NJ.

[Kahn 1985] G. Kahn and S. Nowlan and J. McDermott, Strategies for knowledge acquisition, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7 (5):511-522, September, 19985.

[Kellogg et al., 1989] C. Kellogg, R.A. Gargan Jr., W. Mark, J.G. McGuire, M. Pontecorvo, J.L. Schlossberg, J.W. Sullivan, and N. Singh M.R. Genesareth. The acquisition, verification, and explanation of design knowledge. *SIGART Newsletter, Knowledge Acquisition Special Issue*, (108):163-165, April 1989.

[Klinker, 1988] G. Klinker, ed. S. Marcus, *Automating Knowledge Acquisition for Expert Systems*. Chapter KNACK: Sample-driven knowledge acquisition for reporting systems, 125--174, Kluwer Academic Publishers, MA.

[Kopec *et al*., 1989] D. Kopec and L. Latour and M. Brody, Towards An Expert/Novice Learning System with Application to Infectious Disease, *SIGART Newsletter, Knowledge Acquisition Special Issue*, 108, 140-143, April, 1989.

[Kotonya and Sommerville, 1992] Kotonya, G. and Sommerville, I., Viewpoints for requirements definition, Software Engineering Journal, November, 375-387.

[Lancaster *et al*., 1989] Juliana S. Lancaster, Christopher R. Westpha, and Karen L. McGraw. A cognitively valid knowledge acquisition tool. *SIGART Newsletter, Knowledge Acquisition Special Issue*, (108):152-154., April 1989.

[Lenart *et al*., 1994] M. Lenart, P. Padawitz, and A. Pasztor. *Formal Design Methods for CAD*, chapter Formal Specification for Design Automation, pages 201-220. *IFIP Transactions B18*. Elsevier, North Holland.

[Lenart and Pasztor, 1995] M. Lenart and A. Pasztor, The Chunking Engine: Closing the Gap Between Formal and Informal Design Specifications, *Proceedings of the Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, June 6-9, Melbourne, Australia, Gordon and Breach Science Publisher, pp. 491-499.

[Lenart and Pasztor, 1997] M. Lenart and A. Pasztor, Dynamic Design Specification with Chunking, poster, *Fifteenth International Joint Conference on Artificial Intelligence,* Nagoya, Aichi, Japan, August 23-29.

[Lerner, 1956] D. Lerner. Interviewing Frenchmen. *American Journal of Sociology*, (61):187-194.

[Lundeberg *et al*., 1981] M. Lundeberg, G. Goldkuhl, and A. Nilsson, Information Systems Development: A Systematic Approach, Prentice Hall: Englewood Cliffs.

[Malcolm , 1992] E. Malcolm, SSADM Version 4: A User's Guide, *McGraw-Hill*: London.

[Marcus, 1988] S. Markus. *Knowledge Acquisition Tools for Expert Systems*, volume 2 of *Knowledge-Based Systems*, chapter Taking Backtracking with a Grain of SALT, pp. 211-226, Academic Press.

[Marcus and McDermott, 1989] S. Marcus and J. McDermott, SALT: A knowledge acquisition language for propose and revise system, *Artificial Intelligence*,39(1):1-37, May, 1989.

[McGraw and Seale, 1988] K. McGraw and M.R. Seale. Knowledge elicitation with multiple experts: Considerations and techniques. *Artificial Intelligence Review*, 2:31-44.

[Mullery, 1979] G.P. Mullery, CORE – A Method for Controlled Requirements Specification, Fourth International Conference on Software Engineering, The Institution of Electrical Engineers: London. 126-135.

[Nuseibeh et al., 1993] B. Nuseibeh, J. Kramer, A. Finkelstein, Expressing the relationships between multiple views in requirements specification, Proc. Of the 15th Int. Conf. On Software Engineering, Baltimore, USA, May, IEE CS Press.

[Offutt, 1988] Offutt, D., ed. S. Marcus", *Automating Knowledge Acquisition for Expert Systems*, chapter SIZZLE: A knowledge-acquisition tool specialized for the sizing task, 37-80, Kluwer Academic Publishers, MA.

[Partridge and Wilks, 1987] D. Partridge and Y. Wilks. Does ai have a methodology that is different from software engineering? *AI Review*, 1:111-120.

[Pasztor, 1997] A. Pasztor. Intelligent Agents with Subjective Experience. Extended Abstract, Proceedings of the *Nineteenth Annual Conference of the Cognitive Science Society*, Stanford University , August, 1997 .

[Puerta *et al*., 1992] A.R. Puerta, J.W. Edgar, and M.A. Musen. A multiple method knowledge acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition*, 4:171-196.

[Rattray *et al*., 1991] C. Rattray, J. McInnes, and A. Reeves. *Artificial Intelligence & Software Engineering*, chapter A knowledge-based model for software re-usability, pages 61-82. Ablex Publishing, Norwood, New Jersey.

[Robinson, 1990] K. Robinson, Aut2 User Manual: A Workbench Designed by Analysts for Analysts, *Model*, Systems Ltd.: London.

[Runkel and Birmingham, 1993] J.T. Runkel and W.P. Birmingham. Knowledge acquisition in the small: building knowledge-acquisition tools from pieces. *Knowledge Acquisition*, 5:221-243.

[Schuler and Namioka, 1993] D. Schuler and A. Namioka, eds., *Participatory Design*, eds., Lawrence Erlbaum Assoc.

[Sjøberg and Nett, 19681 G. Sjøberg and R. Nett. *A Methodology for Social Research*. Harper and Row, New York.

[Thayer and Thayer, 1990] R. Thayer and M. Thayer, "Glossary," in System and Software Requirement Engineering, R. Thayer and M. Dorfman, Eds. Los Alamitos, CA: IEEE Computer Society Press.

[Wells, 1989] T.L. Wells, Hypertext as a Means for Knowledge Acquisition, *SIGART Newsletter, Knowledge Acquisition Special Issue*, 108, 136--138, April, 1989.

[Wielinga *et al*., 1992] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. Kads: a modeling approach to knowledge engineering. *Knowledge Acquisition*, 4(l):5-53, March 1992.

[Yost *et al*., 1994] G.R Yost, G. Klinker, M. Linster, D. Marques, and J. McDermott. The sbf framework, 1989-1994:from applications to workplaces. In G. Schreiber and W. Van de Velde, editors, *A Future for Knowledge Acquisition*, 8th European Knowledge Acquisition Workshop, EKAW '94, pages 318-339, Hoegaarden, Belgium, September 1994. Springer. Lecture Notes in AI 867.

[Zeisel, 1981] J. Zeisel, *Inquiry by Design: Tools for Environment-Behavior Research*. Brooks Cole, Monterey, CA.