A Design for Evidence-based Software Architecture Research

WenQian Liu†, Charles L. Chen, Vidya Lakshminarayanan, Dewayne E. Perry

†Software Engineering
Department of Computer Science
University of Toronto, Canada
wl@cs.toronto.edu

Empirical Software Engineering Lab
Electrical and Computer Engineering
The University of Texas at Austin
{clchen,vidya,perry}@ece.utexas.edu

ABSTRACT

Active research is being done in how to go from requirements to architecture. However, no studies have been attempted in this area despite a long history of empirical research in software engineering (SE). Our goal is to establish a framework for the transformation from requirements to architecture on the basis of a series of empirical studies. The first step is to collect evidence about practice in industry before designing relevant techniques, methods and tools. As part of this step, we use an interview-based multiple-case study with a carefully designed process of conducting the interviews and of preparing the data collected for analysis while preserving its integrity. In this paper, we describe the design of this multiple-case study, delineate the evidence trail, discuss validity issues, outline the data analysis focus, discuss meta issues on evidence-based SE particularly on combining and using evidence, describe triangulation approaches, and present two methods for accumulating evidence.

1. INTRODUCTION

Requirements to architecture is the first and hardest step in the process of engineering software systems. The research fields in requirements and architecture have developed independently for more than a decade [21, 8]. Recently, active research is being pursued in understanding the relationships between them and developing methods for the transformation from requirements to architecture [1, 2]. However, there has been no attempt to study the transformation empirically and no evidence has been collected on how that has been done in practice despite the long history of empirical research in software engineering [27].

Our goal is to understand how to transform requirements into architectures, and based on that, to provide practical and effective techniques, methods, processes and tools. We take an empirically-based rather than a technology-based approach (see the discussion in section 6.2) in our research. As the first step of a series of studies, we use an interview-based multiple-case study method [31] to find out what architects do in practice.

Our study involves a series of semi-structured interviews on requirements and architecture topics with multiple subjects. The interviews are semi-structured because we use a predesigned questionnaire with open-ended questions and employ a conversation based style rather than a question and answer form. We call it a multiple-case study because it is collected from nine carefully selected practicing software architects from industry.

There are a number of specific topics that we are interested in this study: (i) how software architects manage requirements and make design decisions particularly concerning non-functional requirements; (ii) what their view of software architecture is in terms of the critical characteristics of software architecture, the driving forces of its creation, and the very meaning of it; (iii) what is their personal opinion on the critical aspects and characteristics of a great or superb software architect; and (iv) how they view the evolution of the software from an architect's point of view.

We have carefully designed the process of conducting the interviews and preparing the data collected for analysis while preserving its integrity. We wish to establish a framework for the transformation from requirements to architecture based on our analysis of the data and proceed to method design. In this paper, we describe the design of the case study, delineate our evidence trail, discuss validity issues, outline our data analysis focus, discuss meta issues on evidence-based software engineering particularly on combining and using evidence, describe our approach towards data triangulation, present two methods for accumulating evidence, and outline our future work.

2. RELATED WORK

In recent years, there has been an active focus on bridging the gap between requirement engineering and architectural design [1, 2]. In particular, a number of approaches stemming from the goal-oriented requirement specification techniques [37, 5, 35] have emerged. Mylopoulos and van

Lamsweerde both extended their goal-oriented frameworks independently toward agent-oriented design paradigms [38, 20, 34]. Perry et al., on the other hand, worked toward architecture prescriptions [4]. Jani et al. have compared and evaluated these two approaches in a case study [16].

Dromey, in his GSE (Genetic Software Engineering) work [6], provided a systematic approach in going from a set of functional requirements to a system-level design by using behavior trees for representation and integration. It focuses on the functional (or behavioral) requirements and emergent properties in design, but does not address quality attributes or tracing rationales.

Leveson [17] introduced the concept of intent specification based on research from the cognitive science community [36] in support of tracing rationales behind design decisions. The focus of that work is to provide convenient notations and processes that match human cognitive capability at every step of system design. It uses intent to link the adjacent development phases such as the *requirement phase – System Purpose*, and the *design phase – System Design Principles*.

Problem frames [15] capture generic patterns in problems and provides a means to break down a large problem into recognizable subproblems and reuse their descriptions. Recently, Nuseibeh et al. have extended this work towards architectural design [13, 29]. These extensions are mainly geared toward a pattern-oriented heuristic. Other heuristics are also introduced such as a rule-based framework [18] and a classification based method [12].

As far as the (interview-based) empirical research goes, to the best of our knowledge, no one has used it to establish a connection between requirements and architecture in a broad sense. The closest work to ours is the interview-based research done by Graaf et al. on the state of practice of requirements engineering and architectural design processes in the embedded software domain [10]. However, a number of independent empirical studies have been done in requirements and architecture separately. An interview-based research of real world influence on software architecture by Mustapic et al. investigated the significant influencing factors over systems and software architectures [19]. Smolander et al. also conducted a case study on three software organizations identifying what is included in a software architecture from the architectural analysis, description and tools perspectives [32]. Hickey and Davis summarized their empirical studies of the selection of requirements elicitation techniques through interviews with nine expert analysts [14].

Our work will be among the first in collecting evidence from the practice in bridging the gap between requirements and architecture, and establishing a frame of reference empirically for further research.

3. THE CASE STUDY DESIGN

In this section, we discuss the case study from its preparation, the evidence chain, to the evidence trail. The preparation is about the design of the questionnaire where we consider both the depth and breadth coverage in the area of our interests. The evidence chain is about finding subjects, conducting the interviews, and collecting evidence. The evidence trail is about preparing the data for analysis and summary while maintaining its integrity.

3.1 Preparation

The major step in our preparation for this research is the design of the questionnaire. The questions are structured by topic and are arranged in sections. All the questions are open-ended and designed to guide conversations rather than to get short answers. The sections are:

A – Problem Domain
B – Overview of Software Architecture
C – Requirements and Architecture
D – Relating to Evolution
E – Professional Background
F – Comments

The questionnaire was initially prepared by one author, then reviewed by all authors and revised. Reviews were also carried out after each interview session and revisions were applied whenever necessary to bring clarity to, or fill in the gaps between, the questions.

Here are some examples from the questionnaire where question *B1* belongs to section *B*, and *C3* and *C4* belong to section *C*.

- (B1) What do you consider to be the critical characteristics of software architecture?
  - What is the essence of software architecture?
  - How many levels of architecture are there?
  - How detailed should it be?
  - Should it be prescriptive or descriptive?
  - What would the architectural representation include?
  - How do you communicate an architecture to the stakeholders?
- (C3) How do you transform the functional requirements into an architecture?
  - Do non-functional requirements play a role in this transformation?
  - Any examples?
- (C4) How do you handle the non-functional requirements?
  - Do you integrate these with the functional ones initially or after you have considered the functional ones and built a skeleton architecture?
  - How do the functional and non-functional aspects interplay in the design of an architecture?
  - Is there an ordering or a set of priorities for non-functional requirements?
  - Are there some non-functional requirements more implementation issues than architectural?

3.2 Evidence Chain

The initiation and conduct of interviews, and collection of data make up our evidence chain. Our subjects have been identified either through direct contact with the authors or references from other contacts in the cities of Toronto and Austin. Our goals include interviewing architects from diverse domains, different organizations, and specialized areas. We plan to include more cases as needed for one of two reasons: to increase the domain and industry coverage; and to deepen our knowledge of specialized areas in practice.

We use the questionnaire as the basis of the interviews. During the interview, we employ a conversation based style which means we start from a question in the questionnaire and carry on with the flow of the current topic and content, rather than interrupt it with the next question. We have the subject describe the details with the help of examples in answering each question. Each interview session is between one and three hours. Each subject may have one or more interview sessions. We also follow up should further clarification be needed.

All conversations are recorded with the permission from the interviewees. We have taken a strict security measure by only allowing members of the project to access the data. In the future, we will make our questionnaire available on the web, and with permissions from our interviewees, we plan to extract away any identity related information and make the remaining data available on the web so that other researchers and interest groups can have access to it.

3.3 Evidence Trail

The focus of the evidence trail is to prepare the data for analysis and summary. There are four steps: (i) transcribe the interview; (ii) annotate the interview according to a set of rules; (iii) partition the interview into specific questions using a processing program; and (iv) select quotations manually. Note that the versions resulting from each step are kept and that nothing in the evidence trail is discarded. Thus, we can move up and down in the evidence trail as needed. We discuss each in depth with some examples.

Transcription The interviews are manually transcribed by two of the authors. For each minute of recorded interview audio, we estimate that three minutes were spent in transcribing and proof reading[1]. The interview audio runs for a total of 1050 minutes ($\sim$ 18 hrs) and the total transcription time is estimated to be around 3150 minutes ($\sim$ 53 hrs).

Annotation The transcriptions are annotated to relate the data to each question. The annotations used are self-explanatory and shown below. For ease of processing, we divided each interview session into several parts.

```
|I|<interviewee name>|      |/I|<interviewee name>|
|P|<interview part number>| |/P|<interview part number>|
|R|<interviewer name>|      |/R|<interviewer name>|
|Q|<question number>|       |/Q|<question number>|
|D|<Date:mm-dd-yyyy>|       |/D|<Date:mm-dd-yyyy>|
|T|<Time:mm-ss>|
```

[1]Each of the two authors transcribes a different part of the interview. The transcriptions are then exchanged between the two authors and thoroughly reviewed.

Here is a fragment of an actual interview with annotations.

```
|I|name U| |P|2| |D|8-3 & 5-2004| |Q|C3| ...
|T|4:16| |Q|D5| I think the things that emerge first are
the things you are familiar with. You're probably going to
spend more time on the things that require new... I think
it's like a cluster. Things will sort of immediately seem
like there's bright spots that you sort of know how to
deal with based on things you've built before, and then
there'll be murkier areas that will require more effort
and thought and probably iteration to develop. ...

|T|5:02| Yeah, I think you probably would be... I mean
this is what I think is behind the patterns work, sort of
recognizing and hopefully actually reusing things other
people have done, not just from your own experience. Cause
that is the challenge right now, I think most of that work
is based on your own individual experience, it's hard to
harvest from other people's experience.

|T|5:25| |R|W| Right. So when that reuse is exhausted,
then you will try to think how are you going to design new
elements in order to...|/R|W|
|T|5:38|  And even then, you'll probably try to break them
down into things that seem familiar or try to attack it
using approaches you've used in the past.|/Q|C3|...|/Q|D5|
|/D|8-3 & 5-2004| |/P|2| |/I|name U|
```

Annotations may be nested and overlapping without any specific order. The same section could be annotated multiple times if it is relevant to multiple questions. If there is any doubt on the relevance of a response to a specific question, the general guideline is to err on the side of having more and annotating it as relevant. At this point, the annotation process was performed by the two authors who did the transcription and a short-term student project member[2]. Each one had gone through the entire body of the interview data and had annotated the data independently. Comparisons and discussions followed among themselves after completing each interview and the final annotated version was created collectively. Again, where there is disagreement, we err on the side of having more rather than less.

Partition The final annotated versions were run through AWK and Shell scripts. The scripts parsed all of the annotated transcriptions and gathered all the sections that are relevant to each question into a single file. Information about the source of each section is inserted at the beginning.

Here is an example from a portion of the generated file pertaining to question *B1* (see section 3.1 for details). *W* in the text refers to an interviewer.

```
<name U> 4, 8-3 & 5-2004 25:34
I think at some point you have to start looking at the
lines between systems and how they work together. And you
would see if something was very cluttered and messy in
terms of messaging and communication and that, then it's
probably a poor design, a poor architecture. I think
simplicity is actually the goal.
-------------------------
<name V> 5, 8-6-2004 0:50
My simple definition, essence of software architecture,
is the organization or interaction of components.
-------------------------
```

[2]The student did not participate in annotating the last interview as the student completed her work term.

```
<name X> 7, 7-15-2004 39:18
Does it enable everyone with a vested interested to do
what they need to do? That's really the bottom line. So
from a user's point of view, do they get something they
want to use? For one thing, product. Do they get
something they want to use, yes. So the architecture
enables the company to deliver that. Do the developers
hit their times and schedules without working more than
40 hour weeks? Do the professional services guys have a
certain level of confidence that they can go in with the
customer and implement stuff they say and sign the deals?
Are the sales guys confident in what they're demo-ing
that they're actually going to fulfill a need? Do the
board of directors see a sustainable business?
-------------------------
<name Y> 2, 8-12-2004 0:01
Sometimes people say architecture is simply the hard
parts of the system or the hard to change parts of the
system. And while there is some truth in this, its not
fundamental, it is more kind of accidental.
-------------------------
<name Z> 1, 7-8-2004 13:24
So I think the less constrained the better, from an
architect's perspective.
-------------------------
<name Z> 1, 7-8-2004 13:33
You want to describe the things that are important and
just leave the other stuff up to the developers.
-------------------------
<name Z> 1, 7-8-2004 15:36
[W: "What is the downside of over-constraining?"]
Well it virtually closes some implementation options, some
things that a developer might be able to do if it were
less constrained and have a more optimal solution at the
implementation level but you've done something at the
architecture that precludes that.
-------------------------
<name Z> 1, 7-8-2004 16:13
For a good abstraction, you have to make sure you capture
and constrain and express the things that are really
important at that level and just leave everything else,
defer it till later, let somebody else worry about it.
```

Hand Selection The partitions gathered by the scripts will be thoroughly reviewed and processed so that only the most relevant answers to the questions are kept for analysis.

4. VALIDITY

Three issues are critical in empirical studies [31]: construct [24], internal, and external validity. The discussions of each are provided below.

4.1 Construct Validity

There are two perspectives that contribute to the construct validity in this case study. One is on the coverage of the questionnaire, and the other is on the abstractions employed.

The goal in designing the questionnaire is to be both thorough and broad. The questionnaire was initially drafted by one author based on brainstorming and underwent a number of reviews by each author. Reviews were carried out among the authors after each interview session where revisions were applied whenever necessary to either clarify the questions or fill in the gaps between the questions.

We used two basic abstractions in this work: requirements and architecture. We asked each interviewee what he/she considers to be the meaning of each one. There were no issues understanding the construct of requirements, but there were with architecture. We were careful and sensitive about that. One interesting thing that came out of the interview is that the descriptive vs prescriptive notions are understood inversely in practice. Practitioners considered *prescriptive* to mean complete detail and *descriptive* to mean sketchy and incomplete. We used *descriptive* to mean the complete details of the architecture and what is not included is not allowed, and *prescriptive* to mean the critical parts and constraints that must be present and leaves the rest open. Once discussed, the interviewees agreed with us that *prescriptive* approaches (based on our definition) should be used with the exception of one who believes that designs should be as complete and descriptive as possible.

4.2 Internal Validity

Semi-structured interviews may suffer from the problem of "leading". This may make the data collected less objective than it should be. The problem arises because we have young researchers involved who are learning how to do empirical research. On the other hand, we know where this occurs and can mitigate that problem by being careful in using the results in these contexts. In addition, this is an educational research project whose secondary objective is to teach graduate students and it is inevitable that some mistakes will be made due to their lack of experience in research. Moreover, we have all interviews transcribed, and when we spot that there is too much "leading", we will use other data instead or note the context of the subjects' comments.

Not all questions are answered in each interview due to time limits. Moreover, during the interviews, we did not keep track of which questions had been answered since we preferred to follow the flow of the discussion and stay as open-ended as possible rather than obeying the strict order of the questionnaire. This raises a consistency issue among interviews. However, this is a case study not a survey; we are looking for collective agreement and disagreement. After the initial analysis, we may follow up with the subjects to supply missing data deemed critical.

4.3 External Validity

So far, we have a variety of domains represented (though quite a few admittedly from the same international organization). We recognize that there may be some bias introduced. However, this work is ongoing, and we will choose more diverse subjects in the future.

5. ANALYSIS

Our analysis and summary of the data will be focused on looking for the following information: (i) specific domain related differences; (ii) agreement and disagreement on answers; (iii) critical issues; and (iv) further questions to be considered. Our general approach is oriented around three perspectives: (a) question specific; (b) interviewee specific; and (c) global account of both of the above.

6. ISSUES ABOUT EVIDENCE

One of our goals is to gain an understanding of what is done in the practice to judge, form and evaluate our research. To achieve this, we direct particular attention to combining, weighing, judging and using evidence. In the following text,

we discuss our approach towards combining evidence and briefly comment on weighing and judging evidence. We also review a history of empirical research that we are involved in, how evidence was used in this work, and the contributions made to the research and practice communities.

6.1 Combining Evidence

The work described in this paper is a multiple-case study in which we interviewed nine practicing architects. This approach allows us to collect more evidence and acquire a broader understanding of the practical meaning of various topics of interest, than we would from a highly specialized and focused view with a single subject. Further, it helps in understanding and analyzing what the architect say they do in practice. There are two dimensions to the units of analysis. One is an individual architect (subject). The other is the individual question from our questionnaire. Our data analysis will follow both threads and provide both the overall understanding of each question from every subject's point of view and that of each subject on all questions.

The interviews help us to understand what practicing architects think of topics on requirements and architecture, and how they describe their practice. However, what they say they do may not match what they actually do. To enhance the strength of our evidence, we plan to use two approaches to satisfy a systematic pluralism approach [31]. The first is to use the "participant observation" technique observing and recording daily activities and interactions over a period of time. In this way, we collect firsthand information on what they actually do. Field notes are the main tool for recording. However, to minimize observer's bias, audio recording and transcriptions will also be used whenever appropriate. These field records will provide evidence on what the architects *actually do* as opposed to what they have *said they do*. Another approach we plan to take is to acquire their process descriptions and documentation for requirements engineering and architectural design, and use content analysis [33] to extract what architects are *supposed to do*.

With these three sets of data, we will have better coverage and be able to identify inconsistencies among them. As in the multiple case studies, we look for agreement in the evidence first; the combined agreement provides very strong evidence about practice. Where there are disagreements in the data, we will weigh the observation results the highest followed by the interview data, and the documentation records the lowest. We do this because people often idealize their processes when discussing them, and documented processes are often not followed or may not reflect current practice. Collectively, the three approaches described above form a *process-based method* for accumulating and combining evidence. The strength of this method is that it combines evidence about what they are supposed to with what they say they do and what they actually do.

In the time studies [28, 3], we used various empirical approaches to gain a better understanding of the development process structure to reduce the development interval. The experiments were performed using different forms of data collection to report on how the time is spent by the developers in the development process. The different levels of granularity in time form a triangulation for the data.

The initial longitudinal study reported the activities of a single developer over the entire development of more than 32 months at the granularity of one day. The primary result of that study was that the developer was blocked 60% of the time. To determine the relevance of that result, a cross-sectional study was done where the developers reported their daily progress in half-hour increments via self reports at the end of the day. The 40% effectiveness result was confirmed in the second study, but the 60% ineffectiveness was blurred due to both blocking and context switching. To determine the reliability of the self reports, a direct observation study was done reporting the activities of the developers at the minute level of granularity. The most interesting result at these finest granularity was the 75 minutes per day on average of short (i.e. $\sim$ 3 minutes) unplanned and informal interactions not observable at the other levels of granularity. Collectively, these three approaches form a *time-based method* for accumulating and combing evidence.

In our search for further evidence about transforming requirements to architecture, we will carry out experiments on two topics and study what works and why: (i) non-functional patterns and transformation; and (ii) methods and tools for transformation (which could be either existing or new). Finally, we plan to combine all of the evidence described above, taking into account both the current practice and research, and construct a grounded theory [9] of requirements and architecture.

6.2 Using Evidence

Collecting and using evidence is essential in driving or informing four areas of software engineering: research, practice, empirical study design and software engineering theory. We first look at one such study and how its evidence informed research and practice, and then indicate how it could effectively inform experimental design. We then show how empirical evidence informed the creation and deployment of a code inspection tool. Finally, we illustrate how evidence from practice can be used to create models for software development processes and organizations.

Perry and Evangelist studied interface faults [25] which uncovered 16 categories of faults in the *prima facie* set of interface error reports. This is the first reported work that treated interface faults as a separate class of software errors. The study provided insight to the requirements and further studies of techniques and methods that may reduce interface faults. It sets the foundation for both Perry's seminal work on researching and developing the Inscape Environment [23] and Rosenblum's most influential paper on annotations aimed directly at affecting practice [30].

In the Inscape work [23], Perry prototyped an integrated software development environment (SDE) intended to address two fundamental problems in building software systems: evolution and scale. The goal was to provide a practical application of formal methods in building large, evolutionary software systems with large groups of people through the use of formal interface specifications. The research and development of this SDE was driven by the data collected in [25]. For example, Inscape has structured exception handling in response to the evidence which showed that 20% of the faults were errors with exception handling; in addition,

because the evidence also showed problems with obligations accounted for 10% of the faults, the formal interface specifications used explicitly contain the idea of an obligation.

Rosenblum, in his annotation work [30], investigated the issue of why programming with assertions as a development technique and tool had little widespread use in practice. He described an assertion tool developed by himself. Based on his experience of using that tool, he classified the kinds of assertions that were most effective in uncovering the kinds of faults described in [25]. The examples used in his paper were taken from the evidence gathered in [25].

Experiments in various forms of code evaluation such as inspections (e.g. [7]) and testing often use lab settings where the examples of code are "seeded" with faults. The heart of the design then is to use various methods, techniques and tools to find these errors while various measures are collected to help compare and evaluate among different approaches. One of the threats to internal validity in all these designs is the lack of justification for the kinds and distributions of the faults seeded. Fault studies such as the ones mentioned here can be used to strengthen the internal validity, and hence also the external validity of these designs, i.e. to justify the faults seeded in much the same way that Rosenblum [30] justified the annotations used.

A typical technological-based approach to designing a distributed code inspection tool would include a CSCW (Computer Supported Cooperative Work) component and mechanisms for describing structures and techniques to provide a general purpose tool "for all seasons". The resulting tool would likely be complex and hard to use. The goal of Perry et al. was instead to use an empirically-based approach to design the simplest possible tool to support distributed code inspections, reduce inspection effort, and not compromise inspection effectiveness. They did this by collecting evidence on inspections with or without meetings [22]. This resulted in a successful implementation and deployment of a tool called HyperCode that was then used in at least twelve projects in six countries [26].

Grinter et al. [11] recorded how organizations coordinate distributed Research and Development (R&D) work among geographically separated sites. Based on the data, they identified four models of distributing R&D work across multiple sites, described the benefits, difficulties and the coordination mechanisms associated with each model. These results are used as a basic framework in understanding how to solve coordination problems in distributed R&D work across multiple geographically separated sites and how to deploy effective tools into such a context.

Collectively, the above research has shown the invaluable utility of empirical evidence for driving both research and practice. In this work, we expect to achieve similar results to those in [11].

7. CONCLUSIONS

In this paper, we presented the design of our interview-based multiple-case study as our first step in deepening our understanding of the transformation from requirements to architecture. We described our process from the preparation to the evidence chain and evidence trail, and provided actual data as examples. We discussed validity issues from three perspectives (construct, internal and external) and outlined our analysis focus. Finally, we addressed meta issues on combining and using evidence in software engineering, discussed data triangulation, and presented two methods for accumulating evidence.

Our next step is to analyze all the data thoroughly, construct a framework for the transformation from requirements to architecture, and use the framework to help us identify and build practical techniques, methods, processes, and tools. Other future work includes carrying out experiments on patterns, methods and tools, and constructing a grounded theory [9] for requirements and architecture.

Acknowledgements

We thank all of our interviewees for their participation and contribution: Bob Blakley, Dale Churchette, Ric Holt, Charles Krueger, Sharon Lymer, Joe McIntyre, Sridhar Muppidi, Dave Stokes, and Stuart Thompson. We gratefully acknowledge our reviewers for their feedback and suggestions on strengthening the paper. This research was supported in part by IBM CAS Fellowship and NSF CISE Grant CCR-0306613.

8. REFERENCES

[1] In J. Castro and J. Kramer, editors, *"The First International Workshop on From Software Requirements to Architectures (STRAW'01)"*. At ICSE'01.

[2] In D. M. Berry, R. Kazman, and R. Wieringa, editors, *"The Second International Software Requirements to Architectures Workshop (STRAW'03)"*. At ICSE'03.

[3] M. Bradac, D. Perry, and L. Votta. "Prototyping A Process Monitoring Experiment". *IEEE Transactions on Software Engineering*, 20(10):774–784, Oct 1994.

[4] M. Brandozzi and D. E. Perry. "From Goal-Oriented Requirements to Architectural Prescriptions: The Preskriptor Process". pages 107–113. 2003. In [2].

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas. "Goal-Directed Requirements Acquisition". *Science of Computer Programming*, 20:3–50, 1993.

[6] R. Dromey. "Architecture as an Emergent Property of Requirements Integration". pages 77–84. In [2].

[7] A. Dunsmore, M. Roper, and M. Wood. Systematic object-oriented inspection an empirical study. In *Proceedings of ICSE '01*, pages 135–144, 2001.

[8] D. Garlan. "Software Architecture: a Roadmap". In *ICSE - Future of SE Track*, pages 371–380, 2000.

[9] B. G. Glaser and A. L. Strauss. *"The Discovery of Grounded Theory: Strategies for Qualitative Research"*. New York: Aldine, 1999.

[10] B. Graaf, M. Lormans, and H. Toetenel. "Embedded Software Engineering: The State of the Practice". *IEEE Software*, 20(6):61–69, 2003.

[11] R. Grinter, J. Herbsleb, and D. Perry. "The Geography of Coordination: Dealing with Distance in R&D Work". In *Proc. Int'l ACM SIGGROUP Conf. Supporting Group Work*, pages 306–315, 1999.

[12] P. Grnbacher, A. Egyed, and N. Medvidovic. Reconciling software requirements and architectures: The cbsp approach. In *Proceedings of RE'01*, pages 202–211, 2001.

[13] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti. "Relating Software Requirements and Architectures using Problem Frames". In *Proceedings of RE'02*, 2002.

[14] A. Hickey and A. Davis. "Elicitation Technique Selection: How Do the Experts Do It?". In *Proceedings of RE'03*, 2003.

[15] M. Jackson. *"Problem Frames: Analysing and Structuring Software Development Problems"*. Addison-Wesley, 2001.

[16] D. Jani, D. Vanderveken, and D. E. Perry. "Experience Report: Deriving Architectural Specification from KAOS Specification". Available at `http://www.ece.utexas.edu/~perry/work/papers/R2A-ER.pdf`, Dec. 2003.

[17] N. Leveson. "Intent Specifications: An Approach to Building Human-Centered Specifications". *IEEE Transactions on Software Engineering*, 26(1), 2000.

[18] W. Liu and S. Easterbrook. "Eliciting Architectural Decisions from Requirements using a Rule-based Framework". pages 94–99. In [2].

[19] G. Mustapic, A. Wall, C. Norstrm, I. Crnkovic, K. Sandstrm, J. Frberg, and J. Andersson. "Real World Influences on Software Architecture - Interviews with Industrial Experts". In *Proceedings of WICSA'04*, 2004.

[20] J. Mylopoulos et al. "Tropos - Requirements-Driven Development for Agent Software". `http://www.troposproject.org/`, 2004.

[21] B. Nuseibeh and S. Easterbrook. "Requirements engineering: a roadmap". In *ICSE - Future of SE Track*, pages 35–46, 2000.

[22] J. M. Perpich, D. E. Perry, A. A. Porter, L. G. Votta, and M. W. Wade. "Anywhere, Anytime Code Inspections: Using the Web to Remove Inspection Bottlenecks in Large-Scale Software Development". In *Proceedings of ICSE'97*, 1997.

[23] D. E. Perry. "The Inscape Environment". In *Proceedings of ICSE'89*, 1989.

[24] D. E. Perry. "An Empirical Approach to Design Metrics and Judgements". In *New Vision for Software Design and Production Workshop*. Vandebilt University, Dec 2001.

[25] D. E. Perry and M. Evangelist. "An Empirical Study of Software Interface Faults—An Update". In *Proceedings of 12th HICSS*, volume II, pages 113–126, January 1987.

[26] D. E. Perry, A. Porter, M. W. Wade, L. G. Votta, and J. Perpich. "Reducing inspection interval in large-scale software development". *IEEE Transactions on Software Engineering*, 28(7):695–705, July 2002.

[27] D. E. Perry, A. A. Porter, and L. G. Votta. "Empirical studies of software engineering: a roadmap". In *ICSE - Future of SE Track*, pages 345–355, 2000.

[28] D. E. Perry, N. A. Staudenmayer, and L. G. Votta. "Understanding Software Development Processes, Organizations and Technologies". *IEEE Software*, July 1994.

[29] L. Rapanotti, J. G. Hall, M. Jackson, and B. Nuseibeh. "Architecture-driven Problem Decomposition". In *Proceedings of RE'04*, pages 73–82, 2004.

[30] D. S. Rosenblum. "Towards a Method of Programming with Assertions". In *Proceedings of ICSE'92*, volume 12, pages 92–104, 1992.

[31] R. Rosenthal and R. L. Rosnow. *"Essentials of Behavioral Research: Methods and Data Analysis"*. McGraw Hill (Series in Psychology), second edition, 1991.

[32] K. Smolander, K. Hoikka, J. Isokallio, M. Kataikko, and T. Mkel. "What is Included in Software Architecture? A Case Study in Three Software Organizations". In *Proceedings of IEEE Inter. Conf. on the Engineering of Computer-Based Systems (ECBS)*, 2002.

[33] S. Stemler. "An Overview of Content Analysis". *Practical Assessment, Research & Evaluation*, 7(17), 2001.

[34] A. van Lamsweerde. "From System Goals to Software Architecture". In M. Bernardo and P. Inverardi, editors, *Formal Methods for Software Architectures*, pages 25–43, 2003.

[35] A. van Lamsweerde, R. Darimont, and E. Letier. "Managing Conflicts in Goal-Driven Requirements Engineering". *IEEE Transactions on Software Engineering*, 24(11):908–926, 1998.

[36] K. J. Vicente. *Cognitive Work Analysis*. LEA, 1999.

[37] E. Yu. *"Modelling Strategic Relationships for Process Reengineering (Agents, Goals)"*. PhD thesis, University of Toronto (Canada), Department of Computer Science, 1995.

[38] E. Yu et al. "Goal-oriented Requirement Language (GRL)". `http://www.cs.toronto.edu/km/GRL/`, 2004.