

Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes

Ann M. Hickey
ahickey@uccs.edu

University of Colorado at Colorado Springs

Alan M. Davis
adavis@uccs.edu

Abstract

By its very nature, software development consists of many knowledge-intensive processes. One of the most difficult to model, however, is requirements elicitation. This paper presents a mathematical model of the requirements elicitation process that clearly shows the critical role of knowledge in its performance. One meta-process of requirements elicitation, selection of an appropriate elicitation technique, is also captured in the model. The values of this model are: (1) improved understanding of what needs to be performed during elicitation helps analysts improve their elicitation efforts, (2) improved understanding of how elicitation techniques are selected helps less experienced analysts be as successful as more experienced analysts, and (3) as we improve our ability to perform elicitation, we improve the likelihood that the systems we create will meet their intended customers' needs. Many papers have been written that promulgate specific elicitation methods. A few have been written that model elicitation in general. However, none have yet to model elicitation in a way that makes clear the critical role played by knowledge. This paper's model captures the critical roles played by knowledge in both elicitation and elicitation technique selection.

1. Introduction

Requirements elicitation is recognized as one of the most critical, knowledge-intensive activities of software development [1]; poor execution of elicitation will almost guarantee that the final project is a complete failure. Since project failures are so rampant [2], it is quite likely that improving how the industry performs elicitation could have a dramatic effect on the success record of the industry [3]. Improving requirements elicitation requires us to first understand it. Although many papers have been written that define elicitation, or prescribe a specific technique to perform during elicitation, nobody has yet defined a unified model of the elicitation process that emphasizes the role of knowledge.

To better understand the importance of the current paper, let us contrast its goal with the goal of the many dozens of writings, e.g., [4, 5, 6], that present a specific methodology for elicitation broken down into multiple steps. Studying the steps provides the reader with an understanding of one particular way of doing elicitation. Some writings, e.g., [7, 8, 9, 10, 11, 12, 13, 14, 15], even provide limited insight into *when* a methodology or specific elicitation technique might or might not be applicable. This paper provides a model for elicitation in general. It then extends that model to include a model of the elicitation technique selection process. This technique selection model will serve as the basis for future research to integrate the tacit knowledge expert analysts¹ use during elicitation and elicitation technique selection into a knowledge-based system for less experienced analysts to help them select the most appropriate technique for their situation. Better technique selection will improve the quality of the requirements elicitation process and increase the success of software development projects.

In this paper we use specific definitions for some terms:

- *Requirements Process.* The activities that when performed result in an understanding and documentation of the desired external behavior (i.e., the requirements) of a system.
- *Process Model.* A representation showing the processes to be performed in order to achieve some well-defined goal.
- *Technique.* A documented series of steps along with rules for their performance and criteria for verifying completion. A technique usually applies to a single process in a process model. Sometimes includes a notation and/or a tool.
- *Methodology.* A process model, along with documented techniques and/or tools to support each process in the model.

¹ Generically, any individual who performs elicitation. Also known by many other names, e.g., requirements engineer.

2. Overview of the research domain

2.1. Software development

Software development is the activity of creating a software system that when used, solves some hitherto unsolved problem. Classic software development follows a well-defined series of phases, typically called a waterfall model [16] (see Figure 1). More commonly, software development is performed iteratively, resulting in a time series of successively more sophisticated products (see Figure 2). In the former case, requirements activities are performed ostensibly at the beginning of the life cycle. However, with the inevitable onslaught of constantly changing needs, requirements activities need to be performed regularly. In the latter case, requirements activities are performed ostensibly at the beginning of each iteration. As in the former case, requirements change constantly. However, if the iterations are close enough together, it is usually easier to defer requirements changes

to the beginning of a subsequent iteration and thus little time need be expended within any iteration performing additional requirements activities.

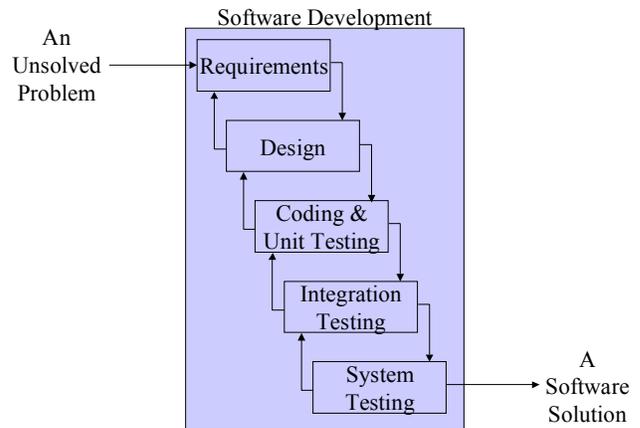


Figure 1. Waterfall model of software development

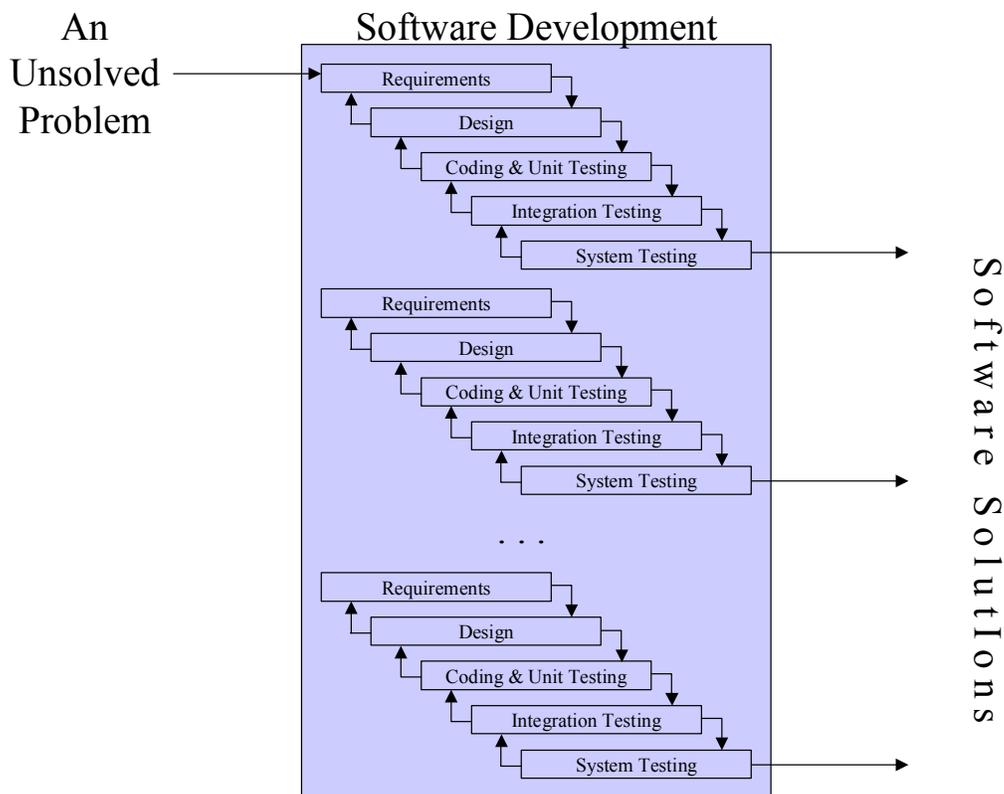


Figure 2. Iterative model of software development

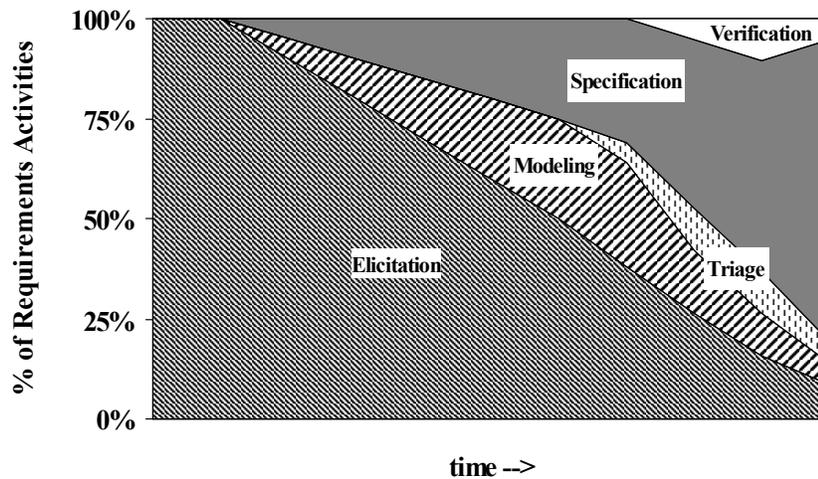


Figure 3. Parallel model of the requirements process

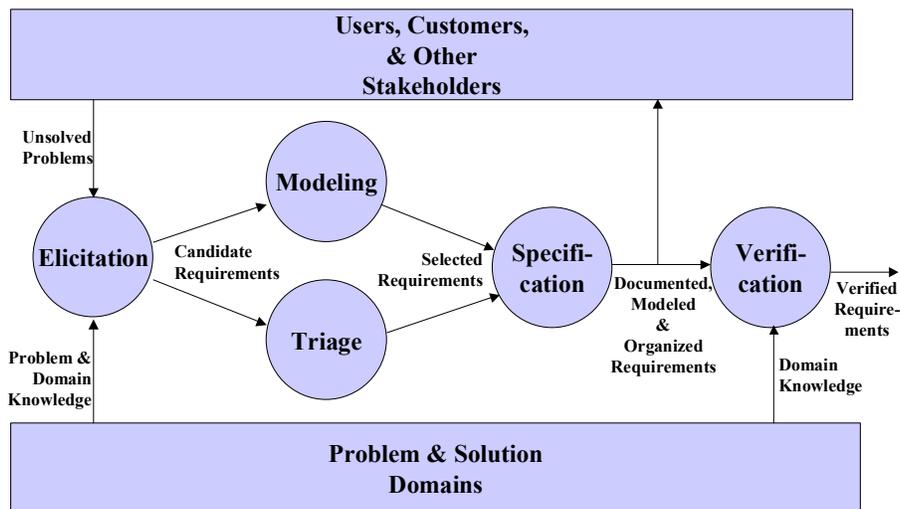


Figure 4. Data flow among requirements activities

2.2. Requirements Process

The requirements process is also often described as a series of activities such as elicitation, modeling, triage, specification, and verification²:

² There is little uniformity in the industry concerning names given to these activities [17]. For example, to paraphrase [18], Davis [8] defines two activities: problem analysis and product description. Graham [19] defines two activities: requirements elicitation and requirements analysis. Zave [20] defines three activities: elicitation, validation, and specification. Jarke and Pohl [21] define three activities: elicitation, expression, and validation. Later, Pohl [22] defines four activities: elicitation, negotiation, specification/documentation, and validation/verification. Finally, Thayer and Dorfman [23] define five activities: elicitation, analysis, specification, verification and management.

- *Elicitation*. Learning, uncovering, extracting, surfacing, and/or discovering needs of customers, users, and other potential stakeholders.
- *Modeling*. Creating and analyzing models of requirements, with the goals of increasing understanding and searching for incompleteness and inconsistency.
- *Triage*. Determining which subset of the requirements ascertained by elicitation are appropriate to be addressed in specific releases of a system.
- *Specification*. The documentation of the desired external behavior of a system.
- *Verification*. Determining the reasonableness, consistency, completeness, suitability, and lack of defects in a set of requirements.

The majority of existing models of the requirements process show it as an ordered sequence of activities. In reality, requirements activities are not performed sequentially, but in parallel as shown in Figure 3. Other models of the requirements process take a different view or add additional information. Figure 4 (adapted from [24]) shows how information flows among the activities of requirements. Playle and Schroeder [25] emphasize the automated tools that support each requirements activity. Others model the requirements process from an input/output perspective, e.g., [10]. Gaska and Gause [26] add controls and mechanisms to their model while Hofmann and Lehner [3] explore how team knowledge, resources, and processes contribute to the success of the requirements process. Other models of the requirements process focus on a specific methodology (e.g., Volere Requirements Process Model [27]).

2.3. Requirements elicitation

As mentioned earlier, elicitation is all about determining the needs of stakeholders. Most models of requirements elicitation focus on specific methodologies or techniques. For example, the Robertsons' Volere requirements methodology includes a detailed process model of its requirements elicitation activities with inputs, outputs, and recommended techniques for each activity [27]. Several researchers, e.g., [28, 29], have developed specific process models that define how to use scenarios for requirements elicitation. Sutcliffe and Ryan [30] present a model of elicitation that combines scenarios, prototypes and design rationale. Sommerville et al. [31] describe their approach for using viewpoints to elicit requirements. Some of the most detailed elicitation process models describe collaborative requirements workshops such as JAD [6].

Very few general models of elicitation exist [32]. Some authors provide overall principles for elicitation, e.g., [33, 34]. Others describe general approaches (e.g., top-down vs. bottom-up). A few focus on one specific view of the process. Maciaszek [35] describes the influences during requirements elicitation by showing how analysts, domain experts, and customers interact to provide domain knowledge and use case requirements, which are used to produce business class and use case models. Dean et al. [36] also take a model-centric view in their process model of Collaborative Requirements Elicitation and Validation (CREV), which defines how activity, data, and scenario models work together with prototypes to generate requirements. Gottesdiener [1] (a) focuses on requirements workshops, (b) presents models of the inputs/outputs and how various requirements models can be used to answer the 'six great focus questions' (who, what, why, when, where, how), and (c)

provides a few sample process models. Another general model of elicitation defines paths of communication that ultimately result in increased knowledge of requirements to be addressed. Adapted from [36], Figure 5 shows how specific elicitation techniques exercise various communication paths between parties. One clockwise circuit around the wheel represents one step of an elicitation methodology. Our view is that the paths around this wheel should not be predetermined, but that each step (i.e., next segment of the path) be selected as a function of what has already been learned (known requirements), as well as current characteristics of the problem domain, solution domain, and project. Browne and Rogich [32] also look at a communication-based model of elicitation, but focus on the cognitive aspects of user/analyst interaction.

2.4. Requirements elicitation technique selection

Requirements elicitation is generally performed using an elicitation methodology or a series of techniques. Many such methodologies and techniques exist, all with the common aim to assist analysts in understanding needs [13]. Although some analysts think that just one methodology or just one technique is applicable to all situations, one methodology or technique cannot possibly be sufficient for all conditions [10, 13, 14, 37, 38, 39].

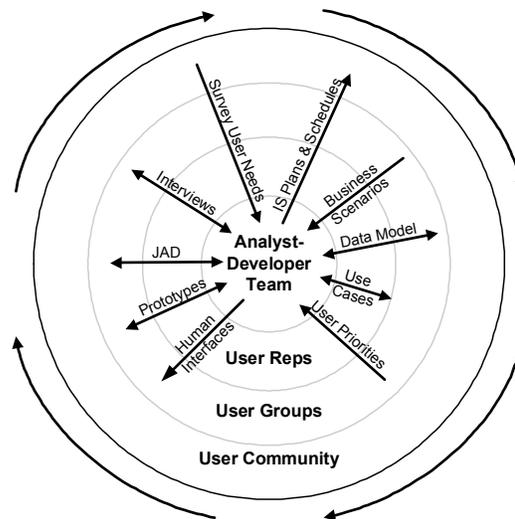


Figure 5. Communication channels in elicitation

Analysts select a particular elicitation technique for any combination of four reasons: (1) It is the only technique that the analyst knows, (2) It is the analyst's favorite technique for all situations, (3) The analyst is following some explicit methodology, and that methodology prescribes a particular technique at the current time, and (4) The analyst understands intuitively that the technique is effective in the current circumstance.

Clearly the fourth reason demonstrates the most "maturity" by the analyst. We hypothesize that such maturity leads to improved understanding of stakeholders' needs, and thus a higher likelihood that a resulting system will satisfy those needs. Unfortunately, most practicing analysts do not have the insight necessary to make such an informed decision, and therefore rely on one of the first three reasons.

3. A new model of requirements elicitation

3.1. Introduction to the new model

In order to improve our knowledge of the elicitation process, elicitation methodologies and techniques, and the elicitation technique selection process, we propose a new unified model of elicitation. In previous sections, we defined the scope of elicitation, and presented other researchers' models of elicitation. As we saw, two classes of models have been documented: (a) those that captured a specific methodology or technique, and (b) those that modeled elicitation in general. In the first class, the models possess a variety of weaknesses:

1. Each describes a *specific* elicitation methodology or technique
2. Each prescribes a specific series of steps, each with its own predefined technique. In effect they are saying "one size [methodology] fits all."
3. Each fails to model either the technique selection process or the situational characteristics that drive that decision process.

In the second class, the models possess different weaknesses:

4. Most have underlying, but unstated, assumptions. One noteworthy exception is [32].
5. None discuss the role of knowledge in performing elicitation or in selecting elicitation techniques. This knowledge includes (a) the current problem, solution, and project characteristics, (b) the awareness of which requirements are known and which are still to be determined, and (c) knowledge of the relationship of the current problem, solution, and project characteristics and the state of the requirements to the selection of an elicitation technique.³

To overcome these weaknesses, Figure 6 expands the elicitation activity from Figure 4 by adding a new elicitation technique selection process along with its driving characteristics. Note that the elicitation technique selection process is driven by problem, solution, and

³ And to make matters worse, guidance of this type is not even available in current textbooks.

project domain characteristics as well as the state of the requirements (The "right" technique to apply in a given situation must be a function of what requirements we already know and what requirements we still need to know; after all, different techniques are good at uncovering different kinds of requirements).

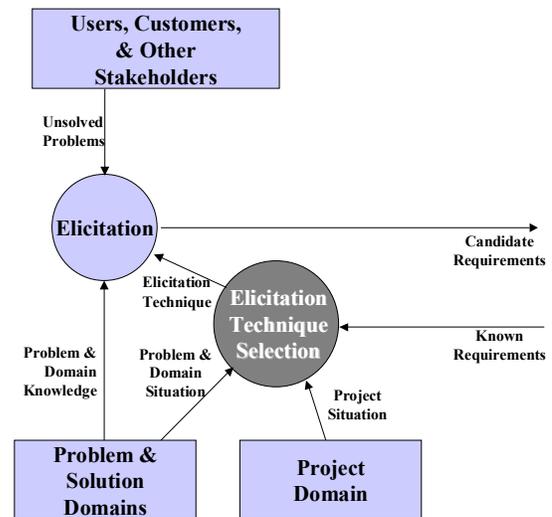


Figure 6. Details of elicitation activities

3.2. The model

This section of the paper describes a model of elicitation that represents a generalization of all known elicitation methodologies and techniques. It

- Explicitly highlights the role knowledge plays in performing both elicitation and elicitation technique selection.
- Provides a unified framework for understanding the purpose and role of requirements elicitation in software development,
- Describes how any elicitation methodology could be represented in terms of that model,
- Shows what assumptions existing elicitation methodologies make about the situation,
- Identifies how easily one can tailor existing methodologies for unique situations, and
- Shows how one can create new elicitation methodologies easily, by defining situational characteristics and then observing and recording the resultant instances of methodologies.

On any project, an analyst performing elicitation moves through a series of activities. The purpose of each activity is to bring the parties closer and closer to a common understanding of the requirements they wish to address. This series of activities can be viewed as the application of a series of mathematical functions, *elicit*₁, *elicit*₂, . . . , each of which creates new requirements by

applying an elicitation technique. Thus, each step of elicitation can be defined as

$$elicit_i(R_i, S_i, t_i) \rightarrow R_{i+1}, S_{i+1}$$

That is, at step i of the elicitation process, elicitation applies technique t_i when situation S_i exists and R_i captures the current state of knowledge of the requirements we need to understand. The result is a new state of the requirements R_{i+1} and a new situation S_{i+1} . Note that $t_i \in T$, the set of all known elicitation techniques.

Clearly this slice of the model addresses problems 1 and 2 above; the methodology and the techniques applied at each step are *not* predefined, but instead the elicitation technique at each step should be selected because it is the most applicable to the current situation and/or is the most likely to uncover requirements that are currently absent. This selection of an elicitation technique should consider:

- What requirements are known and what are not yet known. These are likely to change dynamically throughout the life of the project. This is represented in our model as R_i .
- Characteristics of the problem domain. These are usually static throughout the life of a project.
- Characteristics of the solution domain. These are likely to change whenever a new type of solution to the problem is proposed.
- Characteristics of the project. These are likely to change whenever culture or management changes. This, along with the previous two characteristics, are collectively represented in our model as S_i .

Elicitation technique selection can be modeled as a selector function:

$$\sigma(R_i, S_i, \chi(T)) \rightarrow \{t \in T \mid t \text{ is applicable in situation } S_i \text{ when the current state of the requirements is } R_i\}$$

given characteristics of all elicitation techniques, $\chi(T)$. These characteristics capture the *inherent* aspects of elicitation techniques, such as whether they aid in reducing ambiguity, whether they are effective at helping people converge on a solution, whether they help resolve conflict, whether they help to raise new issues, and so on. They are static and identical for all projects. The goal of the selector function is to identify the best possible match between the characteristics of the techniques and the current state of the requirements and situation. For example, if the requirements are unclear, techniques that reduce ambiguity may be helpful.

Since the *elicit* function requires just *one* elicitation technique and the above selector function creates a set of applicable techniques, we must also define a personal selector function,

$$\pi(\{t\}, P) \rightarrow t_i \in \{t\}$$

where analysts apply their own personal preferences, P , to select just one technique from the set of applicable elicitation techniques.

Thus, requirements elicitation at step i becomes,

$$elicit_i(R_i, S_i, \pi(\sigma(R_i, S_i, \chi(T)), P)) \rightarrow R_{i+1}, S_{i+1}$$

Note that this combined function models the combination of the two bubbles in Figure 6.

3.3. How to use the model

Every requirements elicitation methodology, M_j , containing n steps can be characterized as an instance of a series of n *elicit* steps, i.e.,

$$M_j = elicit_1, elicit_2, \dots, elicit_n$$

and the state of the requirements (i.e., those that have been uncovered and those that have not been uncovered) as a result of applying methodology M_j are.

$$R_n(M_j) = elicit_n(\dots(elicit_2(elicit_1(R_1, S_1, t_{j1}), t_{j2}), \dots), t_{jn})$$

where R_1 is the state of the requirements at the beginning of the project, S_1 is the situation at the beginning of the project, and $t_{j1}, t_{j2}, \dots, t_{jn}$, are the steps prescribed by methodology M_j .

Notice how the function highlights the assumptions that every methodology makes. When the methodology states that the analyst should perform some technique t_i at step i , the methodology is making the assumptions that S_i is true and that requirements state is R_i ! But given all the variation among people and problems, how could such an assumption possibly be made *a priori*?

To tailor an existing methodology so that it makes sense, follow this simple procedure at each step i :

1. Examine the state of the requirements, R_i , including both what requirements are known and what requirements still need to be discovered
2. Examine the characteristics of the problem, the solution, and the project, S_i .
3. Determine if the technique t_i being suggested by the methodology is a member of $\sigma(R_i, S_i, \chi(T))$. If so, you should proceed with the application of t_i as prescribed by the methodology.
4. If the technique t_i being suggested by the methodology is not a member of $\sigma(R_i, S_i, \chi(T))$, then select an alternative technique, i.e., $\pi(\sigma(R_i, S_i, \chi(T)), P)$.

To create a new methodology for your unique situation, or if you do not want to “follow a methodology” but just want to do elicitation in a way that makes most sense, follow this simple procedure at each step i :

1. Examine the state of the known requirements, R_i .
2. Examine the characteristics of the problem, the solution, and the project, S_i .
3. Using your personal preferences, P , select a technique t_i out of the set of all applicable techniques, i.e., apply the function $\pi(\sigma(R_i, S_i, \chi(T)), P)$.

3.4. An example of using the model

The Collaborative Software Engineering Methodology (CSEM) was created to support incremental development of complex systems with large, diverse user populations [36]. CSEM divides the requirements process into six primary activities (which may be repeated, eliminated, or otherwise tailored based on the project situation) and provides specific recommendations on what techniques and tools can be used to conduct those activities. The purpose of showing this example is to demonstrate how any documented methodology can be described in terms of our model.

The following example describes the first three of the six steps of the CSEM requirements process using the terms of our new unified model. It makes a variety of assumptions, including

- That the selector function, σ , has already been created. In reality, we have not yet constructed this function (see later section on future research). This function will be driven by the static values of $\chi(T)$.
- That we are applying CSEM to the first increment of a new, complex information system.
- That the requirements we need to elicit must identify the basic objects, functions, and states [8] of the new system.

For each elicitation step, the example describes the inputs (R_i, S_i), techniques recommended by our model (assuming σ already exists), and outputs (R_{i+1}, S_{i+1}) generated by use of the selected technique. We then compare our model's recommendations to those of the CSEM.

Elicitation Step 1.

1. Assess Requirements (R_i). Since the project has just begun, we only have a broad definition of project scope that provides a high-level description of the desired functions. We do not yet have a common understanding of the business functions, nor have we identified which specific business functions will be included in the project's scope.
2. Assess Situation (S_i). We have a large, diverse user population with common goals, but unique operating environments and a variety of legacy systems providing some of the desired functionality. We have identified a representative group of users who can travel to a face-to-face meeting.
3. Select technique (t_i). The technique selection process, σ , identifies several possible techniques based on the above characteristics:
 - Collaborative workshops rate high because of the diversity of users and the ability to gather representatives in one place.

- Various activity decomposition and modeling techniques (e.g., IDEF0) also rate high because of the need to have a common view of business activities/functions.

- Interviews are another possibility if time is not an issue and users generally agree on functions.

We apply our personal preferences and select a collaborative workshop to develop a simple activity/function hierarchy and agree on which functions are included in the scope of the project.

4. Perform $elicit_i(R_i, S_i, t_i) \rightarrow R_{i+1}, S_{i+1}$. In this case, let us assume that the collaborative workshop succeeded in eliciting the new system's functions. User representatives now share a common understanding of those functions and agree on which functions will be included in the first increment of the new system.
5. CSEM comparison. The above recommendations are comparable to the first CSEM requirements activity, Identify Business Activities, which recommends use of a Group Support System (GSS) tool to collaboratively develop an activity hierarchy or more complete IDEF0 activity model. Note that the use of our model has highlighted the assumptions that CSEM has made.

Elicitation Step 2.

1. Assess Requirements (R_i). We have a static view of the new system's functions. However, we do not have a more dynamic, state view of those functions.
2. Assess Situation (S_i). All users perform the identified functions, but they may have very different processes for doing so. A goal of the new system is to implement the 'best practices' for each function, so we need to explore process differences and reach agreement on those 'best practices.' Users who are expert in the different processes for each function have been identified, but they have no previous requirements elicitation or modeling experience.
3. Select technique (t_i). The technique selection process again rates collaborative workshops high because of the need to reach agreement on 'best practices.' Possibilities for capturing dynamic process/state information include scenarios or use cases, statecharts, and Petri nets. Because users are business, not modeling, experts, we apply our personal preferences and select a collaborative workshop to elicit scenarios.
4. Perform $elicit_i(R_i, S_i, t_i) \rightarrow R_{i+1}, S_{i+1}$. Scenarios elicited during this step describe the best way to perform each business function and define the new system's states. User representatives agree on detailed processes for each business function.
5. CSEM comparison. The above recommendations are comparable to the CSEM requirements activity,

Generate Business Scenarios, which recommends use of a Group Support System (GSS) tool to collaboratively define and agree upon business scenarios for each function included in the first increment.

Elicitation Step 3.

1. Assess Requirements (R_i). We now know function and state requirements, but we still do not know the object (data) requirements.
2. Assess Situation (S_i). We can use existing legacy systems to identify current data requirements, but users will need to reconcile discrepancies between systems and identify any new data required to support new functions and states. Users have no data modeling experience.
3. Select technique (t_i). The technique selection process recommends that we develop a data or class model to capture object requirements using information from the legacy systems. It recommends interviews or a group meeting to reconcile differences. Based on our personal preferences, we choose to create a data model and ask selected users for assistance as needed to reconcile discrepancies.
4. Perform *elicit* _{t_i} (R_i , S_i , t_i) \rightarrow R_{i+1} , S_{i+1} . The data model developed during this step identifies object requirements. We now know the object, function, and state requirements identified as needed for this system. Users agree to these requirements and agree to proceed with development.
5. CSEM comparison. The above recommendations are generally comparable to the CSEM requirements activity, Develop Data Model. However, in this case, CSEM recommends that users collaboratively review and reconcile a preliminary data model (developed by a data modeling expert) and then provide detailed entity and attribute meta-data using a Group Support System (GSS) Data Modeling tool designed to capture that information from non-modelers [36].

The preceding example demonstrates a variety of items:

1. We can use it to analyze the CSEM methodology. For example, the model makes it explicit what (perhaps) tacit assumptions CSEM is making at each step. Thus, our model could be used by CSEM researchers to better understand their own methodology.
2. We can use it to adapt the CSEM methodology. Thus, a user of CSEM could utilize our model to determine (a) the applicability of CSEM to their situation, (b) reasonable alternatives to the defined steps of CSEM, and (c) optimal selections among the alternative techniques recommended as part of the tailoring advice of CSEM.

3. We can use it to analyze our model. As we study CSEM and many other methodologies using our unified model of elicitation, we will likely discover weaknesses in our model.
4. We can use it to compare and contrast the assumptions made by CSEM vs. other methodologies.

4. Future research

The model defined in this paper has become the basis for myriad new research directions. A few are introduced here:

- *Taxonomy of Problem, Solution, and Project Characteristics*. This paper highlights the critical role played by situational characteristics, S_i . Now we need to define all those characteristics, and organize them in a way to make them easy to understand and use. This work will be based on earlier taxonomies documented in [40, 41, 42, 43], but used for quite different purposes.
- *Taxonomy of Requirements Techniques*. Once the problem, solution, and project characteristics have been defined, we need to develop a taxonomy of elicitation techniques sensitive to these characteristics. That is, if two elicitation techniques are applicable in the same situation, they should appear in the same place in the taxonomy.
- *Implementation of the Selector Functions*. We have begun the implementation of a system that accepts as input current situational characteristics (i.e., those that capture the problem domain, solution domain, and project domain), and the state of the known requirements, and outputs the set of applicable elicitation techniques. It uses a hybrid approach, combining an essence of both knowledge management and knowledge engineering. As its use becomes more widespread, the resultant sharing of best practices will assist in the increased success record for software development projects worldwide.

5. Relationship to other research domains

Knowledge management refers to an organization's "efforts to capture, store, and deploy knowledge using a combination of information technology and business practices" [44, p. 36] to help organizations compete more effectively [45]. Knowledge management systems (KMS) are "information systems designed specifically to facilitate the sharing and integration of knowledge" [46]. Results of a recent industry survey on KM show a wide diversity of perceptions ranging from the types of information that should be included in a KMS, to the

cultural and organizational issues with implementing a successful KM program, to the types of technology and systems associated with KM (e.g., data mining/warehousing, executive information systems, and expert and intelligent agent systems) [46]. The same diversity is apparent in KM research, which focuses on a wide range of organizational, cultural, and technical issues related to knowledge creation, storage/retrieval, and application [45].

The field of knowledge engineering (KE) is more specifically focused on the development of expert/knowledge-based systems and depends heavily on artificial intelligence research [47]. KE research and practice include recommended process models for the development of knowledge-based systems (e.g., [48]) and guidelines for knowledge acquisition (KA), representation, and coding of knowledge (e.g., [47, 49, 50, 51]).

While the overlap of the KM, KE and software development research domains seems obvious, surprisingly little cross-disciplinary research occurs. KE researchers continue to highlight areas where KE could improve KM efforts [44, 47]. Similarly requirements researchers highlight the need to explore knowledge acquisition techniques in addition to traditional requirements elicitation techniques [17, 50, 51]. We intend to take this advice in several areas. We have already depended heavily on the KE literature to guide our future research to develop a knowledge-based system to guide analysts in selecting elicitation techniques. Secondly, we will include both requirements elicitation and knowledge acquisition techniques in our list of available techniques. Finally, we will depend heavily on the KM literature to help us address the organizational, cultural, and change management issues that will surely arise as part of the implementation of our planned knowledge-based system.

6. Summary

This paper has introduced a new unified model of requirements elicitation. Although its significance will be determined only in the future, our hope is that this formal model of elicitation becomes the norm among researchers and practitioners. The model described herein highlights the critical knowledge required by, and defines the underlying basis of, an implementation of the elicitation technique selector function, which when complete, will enable:

- All (not just the most experienced) analysts will be able to select elicitation technique based on explicit knowledge, hitherto considered tacit,

- Project managers will have an improved appreciation for the critical role elicitation plays in overall project success,
- Analysts will be able to compare and contrast the assumptions and results achieved by the use of any elicitation technique,
- Analysts will be able to easily tailor existing methodologies for unique situations,
- Researchers will be forced to explicitly state the assumptions their elicitation methodologies are making about the situation at every step,
- Analysts will no longer be bound by pre-defined methodologies, but instead will be able to create new elicitation methodologies easily, by defining situational characteristics and then observing and recording the resultant instances of methodologies.

7. References

- [1] Gottesdeiner, E., *Requirements by Collaboration*, Addison-Wesley, 2002.
- [2] Standish Group, "The Chaos Report," www.standishgroup.com, 1995.
- [3] Hofmann, H., and F. Lehner, "Requirements Engineering as a Success Factor in Software Projects," *IEEE Software*, **18**, 4 (July/Aug 2001), pp. 58-66.
- [4] Gause, D., and G. Weinberg, *Are Your Lights On?*, Dorset House, 1990.
- [5] Jackson, M., *Problem Frames*, Addison-Wesley, 2001.
- [6] Wood, J., and D. Silver, *Joint Application Development*, Wiley, 1995.
- [7] Davis, A., "A Taxonomy for the Early Stages of the Software Development Life Cycle," *Journal of Systems and Software*, **8**, 4 (September 1988), pp. 297-311.
- [8] Davis, A., *Software Requirements: Objects, Functions and States*, Prentice Hall, 1993.
- [9] Goguen, J., and C. Linde, "Software Requirements Analysis and Specification in Europe: An Overview," *First International Symposium on Requirements Engineering*, IEEE Computer Society Press, 1993, pp. 152-164.
- [10] Kotonya, G., and I. Sommerville, *Requirements Engineering*, Wiley, 1998.
- [11] Lauesen, S., *Software Requirements: Styles and Techniques*, Addison-Wesley, 2002.
- [12] Leffingwell, D., and D. Widrig, *Managing Software Requirements*, Addison-Wesley, 2000.
- [13] Macaulay, L., *Requirements Engineering*, Springer, 1996.
- [14] Maiden, N., and G. Rugg, "ACRE: Selecting Methods for Requirements Acquisition," *Software Engineering Journal*, **11**, 5 (May, 1996), pp. 183-192.
- [15] Wiegers, K., *Software Requirements*, Microsoft Press, 1999.
- [16] Royce, W., "Managing the Development of Large Software Systems," *WESCON '70*, 1970; reprinted in *IEEE 9th*

International Conference on Software Engineering, IEEE Computer Society Press, 1987.

- [17] Siddiqi, J., and M. Shekaran, "Requirements Engineering: The Emerging Wisdom," *IEEE Software*, **13**, 2 (March 1996), pp. 15-19.
- [18] Hickey, A., "Integrated Scenario and Process Modeling Support for Collaborative Requirements Elicitation," University of Arizona Department of Management Information Systems PhD Dissertation, 1999.
- [19] Graham, I., *Requirements Engineering and Rapid Development*, Addison-Wesley, 1998.
- [20] Zave, P., "Classification of Research Efforts in Requirements Engineering," *ACM Computing Surveys*, **29**, 4 (April 1997), pp. 315-321.
- [21] Jarke, M., and K. Pohl, "Requirements Engineering in 2001: (Virtually) Managing a Changing Reality," *Software Engineering Journal*, **9**, 6 (June 1994), pp. 257-266.
- [22] Pohl, K., *Process-Centered Requirements Engineering*, Wiley, 1996.
- [23] Thayer, R., and M. Dorfman, *Standards, Guidelines, and Examples on System and Software Requirements Engineering*, IEEE Computer Society Press, 1994.
- [24] Loucopoulos, P., and V. Karakostas, *System Requirements Engineering*, McGraw Hill, 1995.
- [25] Playle, G., and C. Schroeder, "Software Requirements Elicitation: Problems, Tools, and Techniques," *Crosstalk*, **9**, 12 (December 1996), pp. 19-24.
- [26] Gaska, M., and D. Gause, "An Approach for Cross-Discipline Requirements Engineering Process Patterns," *Third International Conference on Requirements Engineering*, IEEE Computer Society, 1998, pp. 182-189.
- [27] Robertson, S., and J. Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999.
- [28] Hsia, P., et al., "Formal Approach to Scenario Analysis," *IEEE Software*, **11**, 2 (March 1994), pp. 33-41.
- [29] Holbrook, H., "A Scenario-Based Methodology for Conducting Requirements Elicitation," *ACM SIGSOFT Software Engineering Notes*, (January 1990), pp. 95-104.
- [30] Sutcliffe, A., and M. Ryan, "Experience with SCRAM, a Scenario Requirements Analysis Method," *Third International Conference on Requirements Engineering*, IEEE Computer Society Press, 1998, pp. 164-171.
- [31] Sommerville, I., et al., "Viewpoints for Requirements Elicitation: A Practical Approach," *Third International Conference on Requirements Engineering*, IEEE Computer Society Press, 1998, pp. 74-81.
- [32] Browne, G., and M. Rogich, "An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques," *Journal of Management Information Systems*, **17**, 4 (Spring 2001), pp. 223-249.
- [33] Gause, D., and G. Weinberg, *Exploring Requirements: Quality Before Design*, Dorset House, 1989.

[34] Pressman, R., *Adaptable Software Model, Preparing for Software Requirements Elicitation*, <http://www.rspa.com/-checklists/reqelicit.html>.

- [35] Maciaszek, L., *Requirements Analysis and System Design*, Addison-Wesley, 2001.
- [36] Dean, D., et al., "Enabling the Effective Involvement of Multiple Users: Methods and Tools for Collaborative Software Engineering," *Journal of Management Information Systems*, **14**, 3 (Winter 1997-98), pp. 179-222.
- [37] Davis, A., and A. Hickey, "Requirements Researchers: Do We Practice What We Preach," *Requirements Engineering Journal*, 2002.
- [38] Glass, R., "Searching for the Holy Grail of Software Engineering," *Communications of the ACM*, **45**, 5 (May 2002), pp. 15-16.
- [39] Yadav, S., et al., "Comparison of Analysis Techniques for Information Requirements Determination," *Communications of the ACM*, **31**, 9 (September 1988).
- [40] Alexander, L., and A. Davis, "Criteria for the Selection of a Software Process Model," *Fifteenth IEEE COMPSAC*, 1991, IEEE Computer Society Press, pp. 521-528.
- [41] Jorgensen, P., *The Use of MM-Paths in Constructive Software Development*, Arizona State University, Department of Computer Science PhD Thesis, 1985.
- [42] Juristo, N., and A. Moreno, *Basics of Software Engineering Experimentation*, Kluwer Academic, 2001.
- [43] Scharer, S., "Pinpointing Requirements," *Datamation*, April 1981, pp. 139-154.
- [44] Preece, A., et al., "Better Knowledge Management through Knowledge Engineering," *IEEE Intelligent Systems*, (Jan/Feb 2001), pp. 36-42.
- [45] Alavi, M., and D. Leidner, "Review: Knowledge Management and Knowledge Management Systems: Conceptual -Foundations and Research Issues," *MIS Quarterly*, **25**, 1 (March 2001), pp. 107-136.
- [46] Alavi, M., and D. Leidner, "Knowledge Management Systems: Issues, Challenges, and Benefits," *Communications of the Association for Information Systems*, **1**, 7 (Feb. 1999).
- [47] Liebowitz, J., *Knowledge Management: Learning from Knowledge Engineering*, CRC Press, 2001.
- [48] Rook, F., and J. Croghan, "The Knowledge Acquisition Activity Matrix: A Systems Engineering Conceptual Framework," *IEEE Transactions on Systems, Man, and Cybernetics*, **19**, 3, (May/June 1989), pp. 586-597.
- [49] Forsythe, D., and B. Buchanan, "Knowledge Acquisition for Expert Systems: Some Pitfalls and Suggestions," *IEEE Transactions on Systems, Man, and Cybernetics*, **19**, 3 (May/June 1989), pp. 435-442.
- [50] Byrd, T., et al., "A Synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques," *MIS Quarterly*, **16**, 1 (March 1992), pp. 117-138.
- [51] Shaw, M., and B. Gaines, "Requirements Acquisition," *Software Engineering Journal*, May 1996, pp. 149-165.