

Elicit: A Method for Eliciting Process Models*

Nazim H. Madhavji^{†‡} Dirk Hölting[†] WonKook Hong[†] Tilmann Bruckhaus[†]

[†]School of Computer Science, McGill University
3480 University Street, Montréal, Québec, Canada H3A 2A7
email: madhavji@opus.cs.mcgill.ca

[‡]Centre de recherche informatique de Montréal (CRIM)
1801 avenue McGill College, Bureau 800
Montréal, Québec, Canada H3A 2N4

Abstract

Eliciting process models from software projects is a first significant step towards process improvement. In this paper, we present a method called Elicit, for eliciting software process models from industrial software environments. What is significant about this method is that it has evolved from an intuitive state – the state that defines the immaturity of current elicitation methods – to a formally defined, repeatable, effective and quantified state. Over the last two years of its usage, the method has been used to elicit models from three industrial-scale processes: preliminary analysis, requirements engineering, and product planning and dependency management. The example given in the paper focuses on the requirements engineering process.

Keywords: Process model elicitation method; Method evolution; Process modelling; Software process models; Process improvement; Industrial-scale case studies.

1 Introduction

There is a general recognition, both in academia and the software industry, that improving software processes is important for building higher quality software products, on time, and within budget. Also, there is a school of thought that one way to improve processes is to first elicit descriptive models of the enacted processes, make appropriate changes to these models, and reflect these changes in the enacted processes [12].

In this paper, we focus on the first of these steps, that of eliciting process models from software projects. The specific question addressed here is: *How should one go about eliciting a process model?*

This is an important question to answer because the method used to elicit process models will have direct impact on the quality of the resultant models, their deliverability, and/or the cost incurred during elicitation. In turn, poor quality descriptive models would clearly not be a sound starting point for any subsequent changes, or for certification, training and other such activities. Similarly, delayed models, even of high quality, often result in the disruption in, and in the worst case cancellation of, follow-up process projects dependent on the timely delivery of elicited process models. The cost factor is also important to consider during elicitation because of finite resources.

Examining published literature, one finds descriptions of several efforts, especially in industry, on defining software processes, for example, at Texas Instruments [5], IBM [15], Software Engineering Institute [10], Paramax [4] and AT&T [16]. However, some of these efforts were directed at building prescriptive models without first eliciting descriptive models. These efforts were thus major re-engineering efforts, often disregarding current practices in the organisations. Many other organisations have described their processes, formally or informally. However, no recognised methods were used during elicitation. Consequently, it is difficult for others to reuse these approaches in future projects.

From all these experiences, one can conclude that currently there are no published methods on how to elicit process models from software projects. This is the gap this paper aims to fill.

In analysing the elicitation problem one should note that:

- The software development environment in which the process is enacted can be quite large, complex and in one of many different states.
- Eliciting a model of the process is more than just *modelling* the process using a process modelling

*This research was, in part, supported by NSERC, Canada, and IBM Canada Ltd., Toronto.

tool.

- The model of a given process has characteristics such as depth, scope, perspectives, etc.
- The elicited process information can be voluminous and thus needs to be managed adequately.

Thus, the problem of eliciting a process model is non-trivial.

Our proposed solution to address this problem is embodied by a meta-model of three key dimensions: *View*, *Method* and *Tool*. The *View* dimension consists of five different perspectives of a process: **Process Steps**, **Artifacts**, **Roles**, **Resources** and **Constraints**. A process model may be elicited from any of these perspectives, and can be viewed in terms of its static and dynamic properties. The *Method* and *Tool* dimensions consist of a set of steps and a tool-set, respectively, for eliciting and modelling a process. The method and tools can be applied to any perspective, thereby producing process models from different perspectives (e.g., activity view, artifact view, role view, etc.).

Our method, called Elicit, covers the aspects of characterising the environment, identifying process modelling goals, planning for elicitation, eliciting process information from numerous sources, synthesis of this information into a formal process model, validation, and analysis of the elicited model. The current tool-set used to support the Elicit method consists of a process elicitation tool (being developed at McGill), and Statemate¹ [6].

Following a pilot study, we have applied our approach in two industrial scale projects. In the first project we elicited a model of a requirements engineering process; in the second project we elicited a product planning and dependency management process. The examples given in this paper are taken from the first project. Detailed examples of the requirements engineering process model can be found in [7], and examples of the model of the product planning and dependency management process were used for a hypertext demonstration published on CD-ROM [7].²

Organisation of the Paper. The next section describes some background material. Section three describes the Elicit method. Section four reports about an example use of the method. Section five describes how the method has evolved over two years. Section six compares the Elicit method with related work. Finally, section seven concludes the paper and lists some observations.

¹Statemate is a trademark of i-Logix, Inc., Burlington, MA, USA.

²This demonstration can also be obtained via anonymous ftp from ftp.cs.mcgill.ca (132.206.51.2), /pub/labs/softeng/dirk.

2 Background

2.1 The State of Process Descriptions

Many organisations, especially large ones, have process descriptions, generally in a natural language form. Often, such descriptions can amount to twenty to thirty volumes, each for a particular process area in the software life-cycle. Below, we reproduce as a small example, an organisation-specific passage defining part of the step *Execute* which is a sub-step of the step *Validate* in the organisation's requirements engineering process.

"It is very important to ensure that the correct target market segment is being surveyed when identifying participants. In most cases, all affected organisations should be invited to the Validate sessions to hear unfiltered Customer input. ...

There are two inputs that will be gathered during the session:

- *Importance - This is a rating of how important the solution is to the Customer. ...*
- *Satisfaction - This is a rating of how satisfied the Customer is with the solution that is being validated. ...*

The importance and satisfaction ratings will be used to determine if the solution being validated should return to Analyze for further work, or if it is ready to be prioritized during the Prioritize subprocess."

No doubt, such informal process descriptions are useful for obtaining a general understanding of a process. However, they give no specific guidelines for engineering requirements in a given software project, aside the fact that they are often ambiguous. Such descriptions can thus lead to difficulties in, for example:

- performing the process properly,
- managing the process,
- training process performers or engineers,
- measuring and analysing the enacted process,
- changing the process,
- setting process standards across several projects,
- making precise comparisons with other processes,
- automating parts of the process, and
- inserting tools into the process.

Such difficulties must then affect the quality of the products of a process, and process costs.

2.2 Industrial Process Modelling Efforts

Still, there are an increasing number of industrial efforts aimed at building formally defined software process models. We describe three such examples.

Researchers at IBM conducted site studies at eight large-system programming development locations. In particular, the work carried out at the development sites was reviewed in order to understand how the software is being developed, propagate better alternatives across the sites, and help the sites in the evolution of

a consistently repeatable discipline for developing software [15]. The analysis of the findings resulted in a process definition in the ETVX notation.

At the Software Engineering Institute (SEI), researchers built descriptive models of processes used by the US Department of Defense [10]. The information regarding the process was elicited through interviews and through examining the applicable regulations. To organise and synthesise the gathered information, a structured narrative description of the major process steps was developed. This was then modelled using Statemate from functional, behavioural and organisational perspectives.

Another example is that of Texas Instruments' (TI) corporate-wide software improvement study to define and document a process representing a reasonable combination of generally accepted practices [5]. By examining smaller units of information and defining the process in terms of the basic information units or artifacts, TI was able to create a process definition which could be correlated to all of the examined process definitions.

Each of the described studies can be categorised in a different way. The SEI study dealt with building a descriptive project-specific process model from an enacted process. The TI and IBM studies both led to a prescriptive generic process model. The TI study derived the model from various other prescriptive generic process models, whereas the IBM studies investigated enacted processes.

In addition, the IBM study describes explicitly the meta process used to elicit the process model, but this process description is coarse grain. The SEI report does not contain explicit information about the meta process; it summarises experiences and concentrates mainly on the usage of appropriate modelling formalisms and tools. The TI study did not define a meta process, and in part this contributed to running down a number of blind alleys.

These experiences, in conjunction with the complexity of eliciting process models lead us to our investigation on a methodical approach for eliciting process models.

3 The Elicit Method

3.1 Overview

Our approach to eliciting process models is characterised by three key dimensions: *View*, *Method* and *Tool*.

The *View* dimension represents the different perspectives from which one can elicit or view a process. We

used five types of perspectives: **Process Steps**, **Artifacts**, **Roles**, **Resources**, and **Constraints**. For each of these, three properties were considered: **descriptive**, **static** and **dynamic**, thus giving many process views. Each view of the process (that is, *perspective* \times *property*) is described by a set of attributes (see Table 1). For example, the static view of artifacts is described by the following attributes: **input to process steps**, **output from process steps**, **input to external user**, **output from external source**, **owned by role**, **controlled by constraints** and **contains artifacts**. Each of these attributes represent specific process information when elicited.

The *Method* dimension is characterised by the Elicit method, described by the following key steps: (1) **Understand the Organisational Environment**, (2) **Define Objectives for eliciting a process model**, (3) **Plan the Elicitation Strategy**, (4) **Develop Process Models**, (5) **Validate Process Models**, (6) **Analyse Process Models**, (7) **Post-Analysis of the usage of the method** and (8) **Packaging of the experience gained**.

The *Tool* dimension is characterised by the two tools we have selected: Elicit and Statemate.

The Elicit tool, which is being developed at McGill, has a built-in knowledge of software process perspectives and properties. Its design is aimed at eliciting process information, at the front-end of the elicitation process. A process view is modelled as a hierarchy, where each element in the hierarchy is described by a set of attributes in the *View* dimension. Each attribute can be filled with actual process values from given projects and can be linked to an ASCII file containing additional descriptions. To guide the elicitation process, each attribute is associated with a specific question that is displayed in the command line while editing the attribute value.

Statemate is a commercially available system design and analysis tool. It provides a set of features originally designed to support the development of reactive systems such as avionics systems, communication and control systems, and interactive hardware or software [6]. Statemate provides editors to model the *functional*, *behavioural* and *structural* aspects of a system. In addition, Statemate provides a set of tools for analysing models with respect to completeness and correctness, simulation of statecharts and their associated activity charts, queries on textual elements, and generation of reports and plots. In his applications of Statemate, Kellner [8, 10] has demonstrated a way of using Statemate for modelling processes.

In our case studies, guided by the Elicit method, we used the Elicit tool to gather and structure process information from the described five perspectives. This information was then used as input to Statemate

Perspectives	descriptive property	static property	dynamic property
Process Steps	identifier achieves goals has purposes specified-by procedure sends messages receives messages characterised by measurements	input artifacts output artifacts performed by roles owned by role needs resources controlled by constraints contains process steps	requires entry conditions fulfills exit conditions assumes states
Artifacts	identifier has purposes stored in formats is of artifact-type characterised by artifact-description	input to process steps output from process steps input to external user output from external source owned by role controlled by constraints contains artifacts	assumes states
Roles	identifier has permissions has obligations requires qualifications	performs process steps owns process steps owns artifacts manages resources contains roles	
Resources	identifier is of resource-type characterised by resource-description	allocated to process steps managed by role contains resources	assumes states
Constraints	identifier is of constraint-type characterised by constraint-description	controls process steps controls artifacts contains constraints	assumes states

Table 1: Process modelling attributes

to perform complementary activities of modelling the static and dynamic aspects of the process.

This paper focuses on the *Method* dimension of our elicitation approach. The *View* and *Tool* dimensions are described in detail in [7], and experiments with the Elicit method are described in [13].

3.2 Steps of the Elicit Method

A formal description of the Elicit method is presented in Figure 1. The figure, an activity chart created using Statemate, depicts the steps and dataflow amongst these steps, where boxes represent process steps, solid arrows represent dataflow, and dashed boxes represent external agents.

Step 1: Understand the Organisational Environment

The main objective in this step is to obtain a basis for a realistic definition of objectives upon which the rest of the elicitation process can be based. The key input to this step is the **Organisational Knowledge** and the key output is **Documented Context** considered useful for the elicitation task.

Example issues of interest in this step are:

- *Organisational issues* identifying the structure of the development environment, size and structure of developing teams; roles played by developers and process engineers; the type and quality of process information various people can provide; the customer who receives and validates the elicited models; etc.
- *Process issues* identifying the processes to be examined; type and state of the current processes (in-

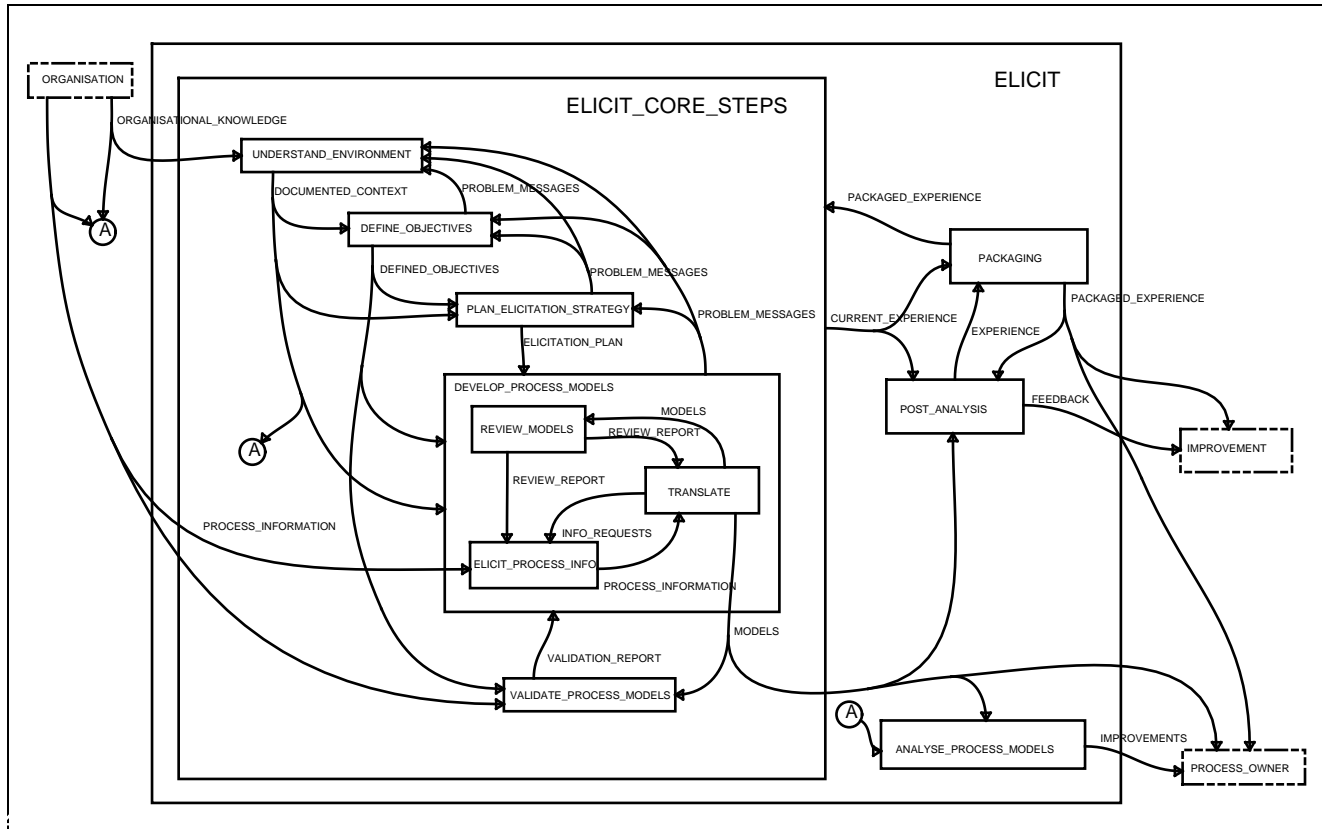


Figure 1: The Elicit method

tuitive, informally described, formally modelled); etc.

- *Project issues* identifying the state of the software projects (embryonic, early post-release, nearing the end of its life-span, etc.); sizes and types of the software products; delivery cycle-times; etc.

Such issues are critical for understanding the context, priority, choices, alternatives, etc., within the environment. They form a solid basis for determining or negotiating the objectives of eliciting process models.

Step 2: Define Objectives

In this step, one should define a set of objectives for the elicitation task so that later on they may be used to measure the quality of the elicited process models and that of the elicitation process. Using the Documented Context, the output of this step is a set of Defined Objectives.

There are two main types of objectives to be considered:

- *Model-oriented objectives*: the software process components to be elicited; granularity of the software process models to be built; and the degree of consistency, completeness, accuracy and clarity that should be achieved in the software process models that are built.
- *Project-oriented objectives*: acceptable elicitation cycle-time and cost; and the available human and computing resources.

These objectives are critical in directing the rest of the process of elicitation. Fuzzy or ill-defined objectives often result in backtracking to the earlier steps at substantial cost.

Step 3: Plan the Elicitation Strategy

Based on the defined objectives and the documented context, the aim in this step is to develop an elicitation plan and to allocate appropriate resources in order to achieve quality models within budget and time.

Some of the key issues in planning for elicitation are identification of the detailed scope of the elicitation

task; identification of risks and deliverables; selection of elicitation methods and tools; scheduling of interviews, demonstrations, discussions, reviews and clarification sessions; and allocation of computing and human resources.

Step 4: Develop Process Models

Here, we elicit process models according to the Documented Context, Defined Objectives, Elicitation Plan, and the Validation Report from step 5. The exit criteria for this step include fulfillment of the objectives, consistency of the process models with respect to static-semantic aspects, and adequate and correct coverage of the available process information. However, backtracking to step 1 may be necessary in order to obtain an improved understanding of the organisational environment, to step 2 in order to redefine the objectives, or to step 3 in order to re-plan elicitation strategies.

This step is decomposed into three substeps: Elicit Process Information, Translate and Review Models. All the inputs to the parent node should also be available to these substeps.

In the step Elicit Process Information, we draw out process information, specified by the attributes in Table 1, from various sources. Additional inputs to this step include Process Information from the organisational environment, the Review Report from the step Review Process Models and Information Requests from Translate.

Examples tasks in this step include:

- reading process and other documents, and preparing questions for eliciting process information,
- identifying process details from documents, discussions and interviews,
- conducting discussions with practitioners, process experts and managers to clarify ambiguous or misunderstood issues, and
- improving process descriptions with respect to connectivity, completeness, accuracy, granularity, etc.

The Process Information drawn out is then translated into a structured and well-defined representation supported by the Elicit tool and Statemate. However, one should ensure the congruence of the models developed using Statemate with the process information in the Elicit tool and, in turn, with the source information. Thus, the Review Report may cause backtracking to the steps Elicit Process Information or Translate. If process experts are participating in this step, it can be executed in parallel with step Elicit Process Information.

Step 5: Validate Process Models

Once the process models have been built, they should be validated and approved formally by the recipient. This includes checks such as whether the model accurately represents what is documented as Process Information or whether it reflects the enacted process in an appropriate way. Other matters such as complexity of the models developed and perspectives addressed are also examined. The Defined Objectives and the process models represented in the elicitation and analysis tools are also used in this validation process. The Validation Report specifies the discrepancies found, which are fed back to Develop Process Models. This step completes the core aspects of the Elicit method.

Step 6: Analyse Process Models

The elicited process model, even if it reflects the source information accurately, may not necessarily be desirable in all respects because it may be falling short of the organisational or desired standard. Thus in this step, we take the opportunity to analyse the static and dynamic aspects of the process model in order to document improvement opportunities. The inputs to this step are Process Information, Organisational Knowledge and Documented Context and the process models developed in step 4. The output is a list of improvement opportunities that is delivered to the Process Owner for possible changes.

Step 7: Post-Analysis

While the elicitation activity itself is over one may analyse the elicitation task to improve the Elicit method. Several points need to be considered during post-analysis: (i) the *level of abstraction* of post-analysis, (ii) the *aspects* of post-analysis, and (iii) the *properties* of post-analysis.

- The *level of abstraction* of post-analysis can be global, black-box, or white-box. In global post-analysis, the core steps of the method are treated as a single block, with links to the post-analysis, model analysis and packaging steps. In black-box post-analysis, we examine the organisation of the steps of the method and their links without examining the details of each step and link. Finally, in white-box analysis, each step and link is examined in depth.
- The *aspects* of post-analysis are functional, data and behavioural aspects, covering the functionality of the method, the data produced and consumed within the method, and the dynamics of elicitation.

- The *properties* of post-analysis are fine-grain details concerning a given aspect at a given level of abstraction. For example, during white-box analysis of the functional aspect of the Elicit method, one may examine the details of the steps, the entry and exit criteria of each step, and the roles and tools involved in the elicitation.

It can be seen that such post-analysis can lead to substantial feedback for making improvements to the elicitation method.

Step 8: Packaging

Finally, one may package the experience gained during elicitation and post-analysis, so as to simplify future applications of the Elicit method. Example aspects of packaging to be considered are:

- the *type of objects* that can be packaged (such as the process models, documents produced during elicitation, the dynamics of the method, or the experience gained in eliciting process models),
- the *type of users* of the packaged information,
- the *usability* of the packaged information for other elicitation tasks, and
- the *cost/benefits ratio* of packaging.

In summary then, one can see that eliciting process models from actual projects is quite involved. Specifically, there are some key steps that should be followed in order to elicit and improve process models. The use of structuring, modelling and analysing tools is also critical because it simplifies the operationalisation of the method. But it is important to note that tools alone do not provide an appropriate context for elicitation.

4 Example Use of the Method

We have applied the Elicit method in one pilot study, and after significant improvements, in two industrial scale projects. In this section, we describe results and our experience from the first of the two projects, which dealt with a requirements engineering process. Because of the lack of space here, we describe only the enactment of technical steps of the Elicit method: **Plan the Elicitation Strategy**, **Elicit Process Information**, **Translate**, **Validate** and **Analyse**. Although these steps consumed 82% of the 126 person hours spent on the project, the front-end activities of understanding the environment and identifying specific objectives are among the most

important ones in the elicitation exercise because they define the orientation of the technical activities to follow.

4.1 Planning for Elicitation

The strategy for this project was to elicit an initial process model from natural language process documents and then to refine it in walkthroughs with, and interviewing, process experts. In planning elicitation, we decomposed the step **Elicit Process Information** into the following five sub-steps:

1. Read the selected process documents and clarify any misunderstandings with the help of process experts.
2. Identify the structure of the process documents and the process perspectives used to organise the documents.
3. Determine the suitable level of details to be aimed at during elicitation, and get this approved.
4. Mark or gather all information which can be captured in the process attribute structure supported by the Elicit tool.
5. Enter the identified information into the Elicit tool.

This strategy was developed and applied successfully in the earlier, pilot study.

4.2 Elicit Process Information

The elicitation was then carried out according to the substeps identified in the elicitation strategy. The structure of the requirements process documents were based on the perspective **Process Steps**. On the top level, the requirements process is decomposed into five steps: *Capture*, *Analyze*, *Validate*, *Prioritize*, and *Commit*. The process documents also described related information, such as: responsibilities, measurements, methods, techniques and tools, implementation of the process, and linkages to other processes.

In order to capture process information, one can identify all nouns and verbs, where nouns represent candidate objects or entities and verbs represent candidate operations on the objects or actions. In our case, however, a noun can denote an artifact, a resource, a role or a constraint. In addition, process attributes such as **Goal** and **Purpose** are at a high level, implying that their values can be complete sentences. This necessitates a proper categorisation of process information.

Another important factor is the quality of the documents to be analysed. If these are well structured and based on an entity-action-model [9, sections 6.1-6.3] or an attribute structure, the capturing method is more likely to succeed. The requirements documents we analysed did not fit exactly into this category. From the first reading it became obvious that the text described mainly the process steps and artifacts, and also described substantial background and environmental information. Because the distinction between process information and background information was not always clear from the structure of the text, the elicitation needed considerable domain understanding and human judgement. We thus concluded that a mechanical analysis on a word or phrase level would not be successful in this case.

The elicitation was thus carried out on a paragraph level: read a paragraph and try to understand it; identify the nouns or verbs which refer to a process perspective; and identify sentences or phrases corresponding to the attributes specifying the process perspectives and relationships.

As a concrete example, we illustrate how this strategy was used to elicit the static, structural and relational properties of a process step. The following text is part of the description of the step *Review*, which is a substep of the top-level process step *Analyze*:

“The product representative selects new requirements from the Requirements Management Tool and adds keywords that describe the contents of the requirement. These keywords will be used to group requirements in later steps.”

The nouns and verbs that refer to process perspectives belonging to this step are: *Product Representative*, *select*, *Requirement*, *Requirements Management Tool*, *add*, *Keywords*. Basically, the text describes a set of activities. Thus, by analysing verb-noun relationships in the document, we were able to build components of the process model.

For example, *Select* refers to the artifact *Requirement*. Thus, we have a process step *Select Requirements* which is carried out by the role *Product Representative* and makes use of the resource *Requirements Management Tool*. The verb *Add* refers to the artifact *Keywords*. Thus, we have a process step *Add Keywords* which is carried out also by the role *Product Representative*. The phrases *“describe the contents of the requirement”* and *“used to group requirements”* identify the purposes of the artifact *Keywords*. It is not quite clear from the text whether the keywords are added to the Requirements Management Tool or to the requirements. As it seems to make more sense to link the keywords to the requirements, the artifact *Keywords* is modelled as a component of the artifact *Requirement*.

Such decisions are, of course, subject to reviews by the process owner.

Similarly, we were able to identify values for all attributes listed in Table 1, such as conditions under which steps are executed, producers and consumers of artifacts, etc. However, one can run into difficulties because references within a sentence can be imprecise, or incomplete; facts can be expressed implicitly; verbs and nouns can be overloaded; and other such problems. In our projects, it was possible to resolve most of these problems from the context or with the help of the process owner.

The identified information was then entered in the Elicit tool, which represents it hierarchically. During this step, all redundancies and inconsistencies in the process document became obvious because, for example, a process step can be described repeatedly in several sections and figures of a process document; whereas in the Elicit tool, there is only one well defined position to store this particular information.

This elicitation step, which required 24 hours or 19% of the total time, resulted in initial process models based on their structural properties.

4.3 Translating into Graphical Models

The process model structure held in the Elicit tool was translated to Statemate without much difficulties.³ However, in order to represent dataflow amongst the activities, it was necessary to have a second pass through the process documents. This activity was simplified by identifying data sources and destinations with the help of traversal facilities of the Elicit tool.

In contrast to this static information, the description of process dynamics was meagre in the documents. Thus, the reasoning about more realistic process dynamics was deferred to the analysis step. The modelling of artifact-state changes, on the other hand, was possible but not without another pass through the process documents.

The translation of the process model took 30h 40' (23% of the total case study effort). In this case study, the translation activity required several re-elicitations, i.e., additional passes through the process documents. These re-elicitations required a third of the time spent in the translation phase.

³ Because of the lack of space here, we are not able to include sample graphical process models. However they are described in [7].

4.4 Model Validation

Once the graphical process models had been built, it was important to get them validated by project personnel in order to gain acceptance. The key issues that were covered in the validation of the requirements process model were completeness, correctness and accuracy of the model. For *completeness* validation, the key question asked was whether the entities and relationships specified in the formal models were complete with respect to source information. Similarly for *correctness and accuracy* of the model, the key question asked was whether the model entities, their decomposition, their relationships, their dynamics, artifact producer/consumer dependencies and artifact decomposition among substeps reflected what was meant in the source information, and whether the technical decisions made to resolve obvious inconsistencies were acceptable. Such sessions were carried out in several iterations, and feedback was used to improve the formal models. Twenty-nine hours (23% of the case study) were spent to conduct the validation sessions and to correct the models accordingly.

4.5 Analysing the Resultant Models

After elicitation was complete, we analysed the static and dynamic aspects of the model in order to assess the quality of the model and identify improvement opportunities. Our experience suggests that this activity is best carried out in the elicitation project rather than a later, change project because of the gained insight into the elicited models and project momentum.

In any case, the dynamic aspects of a process model can be analysed using simulation capabilities of process modelling tools such as Statemate. The simulation tool animates visually the behaviour of a process model and keeps a trace file. One can perform several types of analyses, such as reachability, deadlock, race condition and non-determinism. In addition, it is possible to perform quantitative simulation to aid management tasks such as planning and control [10].

The process information we had access to contained little information on the dynamics of the requirements process. Thus, while we could run the elicited formal models in a simulation mode, there was a lack of adequate project data to carry out realistic analysis. Consequently, such analysis is not treated in this paper.

Static analysis was carried out by applying the Goal/Question/Metric (GQM) paradigm [2]. In essence, this meant that the goals of the analysis were stated, and specific questions related to the goals and, in turn, specific metrics related to each question were

identified. The model was then assessed based on the data gathered by using the metrics. This analysis required 20h30' or 16% of the case study effort. There were three main aspects of static analysis that we examined in the elicited process models: inconsistencies, redundancies and incompleteness, giving a wide coverage for analysis.

Inconsistencies. Inconsistencies, can be contained in both, the input sources (e.g., process documents and gathered information) and the elicited process models. There are two types of inconsistencies: first order and second order. The former type are those that are contained in the input sources and are generally finegrain (e.g., conflicting values of the same process attribute). Such inconsistencies usually surface during elicitation. The second order inconsistencies are violations of consistency constraints which can be detected only by comparing parts of the complete model (e.g., "All work products available to a parent node are also available to each child node").

In the requirements process documents and the gathered information, numerous different cases of inconsistencies were identified. Examples are conflicting information about the decomposition of process steps, and about the process attributes **Goal**, **Procedure**, **Role** and **Exit Criteria**. The overall structure of the process and the decomposition of the top-level process steps are described repeatedly with substantial variability. These inconsistencies were considered significant and were resolved successfully, in conjunction with the process owner.

In contrast to this, inconsistencies in the formalised process model can occur only in complementary attribute pairs such as **produces/is-produced-by**. They are usually due to an erroneous elicitation and can be resolved easily.

Redundancies. Redundancies contain the same or supplementary information which should be captured in a single value of a process attribute. In the requirements process documents, there were numerous cases of redundant information, a majority of which related to the attributes **Process Steps** and **Goal**. It is worthwhile noting that in the Elicit tool, every piece of information can be stored only in one well-defined location, and therefore, redundancies are impossible.

Incompleteness. The question of incompleteness is targeted only at the formal process model because completeness of a natural language text or the gathered information is difficult to quantify. In the Elicit tool,

a description of a process model entity is considered complete if all the process attributes have been used. However, because in practice it may not be appropriate to use all the process attributes in every use of the process model, three different metrics targeting at different levels of completeness were designed: (1) number of non-blank values per attribute to measure the utilisation of each attribute, (2) number and list of process model entities which have attributes with blank values only, and (3) number and list of process model entities which are not described by a minimum set of attributes.

The idea behind the third metric is that the each model entity should be described by at least some basic information. For example, each process step should be specified by at least the following attributes: **Goal** or **Purpose** to explain why the step is needed; **Procedure** or a decomposition into further **Process Steps** to explain what is to be done in this step; and **Artifact Input** and **Artifact Output** because steps which do not process anything are generally useless.

Tables 2 and 3 show the statistics for process steps and artifacts with respect to missing attribute values. The attributes **Artifact Input** and **Artifact Output** were used only when a dataflow was explicitly mentioned in the documents because no consistency rule referring to the availability of data (such as “All work products available to a parent node are also available to each child node.”) was applied by the process author. The missing dataflows were thus considered as a major deficiency of the model; this was resolved subsequently.

Group	Missing Attribute Values	# of Steps (%)
1	Goal & Purpose	19 (28%)
2	Procedure & Process Steps	11 (16%)
3	Artifact Input	36 (53%)
4	Artifact Output	26 (38%)
	all permutations of 1 - 4 above	43 (63%)

Table 2: Missing Attribute Values in Process Steps

Group	Missing Attribute Values	# of Steps (%)
1	Purpose & Description & Artifacts	27 (44%)
2	Producer & External Source	8 (14%)
3	Consumer & External User	7 (11%)
	all permutations of 1 - 3 above	29 (47%)

Table 3: Missing Attribute Values in Artifacts

To summarise, this analysis step is needed for later improvement even if the elicited models are accurate. Also, it should be clear that this type of analysis would be difficult without a formal attribute structure (see

Table 1), and finally, we have found that process descriptions built using ad hoc and informal methods are not of as high a quality as that built using formally defined methods such as Elicit.

5 Evolution of the Method

The Elicit method has evolved over the last two years through three iterative cycles and is undergoing further refinements. In the first cycle, a pilot study, we elicited a preliminary analysis process from an organisation. During this elicitation, we used our modelling experience and tools such as the Elicit tool and Statemate in an intuitive way. Although, we knew how to use these tools and were aware of numerous other process modelling/enacting systems, including Kellner’s use of Statemate for the *design* and *analysis* [8, 10] of process models, we had no repeatable or defined *process* for eliciting software process models. Consequently, the elicitation process broke down frequently and there was significant backtracking. This experience suggested that knowing how to use specific tools alone is sometimes not sufficient for accomplishing a task. Knowing the process in which the tools are embedded is also important.

The post-analysis of the pilot elicitation resulted in the first version of the Elicit method. While we now had an understanding of the key steps of the method, we still lacked (a) a formal definition of the method, (b) knowledge about the details of the method, and (c) concrete insight into the dynamics of the method (such as, iterations, concurrency, or relative effort in different steps).

In the second cycle, we took the first version of the method and enacted it *with care*, implying (a) that we could follow different rules if the method did not help, (b) that we needed to identify detailed activities in the elicitation process, and (c) that we intended to measure the elicitation process. This elicitation exercise was performed on a requirements engineering process from an industrial organisation.

To improve the Elicit method, we analysed the first version of the method in conjunction with the experience gained from the second elicitation effort, and made many changes such as addition, modification and detailing of activities and dataflow, resulting in an improved version of the Elicit method. Also, we measured the elicitation process to build a model of the dynamics of this process [13]. This analysis and improvement process followed Basili’s experimental software engineering paradigms: the *Quality Improvement Paradigm* (QIP) [1] and the *Goal/Question/Metric* (GQM) paradigm [2].

The improved second version of the method was then input into the third improvement cycle, where it was applied to a product planning and dependency management process from the same organisation. The analysis of this elicitation led to a better understanding of the refinements made after the second elicitation and led to a new assumption about the dynamics of the Elicit method [13].

It is important to note that in all these evolutionary cycles, there were two primary objectives. One, to elicit formal models of software processes. This was the main objective of the sponsoring organisation, and only secondary objective of our research. Two, to improve the Elicit method with respect to its completeness, correctness, fidelity and fitness. This improvement effort was our main objective, and only secondary objective of the sponsoring organisation.

In summary, the first version of the Elicit method was coarse-grain, waterfall-like and described in natural language. It lacked important details in the steps where technical work was done and it was not clear on the activities for reviewing, validating and analysing process models. Thus, the method was difficult to enact and improvisations became necessary. In the second version, most of the described problems were fixed. The method description was formalised and the description of each step, dataflow and artifact was refined. In the third version, the technical steps of the method were modified to allow a more iterative participation of the process owner and other process experts. We feel that we now have a well-defined elicitation method that is not difficult to follow and is repeatable in similar projects.

6 Comparisons

As described in the background section, many organisations have elicited process models or described their software processes. However, they have generally used ad hoc, undocumented methods, making it difficult to assess them or to compare the Elicit method against them. Nevertheless, it can be said that the evolutionary cycles have made the Elicit method explicit and quite stable, and thus, reusable by others. This is perhaps the biggest advantage of Elicit over other elicitation methods.

In so far as the quality of the elicited process models is concerned, we have found consistently that the process descriptions resultant from the use of ad hoc methods are deficient in many respects (for example, inconsistent information, incomplete models, and conflicting interfaces) and that the Elicit method has helped identify such quality problems in process documents.

One can relate the Elicit method to the various approaches to process modelling based on formalisms such as APPL/A, MVP-L, HFSP, rules and others (see [11] for pointers to these formalisms). The modelling formalisms together with the tools supporting them impose certain constraints on how to represent process models. In contrast, the Elicit method does not propose any particular representation formalism because it is a *life-cycle* model for eliciting process models. It can thus accommodate the use of any formalism for process model representation.

Currently, there are two tools being used in the Elicit method. One is a process information elicitation tool (also called Elicit). This tool uses a process attribute structure (see Table 1) to represent the elicited process information. This information is then translated into the graphical formalisms supported by the second tool we are currently using, Statemate [6], so that process models can be analysed and simulated. Our use of Statemate mirrors that of Kellner's [8, 10] except that we have modelled two aspects (static and dynamic) of several process perspectives (e.g., process steps and artifacts) in an integrated way; whereas Kellner has modelled process steps [10] and artifacts [8] in an un-integrated manner. Both the Elicit tool and Statemate (and their underlying formalisms), however, may be replaced by other appropriate tools provided that they support the elicitation and analysis/simulation functions.

7 Observations and Conclusions

Eliciting process models from enacted processes is a first significant step towards process improvement. This paper describes a method, called Elicit, for eliciting process models. The method has undergone several improvement cycles during its use on industrial-scale software processes, and is effective.

From our experience with the application of the Elicit method, we note the following observations:

- While certainly useful in managing voluminous process information, tools alone are not adequate for eliciting quality software process models within budget and time; the method is important too.
- The larger or riskier the elicitation project the greater the need for front-end activities such as understanding the environment, defining objectives and planning for elicitation.
- The different perspectives of software processes can help tackle the magnitude and complexity of process models in a piece-meal fashion.

- Both *domain* knowledge and *process engineering* knowledge is critical in the elicitation, analysis and improvement of formal process models.

Our work on process models extends beyond elicitation. In particular, we are carrying out experimental studies in the area of tool insertion [3], process generalisation and process measurements [14], based on formally defined process models. From this and other work, we have no doubt that formally defined process models provide an excellent basis for investigating numerous process issues. However, what is clearly important in all this is to have a quality method that can be used to elicit high-quality software process models.

Acknowledgements

We are indebted to many people who have contributed to the work described in this paper. In particular, we would like to thank Jack Dawson and Steve Hodson of IBM Canada Ltd., Toronto, for providing us with process information and sharing their process experience; John Botsford for his support by forming a bridge between academia and practice; and Jacob Slonim for facilitating a superb experimental research environment for us. Botsford and Slonim are with the Centre for Advanced Studies (CAS) at IBM Canada Ltd., Toronto. Finally, we would like to thank the anonymous referees and Larry Votta of AT&T Bell Labs for giving us excellent feedback which has helped improve this paper considerably.

References

- [1] V.R. Basili. The Experimental Paradigm in Software Engineering. In *Proc. Int. Workshop on Experimental Software Engineering Issues*, pages 3–12, Schloß Dagstuhl, Wadern, Germany, September 1992. Springer Verlag, Berlin, LNCS 706.
- [2] V.R. Basili and D.M. Weiss. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, 10(6):728–738, November 1984.
- [3] T. Bruckhaus. The Impact of Inserting a Tool into a Software Process. In *Proceedings of the 1993 CAS Conference*, pages 250–264, Toronto, Ontario, Canada, October 1993. IBM Canada Ltd. and The National Research Council of Canada.
- [4] D.W. Drew. Developing Formal Software Process Definitions. In *Proc. Conference on Software Maintenance 1993*, pages 12–20, Montréal, Québec, Canada, September 1993. IEEE Computer Society Press.
- [5] D.J. Frailey, R.R. Bate, J. Crowley, and S. Hills. Modeling Information in a Software Process. In *Proc. 1st Int. Conf. on the Software Process*, pages 60–67, Redondo Beach, CA, USA, October 1991. IEEE Computer Society Press.
- [6] D. Harel et al. STATEMATE: A Working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–414, April 1990.
- [7] D. Höltje, N.H. Madhavji, T. Bruckhaus, and W.-K. Hong. Eliciting Formal Models of Software Engineering Processes. In *Proc. of the 1994 CAS Conference (CASCON'94)*, Toronto, Ontario, Canada, October 1994. IBM Canada Ltd. and The National Research Council of Canada.
- [8] W.S. Humphrey and M.I. Kellner. Software Process Modeling: Principles of Entity Process Models. In *Proc. 11th Int. Conf. on Soft. Eng.*, pages 331–342, Pittsburgh, PA, USA, May 1989. IEEE Computer Society Press.
- [9] M. Jackson. *System Development*. Prentice Hall, Englewood Cliffs, NJ., 1983.
- [10] M.I. Kellner. Software Process Modeling Support for Management Planning and Control. In *Proc. 1st Int. Conf. on the Software Process*, pages 8–28, Redondo Beach, CA, USA, October 1991. IEEE Computer Society Press.
- [11] M.I. Kellner and H.D. Rombach. Session Summary: Comparisons of Software Process Descriptions. In *Proc. 6th Int. Software Process Workshop*, Hakodate, Japan, October 1990.
- [12] N.H. Madhavji. The Process Cycle. *IEEE/BCS Software Engineering Journal*, 6(5):234–242, September 1991.
- [13] N.H. Madhavji, D. Höltje, W.-K. Hong, and T. Bruckhaus. Elicit: An Empirically Improved Method for Eliciting Process Models. Technical Report TR-74.141, IBM Canada Ltd., May 1994.
- [14] N. Moukheiber, K. El Emam, and N.H. Madhavji. A Rigorous Method for Defining Metrics. Technical Report SE-94.5, School of Computer Science, McGill University, Montréal, 1994.
- [15] R.A. Radice, J.T. Harding, P.E. Munnis, and R.W. Phillips. A Programming Process Study. *IBM System Journal*, 24(2):91–101, 1985.
- [16] H.T. Yeh. Re-Engineering a Software Development Process for Fast Delivery - Approach & Experiences. In *Proc. 1st Int. Conf. on the Software Process*, pages 106–112, Redondo Beach, CA, USA, October 1991. IEEE Computer Society Press.