2012 IEEE 36th International Conference on Computer Software and Applications

Development of Situational Requirements Engineering Processes: A Process Factory Approach

Omid Jafarinezhad, Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
jafarinezhad@ce.sharif.edu, ramsin@sharif.edu

Abstract— **The Software Product Line (SPL) approach is a paradigm for systematic reuse of software products, and a Software Factory is a SPL aimed at the industrialization of software development. Based on the notion that a software/RE process can be developed via an engineering process (much akin to engineering other types of software), this research aims to provide a feature-based RE process factory to develop RE processes based on the characteristics of the project at hand (project situation). In our approach, the project situation is modeled as the problem domain through using the i* modeling language (resulting in a situation model). A feature model can encapsulate all the features in an SPL; therefore, the abundant riches of the RE field – results of decades of research – have been explored for extracting the variations and commonalities among existing RE processes, the results of which are represented in the form of a feature model, considered as a model of the solution domain. In order to demonstrate the validity of the proposed feature model, it has been compared against RE-related activities found in prominent software development methodologies. A mapping for translating the situation model to the RE process feature model is proposed with the specific aim of promoting traceability and rationality in the selection of RE process features. The efficacy of the approach is demonstrated through an RE process development example.**

Keywordst; Situational Requirements Engineering, Software Process Reuse, Process Factory, Software Product Line.

I. INTRODUCTION

As an integral part of the discipline of Software Engineering (SE), a Software Development Methodology (SDM) or Methodology refers to the framework for applying SE practices. It may also be described as consisting of two main parts: 'A set of modeling conventions comprising a Modeling Language (syntax and semantics); and a Process, which specifies the development activities and their order, provides guidance for monitoring the activities, and specifies what artifacts should be developed using the Modeling Language' [1]. As a consequence of the famous observation that 'software processes are software too' [2], devising an engineering approach to construct/adapt a methodology for specific projects has become a subject of interest for software developers.

Methodology Engineering (ME) emerged in response to the need for applying an engineering approach to constructing a SDM. ME later came to be known as Method Engineering, defined as 'The engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems' [3]. Situational Method Engineering (SME) is a well-known subfield of ME, specifically aimed at constructing/adapting a methodology for a specific project situation. There exist several synonyms for it in literature; examples include Situated Method Engineering, Method Adaptation, and Method Tailoring.

The prevalent belief that no one methodology fits all project situations has been widely accepted in recent years, especially considering the recent experiences on replacing heavyweight methodologies by SME frameworks – for instance, the RUP and OPEN methodologies have now been replaced by Rational Method Composer (RMC) and OPEN Process Framework (OPF), respectively. On the other hand, the tendency to use configurable and flexible processes, as observed in recent methodologies such as Catalysis and Adaptive Software Development (ASD), has resulted in a tendency to use SME [4]. The idea of SME is not particularly novel, however: The 'requirements determination strategies' approach [5] is an early example of a methodology that characterizes a specific project on the basis of certain criteria, and thus provides the means for situational requirements elicitation.

The pivotal role of RE in SE (and ME) is well established. Poorly defined requirements are still one of the main causes of software problems [6], so much so that the term 'requirements problem' has become a cliché, and improving the RE process has become essential [6, 7]. Hence, the notion that RE processes are situation dependent [8, 9] has emerged to address the requirements problem.

The idea of manufacturing software products from reusable components has been around for decades. The Software Product Line (SPL) [10] approach is a paradigm for systematic reuse of software products, and a Software Factory is a SPL for the industrialization of software development. Software Factories are actually the logical next step in the continuing evolution of software development methods and practices.

The major contribution of this paper is a novel approach for developing (engineering) situational RE processes which is based on the notion that SDMs (including RE processes) can be developed via a software engineering process [4]. Our proposed approach focuses on the idea of developing a RE process factory to produce RE processes for specific project situations, promote large-scale reuse, and reduce

0730-3157/12 $26.00 © 2012 IEEE
DOI 10.1109/COMPSAC.2012.39

development costs. To this aim, a feature-based SPL approach is proposed. This paper also shows how RE and SME research efforts can benefit from each other. RE processes can be engineered through using SME; on the other hand, SME can be improved by RE guidelines and best practices. The two disciplines share a common interest, as they both aim at promoting the quality of software development.

The rest of this paper is structured as follows: Section II provides an overview of related research; Section III introduces the proposed approach and provides detailed descriptions for its phases; Section IV contains an illustrative example; and Section V presents the conclusions and suggests ways for furthering this research.

II. RELATED RESERACH

Various approaches have been proposed for alleviating the problems encountered in improving and developing a RE process for a specific project or organization. The related research can be classified into the following four categories: 1) RE Process improvement models (e.g., [7, 8, 11]), which assess the current situation and introduce guidelines and techniques for improving RE processes within an organization to meet new goals and objectives; 2) Requirements determination approaches (e.g., [12, 13, 14]), which suggest project-specific RE processes or techniques to meet the needs of a target situation; 3) Empirical studies (e.g., [15, 16, 17]), which provide rich guidelines on the applicability of certain techniques/practices in specific projects; and 4) SME approaches (e.g., [11, 18, 19]), which provide disciplines aimed at constructing or adapting a SDM to fit a specific situation.

III. FEATURE-BASED RE PROCESS FACTORY

It should be noted that developing RE processes may not be targeted at just one software development project; organizations can use it to develop their own adaptive processes. Therefore, reusability and flexibility are the most prominent requirements in RE process development. Based on the notion of software factory, each organization can have its own RE process factory. For this purpose, a RE process product line–or "process line"–can be utilized. It has two main lifecycles, namely domain engineering (similar to ME) and application engineering (similar to SME). The domain engineering lifecycle is concerned with analyzing the domain and identifying the differences and commonalities between RE processes. The results are modeled in a RE process feature model which describes the features and their relationships; and also the RE process components which have satisfied some of the RE process features. Different RE processes can be produced depending on the features selected. The application engineering lifecycle is then applied; this activity is concerned with the elicitation of the needs, requirements, and expectations of the target RE process in order to develop the RE process which best fits the project situation and the given RE process feature model.

The process of our proposed approach (Figure 1) consists of the following stages:

1) In Situation Analysis, the method engineer analyzes the problem domain (which is the specific project situation) and as a result constructs a situation model. This model is a goal model – expressed using a specific notation, such as the i* modeling language [20] – which defines the project situation in terms of situation factors and RE process criteria (adapted from methodology requirements/criteria [4]).

2) In RE Process Component Implementation, new RE process components may be implemented, or existing ones may be adapted through refactoring; e.g., by applying generic operators [18]. In addition, a RE process feature model is constructed through applying abstraction to RE process components.

3) In Feature Analysis, in order to develop a specific RE process, the project situation is elicited in terms of situation factors and RE process criteria; the degree to which each model element should be satisfied is then computed through evaluating the situation model. After evaluation, the RE process feature model is analyzed through applying the principles of fuzzy set theory, and corresponding features are selected. For this purpose, each feature is annotated based on the Fuzzy Inference System [21] (FIS); the degree to which the model elements of the situation model should be satisfied are fed to FIS as input variables. Each FIS output value can be interpreted as the presence condition [22] for the corresponding feature (it indicates whether the feature should be kept or removed from the final selection). The process of selecting appropriate features from the RE process feature model is referred to as RE process configuration.

4) In RE Process Components Composition, a specific RE process is assembled or adapted based on the RE process configuration.

A detailed description of this approach is presented throughout the rest of this section.

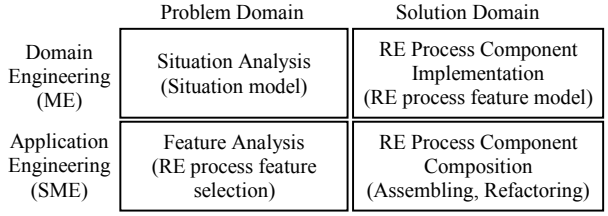

Figure 1. Phases of the proposed approach

A. Situation Model

Seamlessness and smoothness of transition between project situation/need (problem domain) and RE process feature/requirement (solution domain) is vital in process engineering. It also helps promote process requirements traceability and rationality. In other words, transition between the situation model and the RE process feature model should be seamless and smooth.

The project situation is mostly described informally, or through defining a list of characteristic-value pairs (without considering their interdependencies). In our proposed approach, the i* modeling language is used for defining the situation model in terms of situation factors (typically

modeled as goals or tasks) and RE process criteria (typically modeled as soft goals). A goal represents a precise situation factor, and a soft goal represents a qualitative situation factor with no clear-cut level of satisfaction. The situation model can thus depict the mutual effects and interactions of the situation factors (positive or negative), based on which trade-off analysis can be performed.

The situation model thus produced defines the project situation from two viewpoints: The *characteristics of the situation* as elicited from the problem domain, and the process *requirements* which are discovered by the method engineer; these requirements are at a deeper level than the problem domain, and their specification signifies a move towards the solution domain. It should be pointed out that these two viewpoints are never isolated, and have certain relationships with each other: The higher value of one situation factor might influence the values of other attributes. For example, a high value for *requirements volatility* can affect the value of *complexity*. Therefore, the relationships between these elements should also be modeled; e.g., by contribution links.

The situation factors used in our approach have been elicited through reviewing and scrutinizing the empirical studies conducted on enacting certain methods/practices in specific situations (e.g., [15, 16, 17]), as well as the project characterization phases of existing SME approaches (e.g., [11, 18, 19]); synonyms and homonyms have then been resolved based on frequency, abstraction level, and existing guidelines. The basic situation factors that can be considered when defining project situations are as follows:

- Project type: Real-time system, safety-critical system, process-controlled system, information system, and web system are examples of systems targeted by a project, thereby signifying the project type.
- Application domain: Banking and finance, education, energy resources, insurance, medical/health care, telecommunication services, government, military, and transportation are examples of common application domains.
- Project size: Size can be defined in terms of the number of staff and project requirements involved.
- Complexity: Project complexity is defined in terms of quality criteria, distribution characteristics of the application, requirements dependency, number of external interfaces, and understandability of the problem domain.
- Management commitment: This is determined based on the level of support provided, availability of influential project sponsors, and degree of management involvement.
- Degree of resistance: Interests of the people involved (which may be conflicting), attitude of the target domain towards the system, and flexibility and adaptability of the user organization are some of the sub-factors which influence this factor.
- Requirements volatility: The probability of requirements change throughout the project lifecycle. High volatility typically increases development risk.
- Level of criticality: The impact of failure: Danger to the environment, loss of human life, damage to equipment, or depletion of financial resources. An increase in criticality may require a higher level of support for formalism in the process.
- Scarcity of people and resources: Availability of personnel, time, resources, and budget are some of the sub-factors that influence this factor.
- Team size: The number of team members: This is one of the most obvious factors that have a strong impact on the RE process. If the number of people involved in the project is large, face-to-face communication will not be sufficient, and higher support for modeling and documentation might be needed.
- Familiarity with the domain: Possession of relevant knowledge and experience on the problem domain. If the project belongs to an unknown or unfamiliar domain, the development risk is high; this will require a more rigorous RE process to improve the quality of the elicitation process.
- Team RE knowledge: Relevant knowledge and experience of the team as to RE processes and techniques.
- Degree of knowledge about the requirements: The level of availability of requirements specifications, accuracy of business process descriptions, and ease of elicitation.
- The availability of skilled facilitators: The role of a facilitator is to help a group of people in defining and planning to achieve their common objectives. A facilitator can prove indispensable in an RE process.
- Potential for conflict: Stakeholder heterogeneity and the degree of conflict encountered in the problem domain are the main sub-factors that influence this factor.
- Innovation level of the project: Innovation required in the project and the development process, and the need for state-of-the-art equipment and tools.
- Customer availability: Availability of customers, whenever needed, for providing information and feedback.
- Degree of reusability: The importance of the reusability of the software project artifacts (including the customized process used), and the potential software product family.
- Degree of implicit knowledge: Signifying the importance of eliciting implicit knowledge.
- Degree of outsourcing required: The amount of project components that will be outsourced.
- Capability maturity level: The supported level of organization capability maturity, and RE process maturity.
- Organizational impact: Consequences of the project goal on the target domain organization, and the system's impact on the people involved.
- Strategic importance: Project priority and the system's effect in relation to strategic business objectives are the main sub-factors that influence this factor.

It should be mentioned that some of these situation factors (such as "Degree of reusability") are difficult to ascertain during initial cycles of process development; therefore, previous experience and meticulous scrutiny is required for resolving their uncertainty. It should also be noted that the above list is not exhaustive and only includes the core factors: Additional detail and new factors can be added, if deemed necessary.

The impact of situation factors on RE process criteria should be identified and specified as part of the situation model. RE process criteria define requirements from the method engineer's viewpoint. They can be discovered by the method engineer through analyzing the situation factors. The basic RE process criteria are as follows:

- RE process definition: The accuracy, precision, consistency, and completeness of the documentation available on the process. For a heavyweight process, a comprehensive, clear, rational, accurate, detailed, and consistent description should be provided on the lifecycle, work units, producers, modeling languages, work products, rules, and umbrella activities of the process.
- Coverage of RE lifecycle: Degree of support for the RE process life cycle – spanning negotiation, elicitation, analysis, documentation, and validation [23].
- Support for umbrella activities: Provision of adequate support for the umbrella activities that are relevant to the RE process is typically required and should be considered (e.g., requirements change management).
- Seamlessness and smoothness of transition between RE phases, stages and activities: The transition between RE phases and stages should ideally be as smooth and seamless as possible.
- Testability and tangibility of RE artifacts, and traceability to requirements: Testability of the artifacts is the degree to which an artifact facilitates testing. The understandability of an artifact to users and developers is referred to as its tangibility. Secondary RE artifacts are expected to be traceable to the main artifact: Requirements.
- Encouragement of active user involvement: RE is seriously damaged if active user involvement is neglected. Ambassador users and planning and review sessions with user participants are proven agile techniques for this purpose.
- Practicability and practicality: It should be possible to apply the RE process in practice, and in an effective and efficient manner.
- Manageability of complexity: The complexity of RE work-units and work products should be manageable, typically via applying partitioning and layering.
- Extensibility, configurability, flexibility, and scalability of the RE process: Adaptability is a very desirable trait in RE processes.
- Application scope: The intended usage context of the RE process. The combination of the project type and application domain situation factors defines the application scope.
- Support for consistent, accurate and unambiguous modeling: Diverse modeling viewpoints, logical to physical modeling, various levels of abstraction and granularity, and formal and non-formal specifications are relevant concerns in this regard.
- Provision of strategies and techniques for tackling RE model inconsistency and managing model complexity: The modeling language and modeling process are expected to provide features for managing complexity and facilitating consistency checking.

B. RE Process Feature Model

A feature model encapsulates all the features in a SPL and organizes them hierarchically. Connections between a feature and its group of children are distinguished as And- (no arc), Or- (solid arc) and Alternative-groups (unfilled arc). The children of And-groups can be either mandatory (solid circle) or optional (unfilled circle). Feature models have the following semantics: If a feature is selected, so too is its parent. Furthermore, if the parent is selected, all mandatory children (features) of an And-group are selected; in Or-groups, at least one child must be selected, and in Alternative-groups, exactly one child is selected. A feature model may also have constraints, called cross-tree constraints, which cannot be easily expressed hierarchically. Cross-tree constraints can be arbitrary propositional formulas and may be written below a feature diagram.

The RE process feature model is a compact representation of all the phases, stages, and tasks of the generic RE process in terms of features. It is the result of applying abstraction to the high-level processes of RE process models [8, 13]. In order to construct this model, we have explored existing RE processes, extracted their variations and commonalities, and represented them in the form of a feature model which constitutes the solution domain in our approach. The coverage of the proposed RE process model has been evaluated through comparison with prominent object-oriented methodologies [1], agent-oriented methodologies [24], and RE methods (e.g., [25, 26]); a mapping between the proposed RE process features and the RE activities in these well-known methodologies is given in Table I. The evaluation shows that the model provides ample coverage of RE activities in typical concrete RE processes. The major features of the RE process feature model are briefly described throughout the rest of this subsection.

Negotiation: The goals of this feature are to extract different views and needs, resolve conflicts, and reach an agreement with the stakeholders. "Negotiate with individuals", "Negotiate with groups", and "Negotiate around artifacts" are the three subfeatures of this feature. During "Negotiate with individuals", the needs and viewpoints of different stakeholders are elicited from individuals. Interview and Observation are the technical components of this feature. Through "Negotiate with groups", needs and viewpoints are extracted from groups of people. Workshop and Group media (such as project Wikis) are technical instances of this feature. Another source of knowledge extraction is the set of artifacts (such as prototypes, similar systems, and standards), the inherent

knowledge of which is extracted through "Negotiate around artifacts". Requirements prototyping, Requirements reuse, and Reverse engineering are the technical components of this feature.

Elicitation: This feature discovers the required knowledge through negotiation. It consists of five subfeatures: Identify stakeholders, Determine context/scope/interfaces, Elicit definitions, Elicit non-functional needs, and Elicit functional needs. Stakeholders, the relationships between them and their influence on the system are elicited during "Identify stakeholders". Stakeholders are the most important sources of domain knowledge. The Onion model [27] is a technical component of this feature. During "Determine context/scope/interfaces", the system context is identified by focusing on determining the system scope and its interfaces. The Rich picture method [28] of the Soft Systems Methodology (SSM) is a technical component of this feature. The project's data dictionary and glossary are extracted during "Elicit definitions". Quality attributes and constraints imposed by the stakeholders are extracted in "Elicit non-functional needs". The goal model is a technical component of this feature. "Elicit functional needs" is an integral subfeature which will be described in more detail below.

Elicit functional needs: This important feature consists of three subfeatures: Elicit usage scenarios, Elicit rationale and assumptions, and Elicit measurements. A scenario is a narrative explanation of the user's need which is elicited through "Elicit usage scenarios". Use case (structured) and User story (unstructured) [29] are technical examples of this feature. Stating the assumptions and rationale explicitly results in improvements in tracking, prioritization, and design decisions. Assumptions and rationale are extracted in "Elicit rationale and assumptions". The needs' acceptance criteria are extracted in "Elicit measurements". The measurability of needs is an important criterion for requirements validation.

Requirements analysis: This feature deals with mapping needs to requirements. This feature is the entry point to the solution domain. It consists of four subfeatures: Prioritization, Risk analysis and assessment, Feasibility analysis, and Analysis. Requirements are discovered through "Analysis". These requirements are then prioritized through "Prioritization". Binary Search Tree, Numeral Assignment Technique, Planning Game, the 100-Point Method, Theory-W, Requirements Triage, Wiegers' Method, MoSCoW Rules, and the Analytic Hierarchy Process are technical examples of this feature [30]. Requirements risk is estimated through "Risk analysis and assessment". Technical components of this feature have been presented in [31]. Feasibility assessment is performed through "Feasibility analysis". It is an engineering practice which presents information to determine whether or not the requirements should be moved forward to final engineering and construction. Since "Analysis" is a very important subfeature, it will be further described below.

Analysis: This feature can have different objectives depending on the development iteration in which it is performed. For example, in a particular iteration, classifying the needs and discovering their relationships may be the goal; whereas in another, discovering the requirements corresponding to each need may become the main objective. "Analysis" has four subfeatures: Classification, Requirements discovery, Interaction analysis, and Requirements refinement. "Classification" categorizes the requirements according to the criteria determined by the development team. A technical component of this feature is provided in [32]. Requirements are elicited from needs in "Requirements discovery" based on the previous experiences of the requirements engineer. Relationships among requirements are analyzed in "Interaction analysis". Example components for implementing this feature are presented in [33]. "Requirements refinement" reviews and revises the requirements discovered. A framework for requirements refinement has been described in [34].

Documentation: This feature's intent is to produce the software requirements specification. Documentation elaborates on the knowledge and understanding elicited from the domain. It has the following subfeatures: Qualities and constraints model, Dependency model, Definitions model, Risk model, Feasibility report, Usage model, and Requirements model. Non-functional requirements are documented through "Qualities and constraints model". Dependencies between the requirements and the needs are specified by "Dependency model". The project data dictionary is documented in "Definitions model". Requirements risks are modeled by "Risk Model", and feasibility is modeled by "Feasibility Report". Needs and usage scenarios are documented by "Usage model". Requirements discovered in analysis are documented through "Requirements model".

Requirements validation and verification: In this feature, requirements are reviewed (through informal or formal inspection) and validated. It consists of three subfeatures: Setup criteria, Translate representation, and Evaluation. The validation criteria are established in "Setup criteria". The criteria are selected according to validation intent: Completeness and testability are examples of requirements validation criteria. The requirements' representation may be changed according to the criteria through "Translate representation". After the validation criteria are determined, evaluation will be performed through "Evaluation". A framework consisting of common components for this feature is presented in [35]. For example, a validation technique can be based on prototyping, with completeness specified as an evaluation criterion; in this case, requirements are translated in the form of prototypes, and evaluation is then performed based on the scenarios previously defined.

Requirements Management: This feature is present as a continuous feature throughout the RE process, and is concerned with coping with requirements traceability, requirements change management, and other managerial issues. It consists of the following subfeatures: Tool management, Requirements traceability, Requirements change management, and Stakeholder management.

TABLE I. REALIZATION OF RE PROCESS FEATURES IN PROMINENT SOFTWARE DEVELOPMENT METHODOLOGIES

Methodology		RE Activity/Practice	Corresponding Process Features
Object-Oriented [1]	Fusion	Develop overall object model	Negotiation, Analysis, Requirements model
		Develop system object model	Determine context/scope/interfaces, Elicit usage scenarios, Usage model
		Develop system interface model	Determine context/scope/interfaces, Analysis, Requirements model
		Develop data dictionary	Elicit definitions, Definitions model
		Evaluate against checklist	Requirements validation and verification
	RDD	Identify classes, responsibilities, and collaboration	Analysis, Requirements model
		Analyze subsystems and hierarchies	Determine context/scope/interfaces, Interaction analysis, Requirements model
	RUP	Analyze problem	Identify stakeholders, Determine context/scope/interfaces, Analysis, Elicit definitions, Definitions model
		Understand stakeholder needs	Negotiation, Elicit non-functional needs, Elicit functional needs
		Define system	Elicit usage scenarios, Documentation
		Manage system scope	Prioritization, Risk analysis and assessment
		Refine system definition	Analysis, Documentation
	DSDM	Feasibility study	Elicitation, Negotiation, Feasibility analysis
		Business study	Determine context/scope/interfaces, Prioritization, Elicit definitions, Definitions model
		Functional model iteration	Analysis, Requirements model, Documentation, Requirements validation and verification
	XP	Develop user stories	Elicit usage scenarios, Usage model
		Planning game	Feasibility analysis
		Develop metaphor	Elicitation, Negotiation, Documentation, Requirements validation and verification
		Prioritize user stories	Prioritization
		Hold daily stand up meeting	Negotiation with groups, Negotiation around artifacts
		Analysis	Analysis
		System-wide verification and validation	Requirements validation and verification
	ASD	Specify project mission	Elicitation, Requirements analysis
		Create mission artifacts	Documentation
		Obtain approval, Share mission value	Negotiation
		Quality review, Final Q/A and release	Requirements validation and verification
	FDD	Domain-area walkthrough, Study documents	Elicitation, Negotiation
		Develop small group models and a team model	Analysis, Requirements model
		Refine overall object model	Requirements refinement
		Build features list	Analysis
		Plan by feature	Prioritization, Risk analysis and assessment, Risk model
		Software/Models inspection and reporting	Requirements validation and verification
Agent-oriented [24]	MaSE	Capturing goals	Analysis
		Applying use cases	Elicit usage scenarios, Interaction analysis, Usage model
		Refining roles	Requirements refinement, Requirements validation and verification
	Tropos	Early requirements	Identify stakeholders, Stakeholder management, Qualities and constraints model, Dependency model
		Late requirements	Determine context/scope/interfaces, Analysis, Qualities and constraints model, Dependency model, Requirements model
	Gaia	Identify the roles	Identify stakeholders, Elicit functional needs
		Identify and document role protocols	Determine context/scope/interfaces, Dependency model, Requirements traceability
		Elaborate the roles model	Dependency model, Usage model, Requirements model
RE Methods	Volere [25]	Project blastoff	Elicitation, Risk analysis and assessment
		Trawl for knowledge	Negotiation
		Write the requirements	Elicitation, Documentation
		Quality gateway	Requirements validation and verification
		Prototype the requirements	Documentation, Requirements validation and verification
		Requirements retrospective	Requirements management, Risk analysis and assessment, Requirements validation and verification
		Taking stock of the specification	Requirements analysis
		Domain analysis	Elicitation, Negotiation
		Reusing requirements	Requirements analysis
	KAOS [26]	Identify goals and their concerned objects	Elicit non-functional needs, Elicit functional needs, Requirements traceability
		Identify potential agents and their capabilities	Identify stakeholders, Determine context/scope/interfaces
		Operationalize goals into constraints	Analysis
		Refine objects and actions	Requirements refinement
		Derive strengthened objects and actions to ensure constraints	Requirements validation and verification
		Identify alternative responsibilities	Risk analysis and assessment
		Assign actions to responsible agents	Analysis

Identification, selection and use of suitable tools are supported through "Tool Management". Tracing forward and backward between requirements and artifacts is managed through "Requirements traceability". Since changes are inevitable, they have to be managed through a dedicated feature: "Requirements change management". "Stakeholder management" is concerned with keeping track of stakeholders, analyzing their influences, prioritizing them, and involving them in the process.

C. Annotating the RE Process Feature Model

The method engineer usually uses guidelines to select appropriate techniques for a given feature (stage or task); a simple example is shown in Table II. Feature values should be entered into the feature model as informal linguistic terms (such as good, or poor). The guideline corresponding to each of the features in the feature model can be modeled through applying FIS, thereby annotating each feature of the RE process feature model by its corresponding guideline.

TABLE II. EXAMPLES OF RE GUIDELINES (ADAPTED FROM [11, 14])

Situation		Brain-Storming	Focus Group	JAD	…
Time Constraints	very high	−	××	××	…
	high	✓	×	−	…
	medium	✓✓	−	✓	…
	low	✓✓	✓	✓✓	…
	very low	✓✓	✓✓	✓✓	…
Complexity	very high	✓✓	✓✓	✓✓	…
	high	✓✓	✓✓	✓✓	…
	medium	✓	✓	✓✓	…
	low	−	−	✓✓	…
	very low	×	×	✓✓	…

✓✓= very good; ✓ = good; − = borderline; ×= weak; ×× = very weak

FIS is commonly used for handling uncertainty, vagueness, and imprecision of judgment in multi-objective decision-making processes. It uses a Fuzzy Rule Engine (FRE) for mapping an input space to an output space based on fuzzy logic, in which the truth of any statement becomes a matter of degree (by assigning a degree of membership); a Membership Function (MF), which is a generalization of the indicator function in classical sets, defines a curve that maps each point to a membership value (or degree of membership) between 0 and 1. The input and output variables of FIS typically use a specific form of normal fuzzy set, called fuzzy numbers, which can be formulated by trapezoidal fuzzy numbers. A trapezoidal fuzzy number is defined as: $\varpi = (a, b, c, d),\ a \le b \le c \le d$ (if $b = c$, ϖ is a triangular fuzzy number); the MF can be defined as shown in figure 2. FRE is a program that tries to derive a conclusion from a rule base (a set of logic rules in the form of IF-THEN statements).

Throughout the rest of this subsection, we will develop a simple FIS for the Brain-Storming feature (BSF); the result can be used as a presence condition for BSF. For this purpose, the following tasks are performed: 1) Defining input and output variables: In this example, we have two input variables, time constrains and project complexity; and one output variable, fitness of Brain-Storming; 2) Defining the MF for each variable in the form of its scale, as shown in Figure 3; and 3) Defining rules of FIS according to RE guidelines. As an example, some of the corresponding rules for table II are shown in Figure 4.

$$\mu_{\varpi}(x)\begin{cases}\frac{x-a}{b-a} & a \le x \le b\\ 1 & b \le x \le c\\ \frac{x-d}{c-d} & c \le x \le d\\ 0 & other\end{cases}$$

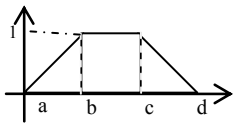

Figure 2. MF for a trapezoidal fuzzy number

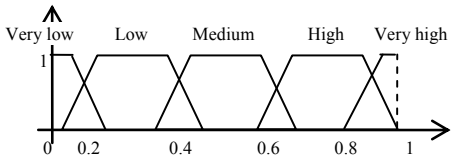

Figure 3. Model of MF for input variables

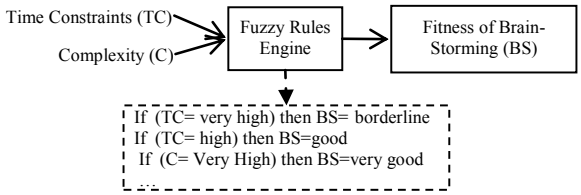

Figure 4. Conceptual model for FIS of a feature

D. RE Process Feature Selection

In order to select RE process features, the following activities should be carried out:

1) The values for situation factors are elicited and the situation model is evaluated; the results determine the degree to which each element of the situation model should be satisfied (in other words, the input variables for FIS are determined through evaluation). The values of situation factors can be determined in two ways: *Evidential* (based on solid evidence) or *Assumptive* (without significant supporting evidence). Situation model evaluation typically starts by assigning values to leaf nodes (these values are then propagated to other nodes). This can be performed by *Automatic resolution* or *Manual resolution*. Automatic resolution follows the i* forward evaluation algorithm [36]. Value propagation will be performed with respect to the current value of the situation factor and the type of the contribution link, based on the propagation rules defined in Table III. If a factor receives various values through its decomposition links, the minimum label is assigned (*Satisfied* > *Weakly Satisfied* > *Conflict* > *Unknown* > *Weakly Denied* > *Denied*). Manual resolution is needed where the automatic approach cannot be applied; in such cases, human judgment determines the value of a factor.

2) The corresponding FIS of each feature in the RE process feature model is used for indicating the fitness of that feature based on the results of situation model evaluation.

The method engineer then selects each feature based on its fitness value and his/her previous experience. Selection is thus a decision making problem: The fitness value of each feature helps the method engineer reach the final decision. This process can be automated, if the method engineer follows a specific pattern in making the decisions (e.g., as a simple example, a feature may be selected if the fitness value is greater than 0.5).

TABLE III. VALUE PROPAGATION RULES (ADAPTED FROM [36])

Original Value		Contribution Link Value				
Label	Name	Help	Hurt	Some+	Some-	Unknown
✓	Satisfied	✓.	✗.	✓.	✗.	;.
✓.	Weakly Satisfied	✓.	✗.	✓.	✗.	;.
;.	Unknown	;.	;.	;.	;.	;.
✗.	Weakly Denied	✗.	;.	✗.	;.	;.
✗	Denied	✗.	;.	✗.	;.	;.

E. Assembly and Refactoring of RE Process Components

RE process composition can be performed through the following three strategies: Assembly-based strategy, Extension-based strategy, and Paradigm-based strategy (meta-model instantiation). Assembly and refactoring of RE process components are the major activities in these strategies [18]. Assembly can be regarded as the act of composing RE processes from cohesive RE process components through integration (refactoring may then be needed), or through association based on their preconditions and post-conditions aimed at achieving specific intentions. Integration involves identifying the common features of RE process components and merging them (through merge, generalization, and specialization operators); this can be done if the elements of integration have the same semantics. The merge operator is applicable for merging components with similar semantics and similar structures. The generalization operator can be used when components have the same semantics but different structures. The specialization operator is used when one component is a specialization of another. Assembly by association simply involves ordering RE processes. The interested reader is referred to [18, 19, 37] for a more detailed discussion on the relevant operators and formal definition techniques.

Validation provides a means for ensuring that the result of composing the RE process is suitable for the given situation, and satisfies the quality criteria. As an example, Harmsen [19] has defined "completeness" as a criterion to ensure that the target process contains all the process components referred to by its constituent components; it is in turn divided into: Input/output completeness, content completeness, process completeness, association completeness, and support completeness. Furthermore, the quality of the constructed situational RE process can be validated based on assessment models such as the RE process maturity model [8], and the major concerns addressed by the RE assessment model [11].

IV. ILLUSTRATIVE EXAMPLE

This section provides an illustrative example for demonstrating how to use the proposed approach.

Market-driven software development companies have unique traits and needs, such as short time-to-market, remote users, very limited opportunity for negotiation, steady stream of new requirements, and the need for frequent delivery of new and improved product releases in order to keep users and customers satisfied. As an illustrative example, we have considered a market-driven situation where a real-time developer studio should be developed. The method engineer uses extension-based SME to construct the RE method by extending the Scrum framework. Each Scrum iteration consists of: *Sprint planning meeting, Daily Scrum, Sprint review,* and *Sprint retrospective*. Scrum needs to be extended to be suitable for a market-driven situation.

Due to space constraints, constructing a complete process, with all the associated detail, is not possible herein; therefore, for sake of brevity, only two features – "Negotiation" and "Elicit usage scenarios" – are considered. Because the extension-based SME approach is selected, a Scrum process line must be constructed in *domain engineering*. Hence, the method engineer models the situation factors and their relationships in the situation model, and produces a Scrum feature model (spanning original features and potential features) as the RE process feature model; the results are shown in figure 5. The method engineer must elicit the values of the situation factors, using both *evidential* and *assumptive* approaches (a subset of which is shown in Table IV). For example, based on team knowledge, the value of *understandability of problem domain* is set to *weakly satisfied*, evidentially. After determining the value of the leaf nodes, the remaining nodes are evaluated by using *automatic resolution* as well as *manual resolution*. For instance, the value of *determine complexity* has been defined as *weakly satisfied* by manual resolution. Automatic resolution could not be applied because it produced different values: Based on the "minimum label" rule for decomposition links, it is evaluated as *weakly denied*, whereas application of the propagation rule determines its value as *weakly satisfied* (because of its *Some+* link). As another example, the value for *determine scarcity of people and resources* has been determined by automatic resolution.

It should be noted that even though the values of the situation factors are determined according to the project/organization situation, there may be a degree of uncertainty involved, which has been handled through applying fuzzy logic in the proposed process. The fitness of each feature is then calculated based on the FIS values. For this example, the fuzzy logic toolbox of MATLAB has been used for implementing FIS; corresponding fitness values are determined by considering the values of the situation model (RV = *Satisfied*, C = *Weakly Satisfied*, etc.) as inputs for the annotated FISs. For example, use cases are widely applied to *elicit usage scenarios*. However, the fundamental problem in market-driven RE is that the user is not known beforehand; therefore, based on the implemented guidelines of FIS, the

persona-scenario model [38] has a better fitness value. Finally, the method engineer uses the fitness values to make a decision about selecting the final features, and composes the selected features to develop the targeted situational (*project*-specific) RE process. The schema of this process is shown in figure 5. It should be noted that there are several resources for discovering RE process components and their corresponding guidelines; examples include: Relevant RE literature [5, 8, 11, 13, 14, 23], method repositories [39, 40, 41], and empirical studies.

TABLE IV. A SUBSET OF THE SITUATION FACTORS ADDRESSED IN THE ILLUSTRATIVE EXAMPLE

Situation Factor	Value
Various type of quality criteria	Medium
Distributed application structure	Unknown
Requirements dependency	High
Number of external interfaces	5
Types of customers	Not clearly known
Requirements handling	Continuous requirements flow
Organizational support	High

V. CONCLUSIONS AND FUTURE WORK

In this paper, a process factory for developing situational RE process is proposed. To this aim, a feature-based SPL is applied; it takes into account the domain-engineering lifecycle as ME, and the application lifecycle as SME, specifically targeted at developing RE processes. It also implicitly distinguishes between the problem and solution domains, and provides a mapping between the two through the use of FIS and guidelines. Thus, the proposed approach applies SPL best practices to improve SME, and uses SME to construct situational RE processes.

The proposed approach can also be used as a convenient way for developing software processes; for this purpose, process components and the corresponding feature model should be adapted to the target context. Constructing full-scale processes from scratch, without the use of repositories of method components and guidelines, can be a complicated and time-consuming process. Nevertheless, if an evolutionary strategy is applied to the enactment and management of the process factory and the constructed processes, development costs can be reduced in the long run; this strategy ensures that the lessons learnt from previous projects are reflected to the process factory and its products, mainly as new or updated guidelines, situation patterns, and feature models.

This research can be furthered by creating a tool to construct bespoke RE processes supporting this approach. Another strand of research can focus on constructing a repository of RE process components, providing detailed specifications for the features. It can also be completed through the use of feature-oriented process languages for implementing RE process components by considering existing repositories, process pattern languages, and process metamodels. In this manner, similar to feature-oriented programming in software development, method engineers will be able to program their own processes through feature-oriented process languages.

REFERENCES

[1] R. Ramsin and R. F. Paige, "Process-Centered Review of Object-Oriented Software Development Methodologies," ACM Computing Surveys, vol. 40, no. 1, pp. 1–89, 2008.

[2] L. Osterweil, "Software processes are software too, revisited," In Proceedings of the 19th International Conference on Software Engineering, pp. 540–548, 1997.

[3] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," Information and Software Technology, vol. 38, no. 4, pp. 275–280, 1996.

[4] R. Ramsin and R. F. Paige, "Iterative criteria-based approach to engineering the requirements of software development methodologies," IET Software, vol. 4, no. 2, pp. 91–104, 2010.

[5] G. Davis, "Strategies for information requirements determination," IBM Systems Journal, vol. 21, no. 1, pp. 4–30, 1982.

[6] S. Beecham, T. Hall, and A. Rainer, "Software process improvement problems in twelve software companies: An empirical analysis," Empirical Software Engineering, vol. 8, no. 1, pp. 7-42, 2003.

[7] S. Beecham, T. Hall, and A. Rainer, "Defining a requirements process improvement model," Software Quality Journal, vol. 13, no. 3, pp. 247–279, 2005.

[8] I. Sommerville and P. Sawyer, Requirements Engineering: A good practice guide, John Wiley & Sons, 1997.

[9] C. R. Coulin, A situational approach and intelligent tool for collaborative requirements elicitation, Doctoral Thesis, University of Technology, Sydney, 2007.

[10] P. Clements and L. Northrop, Software Product Lines, Addison-Wesley, 2001.

[11] L. Jiang, A framework for requirements engineering process development, Ph.D. Dissertation, University of Calgary, 2005.

[12] S. Lauesen, Software Requirements: Styles and Techniques, Addison-Wesley, 2002.

[13] I. Alexander and L. Beus-Dukic, Discovering Requirements: How to Specify Products and Services. John Wiley & Sons, 2009.

[14] N. Maiden and G. Rugg, "ACRE: selecting methods for Requirements acquisition," Software Engineering Journal, vol. 11, no. 3, pp. 183–192, 1996.

[15] R. Moore, K. Reff, J. Graham, and B. Hackerson, "Scrum at a Fortune 500 Manufacturing Company," In Proceedings of Agile'07 Conference, pp. 175–180, 2007.

[16] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, "An experimental investigation of personality types impact on pair effectiveness in pair programming," Empirical Software Engineering, vol. 14, no. 2, pp.187–226, 2009.

[17] M. J. O'Donnell and I. Richardson, "Problems Encountered When Implementing Agile Methods in a Very Small Company," Communications in Computer and Information Science, vol. 16, no. 1, pp. 13–24, 2008.

[18] J. Ralyté, R. deneckére, and C. Rolland, "Towards a generic model for situational method engineering," Lecture Notes in Computer Science, vol. 2681, pp. 95–110, 2003.

[19] A. F. Harmsen, Situational Method Engineering, Doctoral Thesis, University of Twente, 1997.

[20] E. Yu. "Towards modelling and reasoning support for early-phase requirements engineering," In Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, pp. 226–235, 1997.

[21] J. Jantzen, Foundations of Fuzzy Control. John Wiley & Sons, 2007.

[22] K. Czarnecki and M. Antkiewicz, "Mapping features to models: A template approach based on superimposed variants," Lecture Notes in Computer Science, vol. 3676, pp. 422–437, 2005.

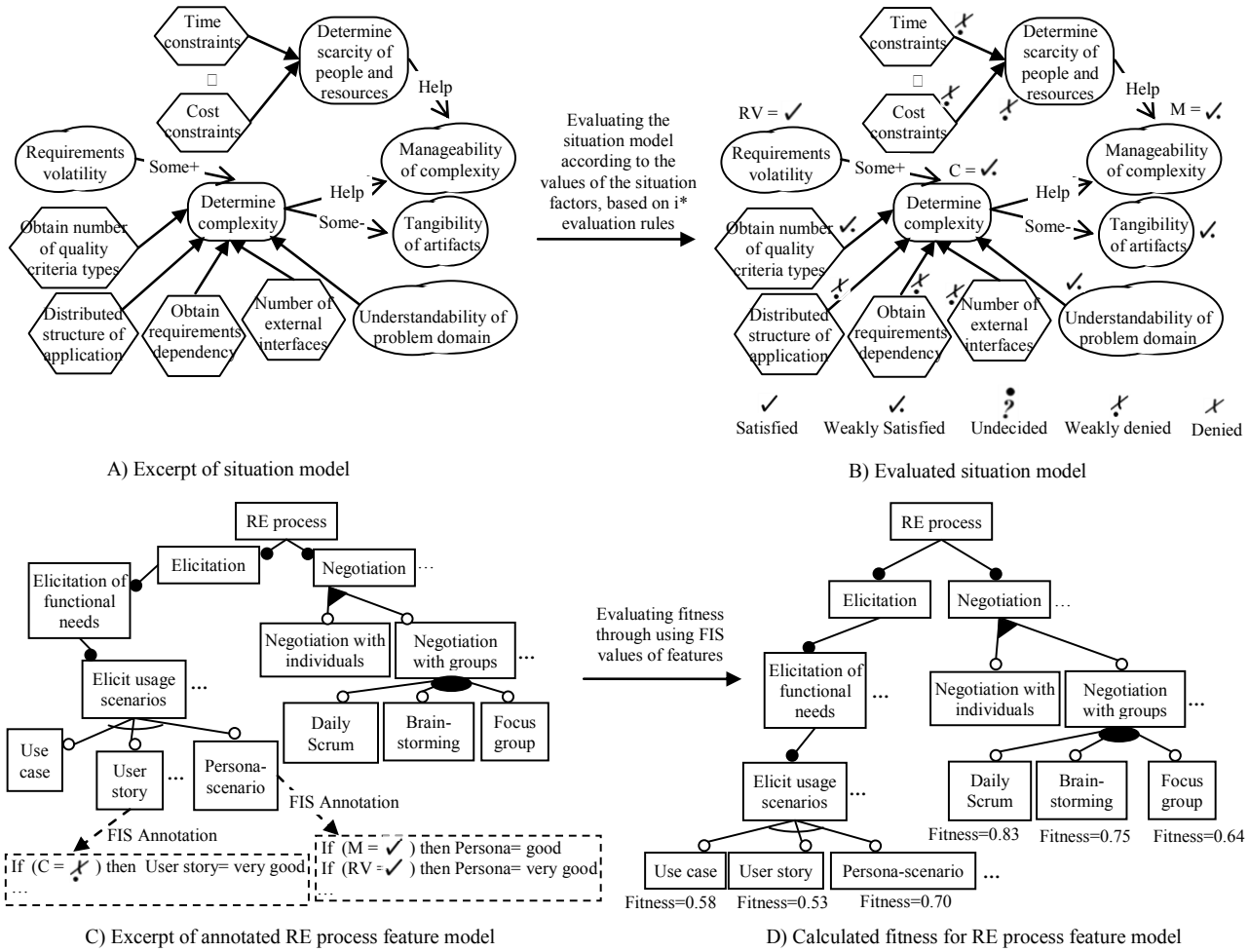

Figure 5. Schema of the proposed approach, as applied in the illustrative example

[23] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques. John Wiley & Sons, 1998.

[24] B. Henderson-Sellers and P. Giorgini, Agent-Oriented Methodologies. IGI Global, 2005.

[25] S. Robertson and J. Robertson, Mastering the Requirements Process. Addison-Wesley, 2006.

[26] A. Van Lamsweerde, Requirements Engineering: From system goals to UML models to software specifications. John Wiley & Sons, 2009.

[27] I. Alexander, "A Taxonomy of Stakeholders: Human Roles in System Development," International Journal of Technology and Human Interaction, vol. 1, no. 1, pp. 23–59, 2005.

[28] P. Checkland, "Soft Systems Methodology: A Thirty Year Retrospective," Systems Research, vol. 17, no. 1, pp. 11–58, 2000.

[29] M. Cohn, User Stories Applied: For Agile Software Development. Addison-Wesley, 2004.

[30] Q. Ma, "The effectiveness of requirements prioritization techniques for a medium to large number of requirements: A systematic literature review," AUT University, 2010.

[31] D. Vose, Risk Analysis: A Quantitative Guide. John Wiley & Sons, 2008.

[32] N. Nurmuliani, D. Zowghi, and S. P. Williams, "Using card sorting technique to classify requirements change," In Proceedings of the 12th IEEE International Requirements Engineering Conference. pp. 240–248, 2004.

[33] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements Interaction Management," ACM Computing Surveys, vol. 35, no. 2, pp. 132–190, 2003.

[34] W. Q. Liu, "A requirements refinement framework," In Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 658–659, 2008.

[35] A. Katasonov and M. Sakkinen, "Requirements quality control: A unifying framework," Requirements Engineering, vol. 11, no. 1, pp. 42–57, 2006.

[36] J. Horkoff and E. Yu. "A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models," In Proceedings of CEUR Workshop in CAiSE, pp. 19–24, 2009.

[37] D. Gupta and N. Prakash, "Engineering methods from method requirements specifications," Requirements Engineering, vol. 6, no. 3, pp. 135–160, 2001.

[38] M. Aoyama, "Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products," In Proceedings of the 13th IEEE International Requirements Engineering Conference, pp. 85–94, 2005.

[39] Rational Process Library. http://www-01.ibm.com/software/awdtools/rmc/library/

[40] OPEN Process Framework (OPF) Repository. http://www.opfro.org/

[41] Ralph Young repository. http://www.ralphyoung.net/artifacts.html.