Semi-Automatic Composition of Situational Methods

Anat Aharoni
Department of Information Systems Engineering, The School of Engineering, Kinneret College, Israel
Contact Details:
Anat Aharoni
Department of Information Systems Engineering
Kinneret College on the Sea of Galilee,
Mobile Post Emek Hayarden, Israel
anat@a-aharoni.com, anata@kinneret.ac.il

Iris Reinhartz-Berger*
Department of Information Systems, University of Haifa, Haifa 31905, Israel
Contact Details:
Iris Reinhartz-Berger
Department of Information Systems
University of Haifa
Carmel Mountain, Haifa 31905, Israel
iris@is.haifa.ac.il
phone: 972-4-8288502
fax: 972-4-8288522

* Corresponding author
Semi-Automatic Composition of Situational Methods

ABSTRACT

Situational methods are approaches to the development of software systems that are designed and constructed to fit particular circumstances that often refer to project characteristics. One common way to create situational methods is to reuse method components, which are the building blocks of development methods. For this purpose, method components have to be stored in a method base, and then retrieved and composed specifically for the situation in hand. Most approaches in the field of situational method engineering require the expertise of method engineers to support the retrieval and composition of method components. Furthermore, this is usually done in an ad-hoc manner and for pre-defined situations. In this paper, we propose an approach, supported by a tool, that creates situational methods semi-automatically. This approach refers to structural and behavioral considerations and a wide variety of characteristics when comparing method components and composing them into situational methods. The resultant situational methods are stored in the method base for future usage and composition. Based on our experimental study of the approach, we claim that it provides correct and suitable draft situational methods, which human evaluators have assessed as relevant for the given situations.

Keywords: Situational Method Engineering, ISO/IEC 24744, Development Methods, Metamodeling

INTRODUCTION

Method engineering deals with the design, construction, and adaption of approaches, techniques, and tools for the development of information and software systems (Brinkkemper, 1996). Siau, Long, and Ling (2010) claim that development methods are one of the key factors for the success of information systems development. However, since projects vary in their characteristics, standard development methods in a textbook or manual may require specific adaptations so that they will support all software development properly. Situational method engineering (SME), which is a sub-field of method engineering, focuses on in-house construction of organization- or project-specific development methods (Kumar and Welke, 1992; Brinkkemper, 1996; Domínguez and Zapata, 2007; Henderson-Sellers and Ralyté, 2010). The main terms used in SME are method components, situations, and situational
methods (Mirbel and Ralyté, 2006; Henderson-Sellers and Ralyté, 2010). Method components, the building blocks of SME, are development methods, or any coherent parts of them. A situation can be defined as a vector of characteristics that relate to various entities in software development, such as the project in hand, the software development organization, the software development team, and so on. Finally, a situational method is an approach used in the development of software systems that is designed and constructed to fit particular situations.

In their review of twenty method engineering approaches, Becker, Janiesch, and Pfeiffer (2007) found five important mechanisms for composing method components into situational methods. The most utilized mechanism is aggregation, which combines independent method components to create a "larger" method component. This mechanism, which is also called assembly, construction, or integration, appeared in 70% of the reviewed approaches. Specialization, which is sometimes called tailoring, was found to be the second most popular mechanism, appearing in 45% of the approaches. Analogy construction (van Offenbeek and Koopman, 2006; Ralyté and Rolland, 2001; Raylte, Deneckere, and Rolland, 2003), configuration (Karlsson and Ågerfalk, 2005; Becker, Knackstedt, Pfeiffer, and Janiesch, 2007), and instantiation (Nuseibeh, 1994) were utilized much less frequently and usually in addition to aggregation or specialization. Becker et al. (2007) further claim that aggregation and specialization, which are classified by Ralyté et al. (2003) as assembly-based method engineering, can be used in a much wider variety of situations than the other mechanisms, as they provide flexible means to adapt a solution to the specific needs of a given situation.

Recently, due to the increasing number and variety of development methods and the emerging requirements of development processes (e.g., in the form of the CMMI model (Chrissis, Konrad, and Shrum, 2003)), efforts have been made to standardize the area of method engineering. These efforts have yielded the OPEN Process Framework (OPF) (Firesmith and Henderson-Sellers, 2001; Henderson-Sellers and Serour, 2005), OMG's Software Process Engineering Metamodel (SPEM) (OMG, 2005), and ISO/IEC 24744 (Gonzalez-Perez, 2007; ISO, 2007). These frameworks and approaches specify the core terminology of development
methods and divide method components primarily into structural and behavioral ones. Structural method components, also called products or work products, represent the different possible artifacts in the development methods, such as documents, while behavioral method components, which are also called work units or processes, represent tasks, techniques, and activities in the development lifecycles. Other aspects, e.g., the stakeholder's involvement in software development, temporal aspects, and the language and modeling units, are also handled in these frameworks and approaches.

In order to guide the retrieval and composition of situational methods, the method components are associated with various situational characteristics, i.e., features that characterize certain situations. Examples of situational characteristics mentioned in the literature are: type of development, stakeholder cohesion or contention, project scale, distribution of project organization, domain experience of development team, degree of novelty, technical complexity, management complexity, architectural risk, incremental method evolution, company conditions, and organizational culture (Park, Na, Park, and Sugumaran, 2006; van de Weerd, Versendaal, and Brinkkemper, 2006). Mirbel and Ralyté (2006) propose what they term a 'reuse frame' for aggregating different situational characteristics relevant to a single critical development aspect. According to their proposal, situational characteristics are primarily divided into human-related, organizational, and application domains. The different characteristics in the reuse frames are organized in a tree of successively refined aspects, such that the actual characteristics appear in the leaves of the tree. A software development organization may adopt only the portion of this tree that fits its requirements.

While the different approaches in situational method engineering concentrate on providing representation and measuring aids to assist method engineers to create situational methods, these methods are usually actually composed and created manually by method engineers, who base their work on their experience and understanding of the situations and of the various method components. Furthermore, they usually compose the methods utilizing structural considerations only, in an ad-hoc manner, and for pre-defined situations. In this work, we
present an approach, supported by a tool, the objective of which is to retrieve the most suitable method components for given situations and compose them semi-automatically into situational methods. The method components and resultant situational methods are represented in an XML-based notation, visualized by Object-Process Methodology (OPM) (Dori, 2002), and stored in a method base. Four similarity metrics, which refer to both the structural and the behavioral aspects of the method components as well as to additional meta-information regarding their role and essence, are identified. Having calculated the similarity between different method components, the approach defines two structural composition operations (merging and generalization) and five behavioral composition operations (sequential, concurrent, incremental, iterative, and alternative compositions) that can be used for creating situational methods. An experimental study of the suggested approach has shown that the popular aggregation and specialization mechanisms are supported at a satisfactory level, yielding situational methods that suit the given situations.

The main contribution of the study is two-fold. First, it enhances the retrieval and the composition of method components with both structural and behavioral aspects, and utilizes them in a semi-automatic manner, thus providing method engineers with a systematic tool. Secondly, the study compares the recommendations of humans (advanced and graduate students of information systems) and of an automatic tool for the same situations, and analyzes the differences between them and the possible reasons for these differences.

The structure of the rest of the paper is as follows. First, we provide an overview of the approach, presenting its method component model. We then elaborate on retrieval issues and the composition capabilities of the approach in separate sections. A report of our experience with the approach follows. Finally, we refer to related work, highlight the benefits and shortcomings of the suggested approach, summarize the work, and suggest future research directions.
THE APPROACH OVERVIEW

The automatic creation of situational development methods requires the existence of a *method base*, which stores models of method components or of complete methods that are commonly used or can be used in a certain software development organization (Saeki, Iguchi, Wen-yin, and Shinohara, 1993; Ralyté, 1999). A possible scenario of such an automatic creation may include the following four steps (depicted in Figure 1): (1) specifying the situation query for which a development method is required; (2) retrieving potential or relevant method components from the method base; (3) applying different composition operations in order to satisfy completely or partially the situation query; and (4) representing and ranking the suggested situational methods according to their suitability to the query in hand. The situational methods thus created are then recorded in the method base for future use.

![Figure 1. A possible scenario of automatic creation of situational development methods](image)

Before these activities can be carried out, a method component model first has to be defined. This model must support different types of method component, including: fragments, which can be product or process parts of development methods (Harmsen, Lubbers, and Wijers, 1995); chunks, which are autonomous and coherent parts of methods that support the realization of some specific information systems development activities (Kornyshova, Deneckere, and Salinesi, 2007); method services, which incorporate assistance-based utilities.
into method components [0]; road maps, each of which is composed of one or more coherent sequences of method chunks (Rolland, 2009); and patterns, which suggest domain-specific guidance to the creation of process or product fragments (Rolland, Nurcan, and Grosz, 2000). Thus, we use the following general definition for a method component model.

**Definition 1 (method component model):** A method component model, $MC$, is a quintuplet, $<n, r, Str, Bhv, SC>$, where:

- $n$ is the name of the method component;
- $r$ is the role the method component plays in the context of method engineering (e.g., a document, a technique, or a model unit).
- $Str$ represents the structure of the method component;
- $Bhv$ represents the behavior of the method component;
- $SC$ represents the situation which the method component fits. In other words, $SC$ is a set of pairs $<f, v>$, where $f$ is a situational feature and $v$ is its value in the particular method component.

The role of the method component is taken from a method engineering framework, such as ISO/IEC 24744 (ISO, 2007), which defines a metamodel for development methodologies. This standard refers primarily to five methodological aspects: (1) work units describing behaviors; (2) work products describing structures and artifacts; (3) producers specifying human-related aspects; (4) model units specifying the language aspect of development methods; and (5) stages specifying the temporal aspect of method components. Each aspect is further specialized to represent particular methodological concepts, such as tasks, techniques, and processes, all of which are work units. As will be explained later, the role of the method component is used when measuring the similarity between method components.

The structure and behavior of a method component are now defined.

**Definition 2 (structure of method component):** The structure, $Str$, of a method component, $MC$, is a triple $<Exh, Agg, Rel>$, where:

1. $Exh$ is the set of features and attributes of $MC$, and is called its **exhibition set**;
2. $Agg$ is the set of parts of $MC$, and is called its **aggregation set**;


(3) \( \text{Rel} \) is the set of the method components to which \( \text{MC} \) is related, and is called its \textit{relation set}.

Note that the elements in the aggregation and relation sets are method components in their own right, whereas the elements in the exhibition set may not be method components. Nevertheless, we refer to all the members in these three sets as elements that can be composed according to similarity metrics, i.e., as method components.

**Definition 3 (behavior of method component):** The behavior, \( \text{Bhv} \), of a behavioral method component \( \text{MC} \) is a triple \(<\text{Inp}, \text{Out}, \text{Tri}>\), where:

(1) \( \text{Inp} \) is the set of inputs of \( \text{MC} \), and is called its \textit{input set};

(2) \( \text{Out} \) is the set of outputs of \( \text{MC} \), and is called its \textit{output set};

(3) \( \text{Tri} \) is the set of triggers of \( \text{MC} \), as is called its \textit{trigger set}.

Here again, the elements in these sets are method components. The inputs and the outputs of a work unit, for example, may be work products, whereas its triggers may be producers or work products.

In order to visualize the method component model described above, we use a simplified version of Object-Process Methodology (OPM) (Dori, 2002), which is a holistic approach for the modeling, study, development, and evolution of software systems. It combines ideas from object-oriented and process-oriented approaches into a single frame of reference, making it possible to express mutual relationships and effects between objects (structure) and processes (behavior). The main reasons for choosing OPM for the purpose of representing method components are: its balanced treatment of structure and behavior, and the mutual relationships between them; its scalability, which is achieved through refinement and abstraction mechanisms that enable recursive specification of the modeled element to any desired level of detail without losing the legibility, comprehension, and consistency of the complete model; its formality expressed by a metamodel (Reinhartz-Berger and Dori, 2005); and its accessibility to different types of users, as has been examined by Peleg and Dori (2000) and Reinhartz-Berger and Dori (2005). More details about the visual representation of method components in OPM can be found in the work of Aharoni and Reinhartz-Berger (2008).
As an example of the proposed method component model, consider the processes "Requirements Extraction" and "Manage User Stories," represented in Figure 2. As noted, a process in ISO/IEC 24744 is a large-grained work unit operating within a given area of expertise. The "Requirements Extraction" process, which is taken from the Rational Unified Process (RUP) (Kruchten, 2000), supports the procedure of discovering the requirements of a software system through communication with clients. The main artifacts of this process are requirements documents and business domain glossaries. The process is triggered by the requirements engineers and the systems analysts who use the client's initial information, follow the contract (and modify it if required), and also utilize various requirement templates. The clients themselves may be involved in this process too. In addition, this model specifies the situations for which the entire "Requirements Extraction" process is suitable: the minimal requested capability level is 2, the level of the project's flexibility to changes is low, and the project duration is at least one year. These situational characteristics, which appear in grey in the figure, are classified according to the reuse frame proposed in [0], and recorded in the upper left corner of the shapes as roles. Situational characteristics that do not appear in this reuse frame, such as Method Source, are classified as general.

The model also specifies that the process comprises three tasks: (1) obtain an initial understanding of the domain, (2) draw up a set of requirements, and (3) delimit the domain scope. Each task is a method component into which the relevant inputs, outputs, and triggers are percolated. The model of the method component "draw up a set of requirements" (not shown here), for example, includes its trigger (the end of the task "obtain an initial understanding of the domain"), its inputs ("Client's Initial Information," "Contract," "Requirements Templates," and "Business Domain Glossary"), and its outputs ("Business Domain Glossary," which may be modified by this task, and "Requirements Document"), as well as additional specific situational characteristics and recommended techniques for performing this task. Rules are defined and maintained in OPM in order to preserve consistency between diagrams that contain the same element (method component), describing its different aspects (Dori, 2002).
The "Manage User Stories" process, taken from eXtreme Programming (XP) (Beck and Andres, 2004), comprises the tasks "Write User Stories" and "Create Acceptance Tests". The techniques "Write 3 Sentences in Customer's Terminology" and "Focus on User Needs and Benefits" are recommended for composing user stories; customers and developers are required to participate in this task.

Figure 2. The Requirements Extraction process taken from RUP: (a) an OPM model and (b) a formal specification. The Manage User Stories process taken from XP: (c) an OPM model and (d) a formal specification. The method components in the aggregation, input, output, trigger, and situational characteristics sets are represented by their names.
RETRIEVAL CAPABILITIES

For the purpose of retrieving relevant method components, we define four types of similarities between method components, linguistic, meta-informational, structural, and behavioral, each of which is described in the following sub-sections. The general (overall) similarity is a weighted average of the relevant linguistic, meta-informational, structural, and behavioral similarities, and is formally defined in Definition 9. The weights are set and maintained by the method engineers, according to different characteristics of the models in the method base. A discussion of these weights, as well as reference to the usage of situational characteristics and the definition of situation queries in the retrieval process, are given later in this section.

Linguistic Similarity

The similarity between elements is commonly defined in terms of the measured distance between their names. Different similarity metrics have been proposed for calculating the distances between terms and sentences (see, for example, a summary of WordNet-related metrics by Budanitsky and Hirst (2006)). In this work, we adopted Dao and Simpson's (2005) similarity metric between two sentences, using WordNet, which is a large, general-purpose lexical database of English. We chose this metric since it is simple, straightforward, and does not require a large corpus of statistics. This metric is based on Wu and Palmer's (1994) formula for comparing two words.

Definition 4 (linguistic similarity): The linguistic similarity between two method components $MC_1$ and $MC_2$, $L_{sim}(MC_1, MC_2)$ is a number between 0 and 1, which reflects the similarity between their names. Formally expressed:

$$L_{sim}(MC_1, MC_2) = \frac{\sum_{i=1}^{m} \max_{j=1..n} t_{i,j} + \sum_{j=1}^{n} \max_{i=1..m} u_{i,j}}{m+n},$$

where:

$t_{j...m}$ is the name of $MC_1$ (i.e., $MC_1.n$) and $u_{j...n}$ is the name of $MC_2$ (i.e., $MC_2.n$)

$m$ and $n$ are respectively the numbers of words in $MC_1$ and $MC_2$ names.
\[ l_{ti,uj} = \frac{2+N_3}{N_1+N_2+2+N_3} \] is Wu and Palmer's formula for comparing two words (see Figure 3).

As an example, consider the two processes represented in Figure 2, "Requirements Extraction" and "Manage User Stories." Following the calculations in Table 1, the linguistic similarity between these method components is:

\[ L_{sim}("Requirements Extraction", "Manage User Stories") = \frac{(0.54 + 0.5) + (0.5 + 0.54 + 0.46)}{2 + 3} = 0.51. \]

LCS is the least common super-concept of \( t_i \) and \( u_j \) in WordNet. 
\( N_1 \) is the number of nodes on the path from \( t_i \) to LCS in WordNet. 
\( N_2 \) is the number of nodes on the path from \( u_j \) to LCS in WordNet. 
\( N_3 \) is the number of nodes on the path from LCS to the root in WordNet.

Figure 3. Calculating the similarity between terms that are hierarchically related using Wu and Palmer's formula

<table>
<thead>
<tr>
<th></th>
<th>Manage</th>
<th>User</th>
<th>Stories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>0.5</td>
<td>0.54</td>
<td>0.46</td>
</tr>
<tr>
<td>Extraction</td>
<td>0.4</td>
<td>0.5</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Table 1. The linguistic similarity between "Requirements Extraction" and "Manage User Stories"

Note that, although the intentions of “Manage User Stories” and “Requirements Extraction” are alike and they may be used in the same early development phase of eliciting and analyzing requirements, their linguistic similarity is relatively low, and thus additional similarity metrics are required.

Meta-Informational Similarity

The various method components are annotated with the roles they play in the context of method engineering. Method components that have the same role are more similar than method components that bear different annotations. As an example of this claim, consider processes, tasks, and techniques, which are all work units. As opposed to a process, which is a large-grained work unit, tasks and techniques are small-grained work units: tasks are conceptual work units that focus on what must be done in order to achieve given purposes,
while *techniques* are technical work units that refer to *how* to achieve these purposes (ISO, 2007). We would like to be able to say, for example, that two processes with similar characteristics (e.g., name, structure, and behavior) are more similar than a process and a task that have the same goal.

Like the linguistic similarity, the meta-informational similarity is calculated using Wu and Palmer's formula. However, here the formula is calculated on the hierarchy of method engineering concepts, as derived from standard frameworks, such as ISO/IEC 24744.

**Definition 5 (meta-informational similarity):** The *meta-informational similarity* between two method components, $MC_1$ and $MC_2$, $MI_{sim}(MC_1, MC_2)$, is a number between 0 and 1 that reflects the similarity between the method engineering roles of the compared method components. Formally expressed:

$$MI_{sim}(MC_1, MC_2) = \frac{2\cdot N^3_i \cdot N^3_j}{N^1_i + N^2_i + 2\cdot N^3_i},$$

where $MC_i.r, i=1, 2,$ is the role of method component $MC_i$.

In other words, $MI_{sim}(MC_1, MC_2)$ is Wu and Palmer's metric applied to the roles associated with $MC_1$ and $MC_2$, where only inheritance relations are taken into consideration for this calculation (in order to imitate a lexicon or a thesaurus). Note that $MI_{sim}(MC_i, MC_j)=1$ if and only if the two method components have the same role.

As an example of meta-informational similarity calculation, consider the partial hierarchy of ISO/IEC 24744, depicted in Figure 4. The main method engineering concepts, namely work products, producers, modeling units, stages, and work units, appear in this figure on the same level (the third). They are further divided into sub-concepts. Work units, for example, are specialized into processes, tasks, and techniques. In addition, since work products, producers, and modeling units are mainly structural in concept (triggering, providing inputs, or being produced as outputs of work units), an upper level that includes the distinction between structural and behavioral concepts is added, and the various method-engineering concepts are divided accordingly. The top element in this hierarchy is 'Method Engineering Concept'; it is added in order to create a single tree of concepts. According to the resultant tree, the meta-
informational similarity between a process and a task, for example, is 0.67, whereas the meta-information similarity between a stage and a process is 0.4. The meta-informational similarity between method components that belong to completely different branches of this tree (e.g., work units and work products) is 0.

![Figure 4. The top level hierarchy derived from ISO/IEC 24744](image)

**Structural Similarity**

The internal structure of method components, as well as their external relationships to other method components, can be used to measure their similarity. Method components that exhibit similar features and parts and are connected to similar method components are considered more similar than method components that do not share a similar structure. As noted, the structure of a method component in our model consists of three sets: exhibition, aggregation, and relation sets. For simplicity, we assume that the method component model includes no generalization-specialization (inheritance) relations. If this is not the case, a pre-processing in which all the structural and behavioral relationships are percolated to the specialized method components has to be performed prior to similarity calculation. The following definitions describe structural similarity calculation. These definitions use the notion of general similarity, which is defined later in Definition 9. If the models include cycles of relationships, then the formula must traverse the model a fixed number of steps or until a cycle is detected.

**Definition 6 (set similarity):** The *set similarity* between two sets of method components, $MCS_1 = \{MC_{i_1}\}$ and $MCS_2 = \{MC_{i_2}\}$, $SS(MCS_1, MCS_2)$, is a number between 0 and 1 that reflects the general similarity between the sets' members. Formally expressed,
\[ SS(MCS_1, MCS_2) = \frac{\sum_{i=1}^{\lvert MCS_i \rvert} \max_{j=1,\lvert MCS_j \rvert} G_{sim}(MC_{i1},MC_{j2}) + \sum_{j=1}^{\lvert MCS_j \rvert} \max_{i=1,\lvert MCS_i \rvert} G_{sim}(MC_{i1},MC_{j2})}{\lvert MCS_i \rvert + \lvert MCS_j \rvert}, \]

where \( MC_{i1} \in MCS_i \), \( MC_{j2} \in MCS_j \), \( \lvert MCS_i \rvert \) and \( \lvert MCS_j \rvert \) are respectively the numbers of method components in \( MCS_i \) and \( MCS_j \), and \( G_{sim}(MC_{i1},MC_{j2}) \) is the general similarity between method components \( MC_{i1} \) and \( MC_{j2} \).

**Definition 7 (structural similarity):** The structural similarity between two method components, \( MC_i \) and \( MC_2 \), \( ST_{sim}(MC_i, MC_2) \), is a number between 0 and 1, which is calculated as a weighted average of the set similarities between the method components' exhibition, aggregation, and relation sets. Formally expressed,

\[ ST_{sim}(MC_i, MC_2) = w_{exh} \times SS(MC_i.Exh, MC_2.Exh) + w_{agg} \times SS(MC_i.Agg, MC_2.Agg) + w_{rel} \times SS(MC_i.Rel, MC_2.Rel), \]

where \( MC_i.Exh \), \( MC_i.Agg \), and \( MC_i.Rel \) are respectively the exhibition, aggregation, and relation sets of method component \( MC_k \) \( (k=1,2) \), \( w_{exh} \), \( w_{agg} \), \( w_{rel} \) are respectively the weights assigned to the similarity between the exhibition, aggregation, and relation sets in the structural similarity, and \( w_{exh} + w_{agg} + w_{rel} = 1 \).

The structural similarity between "Requirements Extraction" and "Manage User Stories," depicted in Figure 2, is 0.56. This calculation is based on the aggregation sets only, since the exhibition and relation sets of these method components are empty. As can be seen, although the purpose of the two components is similar, in terms of eliciting requirements their structure is different, yielding a relatively low similarity value. Furthermore, since the different parts of these components are not detailed, the calculation of the general similarity between the parts takes into consideration only their linguistic and the meta-information similarity.

**Behavioral Similarity**

The behavior of work units and stages can be described through their interfaces, i.e., their inputs, outputs, and triggers. Method components that have similar inputs, outputs, and triggers may perform the same activity in different ways and, thus, may be considered as
alternative method components. The following definition refers to the calculation of the
similarity between method components based on their behaviors. Here again, the definition
implicitly uses the notion of general similarity, which is defined later in Definition 9. If the
models include cycles of triggers, then the formula must traverse the model a fixed number of
steps or until a cycle is detected.

**Definition 8 (behavioral similarity):** The *behavioral similarity* between two (behavioral)
method components, $MC_1$ and $MC_2$, $BH_{sim}(MC_1, MC_2)$, is a number between 0 and 1, which is
calculated as a weighted average of the set similarity between the behavioral methods' input,
output, and trigger sets. Formally expressed,

\[
BH_{sim}(MC_1, MC_2) = w_{inp} \ast SS(MC_1.Inp, MC_2.Inp) + w_{out} \ast SS(MC_1.Out, MC_2.Out) + \\
\quad w_{tri} \ast SS(MC_1.Tri, MC_2.Tri),
\]

where $MC_k.Inp$, $MC_k.Out$, and $MC_k.Tri$ are respectively the input, output, and trigger sets of
$MC_k$ (k=1,2), $w_{inp}$, $w_{out}$, $w_{tri}$ are respectively the weights assigned to the similarity between
input, output, and trigger sets in the behavioral similarity, and $w_{inp} + w_{out} + w_{tri} = 1$.

Assuming the process inputs, outputs, and triggers are of equal importance, the behavioral
similarity between "Requirements Extraction" and "Manage User Stories" is 0.7, which is
quite a high similarity level, potentially justifying considering the two processes as
interchangeable method components.

Weights and Suitability

As mentioned, the general similarity between method components is defined as a weighted
average of the relevant linguistic, meta-informational, structural, and behavioral similarities.

**Definition 9 (general similarity):** The *general similarity* between two method components,
$MC_1$ and $MC_2$, $G_{sim}(MC_1, MC_2)$, is a number between 0 and 1 that is calculated as a weighted
average of their linguistic, meta-informational, structural, and behavioral similarity metrics.
Formally expressed,

\[
G_{sim}(MC_1, MC_2) = w_{ling} \ast L_{sim}(MC_1, MC_2) + w_{meta} \ast MI_{sim}(MC_1, MC_2) + w_{str} \ast ST_{sim}(MC_1, MC_2) + \\
\quad w_{bhv} \ast BH_{sim}(MC_1, MC_2),
\]
where \( L_{\text{sim}}(MC_1, MC_2) \), \( MI_{\text{sim}}(MC_1, MC_2) \), \( ST_{\text{sim}}(MC_1, MC_2) \), and \( BH_{\text{sim}}(MC_1, MC_2) \) are respectively the linguistic, meta-informational, structural, and behavioral similarities, \( w_{\text{ling}} \), \( w_{\text{meta}} \), \( w_{\text{str}} \), and \( w_{\text{bhv}} \) are respectively the weights assigned to linguistic, meta-informational, structural, and behavioral similarities, and \( w_{\text{ling}} + w_{\text{meta}} + w_{\text{str}} + w_{\text{bhv}} = 1 \).

The structural and behavioral similarities are further defined as the weighted averages of the similarity between their constituent sets. The initial values of these weights are: \( w_{\text{ling}} = w_{\text{meta}} = w_{\text{str}} = w_{\text{bhv}} = 0.25 \) and \( w_{\text{exh}} = w_{\text{agg}} = w_{\text{rel}} = w_{\text{inp}} = w_{\text{out}} = w_{\text{tri}} = 0.33 \). However, they can be controlled by the method engineers, who are enabled to take into consideration both their own previous experiences and the characteristics of the various components in the method base. A high weight given to the linguistic similarity, for example, may indicate the existence of an established, relatively homogenous vocabulary for the different method components. Still, there may be similar method components, whose intentions are alike but whose calculated linguistic similarity is low. Furthermore, utilizing a general vocabulary in the form of WordNet may lead to skewed results. In this case, one should consider reducing the weight of the linguistic similarity or using a specific method engineering-related vocabulary. Increasing the weights of behavioral and structural similarities is advisable in cases where the method base includes detailed method components that specify the structure and the behavior of the various method components. In the example that we have used to introduce and explain the different similarity types, we have seen that the linguistic similarity between the method components "Requirements Extraction" and "Manage User Stories" is 0.51, their meta-informational similarity is 1, their structural similarity is 0.56, and their behavioral similarity is 0.7. If all these similarity types are given the same weights (i.e., \( w_{\text{ling}} = w_{\text{meta}} = w_{\text{str}} = w_{\text{bhv}} = 0.25 \)), the general similarity between these method components is 0.69. However, since these processes are derived from two different development paradigms (RUP and XP) that use different vocabularies, it may be reasonable to reduce the weight of the linguistic similarity (e.g., \( w_{\text{ling}} = 0.1 \) and \( w_{\text{meta}} = w_{\text{str}} = w_{\text{bhv}} = 0.3 \)). In this case, the general similarity between the two method components will increase to 0.73.
It is important to understand that these weights are set for the entire method base. However, when calculating the similarity between particular method components, whose models do not include certain sets relevant to similarity calculation, the weights are proportionally divided into the other (existing) sets. If, for example, \( w_{\text{exh}} = w_{\text{agg}} = w_{\text{rel}} = 0.33 \), but the compared method components have no exhibition sets, their structural similarity will be calculated assuming \( w_{\text{agg}} = w_{\text{rel}} = 0.5 \) (and \( w_{\text{exh}} = 0 \)).

Note that usually the method base used by a software development company is relatively fixed and rarely changes. Thus, defining and tuning the different weights should seldom be necessary. However, the software projects in which the company is involved may have a wide variety of characteristics, requiring a more frequent creation of situational methods. For each such project, a situation query has to be defined. The method component model can be used for this purpose. Figure 5, for example, specifies the following situation query as a "regular" method component: create a method component for eliciting project requirements, which is triggered by a systems analyst, receives the client's information and agreement as inputs, and produces requirement specifications. Furthermore, the retrieved method components should suit situations in which the level of flexibility to changes in the project requirements is low and the project duration is at least one year. Once the situation query has been specified, the approach retrieves an initial set of method components whose general measure of similarity to the situation query is the greatest. The absolute values of these general similarity metrics may, however, be low, requiring several method components to be composed. Furthermore, the method engineers can control the number of method components in this initial set and even specify that it should be the number of method components in the method base. The retrieved method components must satisfy the situational characteristics modeled in the situation query. The initial set of the retrieved method components is then extended using different composition operations, as described next.
**Figure 5. Specification of a situation query: (a) in OPM and (b) in a formal manner**

**COMPOSITION CAPABILITIES**

In order to create situational development methods, different method components have to be combined to create larger method components that will eventually become the requested situational development methods. We elaborate here on the structural composition of method components (both structural and behavioral) and on the behavioral composition of method components. For simplicity, we give an example of these different composition operations applied concurrently on two method components; however, these operations can be applied successively. Finally, we also refer in this section to the general composition algorithm and its complexity.

**Structural Composition**

The structural composition of method components can be applied to both structural and behavioral method components. It focuses on the structural aspects of the method components, i.e., their internal structure and their external relationships with other method components. We can differentiate between two structural composition operations: merging and generalization. *Merging* combines all the method components into a single one, unifying the exhibition, aggregation, relation, input, output, trigger, and situational characteristics sets of the method components and associating the unified sets with the resultant method component. When merging two method components and unifying their sets, the elements in these sets are also structurally composed, applying merge or generalization operations.
Generalization enables one to abstract away aspects that differ between method components. While the merging operation can be applied to "very similar" method components, i.e., method components whose general similarity is greater than a high similarity threshold \( \text{hTH} \), specialization can be applied to "similar enough" method components, i.e., method components whose general similarity is greater than a different similarity threshold \( \text{sTH} \) which satisfies \( \text{sTH} < \text{hTH} < 1 \). Table 2 summarizes the types and conditions of the structural composition operations. In this table, exhibition links are representative of all structural relations. However, they can also be replaced by other structural links.

**Table 2. Structural composition of method components**

<table>
<thead>
<tr>
<th>Composition operation</th>
<th>A representative model before the operation</th>
<th>A representative model after the operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Merging</strong></td>
<td><img src="image" alt="Diagram of merging operation" /></td>
<td><img src="image" alt="Diagram of merged model" /></td>
</tr>
<tr>
<td><strong>Conditions:</strong></td>
<td>Two structural method components, ( a ) and ( b ), can be merged if they are &quot;very similar,&quot; i.e., their general similarity is greater than a high threshold, ( \text{hTH} &lt; 1 ). Note: 1. Elements in the exhibition, aggregation, general relation, input, output, and trigger sets are further structurally composed, applying merging or generalization operations. 2. ( a ) and ( b ) may be represented by OPM objects (as exemplified in the representative models) or by OPM processes.</td>
<td></td>
</tr>
</tbody>
</table>

| **Generalization**    | ![Diagram of generalization operation](image) | ![Diagram of generalized model](image) |
| **Conditions:**       | Two structural method components, \( a \) and \( b \), can be generalized if they are "similar enough," i.e., their general similarity is greater than a certain threshold, \( \text{sTH} \), which is still smaller than \( \text{hTH} \) \( (\text{sTH}<\text{hTH}<1) \). The two constituent method components, i.e., \( a \) and \( b \), specialize the generalized composition "a and b Generalization." Note: 1. Elements in the exhibition, aggregation, general relation, input, output, and trigger sets are further structurally composed, applying merging or generalization operations. 2. \( a \) and \( b \) may be represented by OPM objects (as exemplified in the representative models) or by OPM processes. |
As can be seen, the name of the new structural composition that resulted from applying a merging or generalization operation, is derived from the constituents (i.e., the method components) and the operation: the merged method components are named 'MC\textsubscript{1} and MC\textsubscript{2}', while the generalized method components are named 'MC\textsubscript{1} and MC\textsubscript{2} generalization', where MC\textsubscript{1} and MC\textsubscript{2} are the names of the constituents. In any case, the role of the structurally composed method components is the least common super-element in the hierarchy of method engineering concepts, derived, for example, from ISO/IEC 24744. If, for instance, MC\textsubscript{1} is a 'process' and MC\textsubscript{2} is a 'task', then the role of the composed method component will be 'work unit'.

**Behavioral Composition**

Behavioral composition deals with combining two or more method components into one composite method component, taking temporal aspects into consideration. As such, behavioral composition can be applied on behavioral method components, i.e., work units. The five behavioral composition operations used in different development methods are: (1) alternative, in which the constituent behavioral components are substituted and perform similar tasks; (2) sequential, in which the constituent behavioral components perform consecutively; the preceding behavioral component results in (structural) method components that serve as inputs to the following behavioral component; (3) concurrent, in which the constituent behavioral components perform in parallel or independently; (4) iterative, in which the constituent behavioral components are repeated and in each cycle the produced products are detailed and refined; and (5) incremental, in which the constituent behavioral components are repeated and in each cycle new increments are added to the produced products. Table 3 summarizes the conditions for carrying out these operations, along with representative associated models for each operation. The composed method component can be represented, for example, via the 'Stage' concept from ISO/IEC 24744, which specifies the temporal aspect of development methods.
### Table 3. Behavioral composition of method components

<table>
<thead>
<tr>
<th>Composition operation</th>
<th>A representative model before the operation</th>
<th>A representative model after the operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative</td>
<td><img src="image1" alt="Alternative Diagram" /></td>
<td><img src="image2" alt="Alternative Diagram" /></td>
</tr>
<tr>
<td>Sequential</td>
<td><img src="image3" alt="Sequential Diagram" /></td>
<td><img src="image4" alt="Sequential Diagram" /></td>
</tr>
<tr>
<td>Concurrent</td>
<td><img src="image5" alt="Concurrent Diagram" /></td>
<td><img src="image6" alt="Concurrent Diagram" /></td>
</tr>
<tr>
<td>Iterative</td>
<td><img src="image7" alt="Iterative Diagram" /></td>
<td><img src="image8" alt="Iterative Diagram" /></td>
</tr>
<tr>
<td>Incremental</td>
<td><img src="image9" alt="Incremental Diagram" /></td>
<td><img src="image10" alt="Incremental Diagram" /></td>
</tr>
</tbody>
</table>

**Conditions:** Two behavioral method components $A$ and $B$ are considered as alternatives if the behavioral similarity between them is greater than a certain threshold. Note that the internal structure, names, and even roles of $A$ and $B$ may be different, but their similar interfaces indicate that they may have similar purposes.

- **Sequential**
  - Two method components, $A$ and $B$, can be sequentially composed if they are not alternatives and one produces an output that may be used as an input by the other. Formally expressed, $\exists a \in A.Inp, b \in B.Out$ such that the general similarity between $a$ and $b$ is greater than a certain threshold. In this case, $B$ should be executed before $A$.

- **Concurrent**
  - Two method components, $A$ and $B$, can be concurrently composed if they are not alternatives; they cannot be sequentially composed (in either order); and they are triggered in similar conditions (formally expressed, $\exists a \in A.Tri, b \in B.Tri$ such that the general similarity between $a$ and $b$ is greater than a certain threshold).

- **Iterative**
  - Two method components, $A$ and $B$, can be iteratively composed if they are not alternatives and similar method components have already been iteratively composed in the method base (formally expressed, $\exists A', B', \text{ and } P$ such that $A'$ is similar to $A$, $B'$ is similar to $B$, $A', B' \in P.Agg$, and $\exists C' \in P.Agg, C' \in P.Tri$). Note that $C'$ may be $A'$, $B'$, or a completely different method component.

- **Incremental**
  - Two method components, $A$ and $B$, can be incrementally composed if they are not alternatives and they produce parts of the same method component (formally expressed, $\exists A', B', a, b, \text{ and } c$ such that $A'$ is similar to $A$, $B'$ is similar to $B$, $a \in A.Out$, $b \in B.Out$, and $a, b \in c.Agg$). Note that $a$ and $b$ may be the same method component or different method components.
As can be seen in Table 3, the name of the resultant composite method component is derived from the operation (e.g., 'or' for alternatives and 'before' for sequential composition) and the names of the method components that it comprises. The aggregation set of the composite method component includes the method components that comprise it, whereas the input, output, and trigger sets of all the method components that comprise them are percolated to the new composite method component. Finally, the situational characteristics are also percolated from the constituent method components to the composite method component, but if different constituent method components have the same situational feature, \( f \), with different values, \( v_1, \ldots, v_n \), then the situational characteristics set of the composite method component will include the pair \( <f, 'v_1 \text{ or } \ldots \text{ or } v_n'> \).

As an example, consider the composition of the two method components depicted in Figure 2, "Requirements Extraction" and "Manage User Stories." As already noted, the level of behavioral similarity between these method components is relatively high, justifying their composition as alternatives. Figure 6 depicts the resultant composite method component. Note that merged and generalized inputs, outputs, and triggers are percolated to the wrapping stage (the alternative stage in this case), preserving OPM consistency rules between diagrams. "Requirements Document and User Stories Generalization," for example, is an output for Requirements Extraction, and both an input and an output for Manage User Stories. Thus, it is considered both an input and an output for the alternative stage. Furthermore, OPM enables implicit specification of aggregation and exhibition relationships: processes that appear within the frame of a process define aggregation relationships between the processes, whereas objects that appear within the frame of a process define exhibition relationships between the elements. Therefore, for Requirements Extraction, \( MC.\text{Exh} = \{"Selected \text{ Process}"\} \) and \( MC.\text{Agg} = \{"Process \text{ Selection}", "Requirements \text{ Extraction}", "Manage \text{ User \text{ Stories}}"\} \).

An examination of the five behavioral composition operations reveals that the last four can be classified as aggregation according to Becker et al.'s classification (2007), whereas the alternative operation can be classified as specialization: the two constituents (i.e., method components) can be generalized to a single method component.
As noted, the creation of a situational method requires the definition of a situation query, the retrieval of suitable method components from a method component base, and the composition of method components (denoted in the following as $MC_{SQ}$, $MC_{B}$, and $MC_{SQ}Set$, respectively). Algorithm 1 describes how the initial set of retrieved method components is achieved: for each element in the situation query, the algorithm associates similar method components from the method base, where similar elements are those whose general similarity is greater than a certain threshold ($sTh$).

**The General Composition Algorithm**

As noted, the creation of a situational method requires the definition of a situation query, the retrieval of suitable method components from a method component base, and the composition of method components (denoted in the following as $MC_{SQ}$, $MC_{B}$, and $MC_{SQ}Set$, respectively). Algorithm 1 describes how the initial set of retrieved method components is achieved: for each element in the situation query, the algorithm associates similar method components from the method base, where similar elements are those whose general similarity is greater than a certain threshold ($sTh$).
After creating the initial situation query similarity set (MC\textsubscript{SQ}Set), Algorithm 2 runs iteratively, creating compositions or variations of method components that fit the situation query better. The different structural and behavioral composition operations are applied on the initial situation query similarity set (MC\textsubscript{SQ}Set) and the method components base (MCB), in an effort to compose larger method components that suit the situation query in hand better, until no composition is applicable or the maximum number of iterations is reached. Note that any created composition (newMC) is checked against the situational characteristics specified by the method engineer in the situation query, MC\textsubscript{SQ}. Only if it satisfies these situational characteristics, as a whole composite method component, is it inserted into the resultant method components set, MC\textsubscript{SQ}Set.

An analysis of the complexity of the Create Situational Composition algorithm (Algorithm 2) shows that it contains four loops, two of which are bound by constants; the outermost loop is bound by an uppermost limit, whereas the innermost loop is bound by the number of possible composition operations, namely 7. Each of the two other loops runs at most |MCB| times, where |MCB| is the total number of structural and behavioral method components in the method base. Thus, the complexity of the Create Situational Composition algorithm is \(O(|MCB|^2)\). Prior to this algorithm, the Situation Query Similarity Set has to be built (Algorithm 1). The complexity of this preprocessing is also bound by \(|MCB|^2\). The

---

**Algorithm 1 - Build Situation Query Similarity Set**

Input: \(\text{MCB, MC}_{\text{SQ}}\)

Output: \(\text{MC}_{\text{SQ}}\text{Set}\)

\(\text{MC}_{\text{SQ}}\text{Set} = \emptyset\)

For each \(e \in \text{MC}_{\text{SQ}}\)

For each \(\text{MC} \in \text{MCB}\)

// Consider only method components that satisfy the situational characteristics required

// (in \(\text{MC}_{\text{SQ}}\)) from all method components that will compose the development method

// to be built

If (checkSituationalCharacteristics(\(\text{MC, MC}_{\text{SQ}}\)) = True) then

If (\(G_{\text{MC}}(e, \text{MC}) > s\text{Th}\)) then // \(s\text{Th}\) stands for similarity threshold

\(\text{MC}_{\text{SQ}}\text{Set} = \text{MC}_{\text{SQ}}\text{Set} \cup \{\text{MC}\}\)

End if

End if

End for

End for

Return \(\text{MC}_{\text{SQ}}\text{Set}\)
complexity of the overall automatic method components composition is therefore \( O(|MCB|^2) \).

Algorithm 2 - Create Situational Composition

<table>
<thead>
<tr>
<th>Input: MCB, MC(<em>{SQ})Set, MC(</em>{SQ})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: MC(_{SQ})Set</td>
</tr>
<tr>
<td>Do</td>
</tr>
<tr>
<td>MC(_{SQ})SetTemp = ( \emptyset )</td>
</tr>
<tr>
<td>For each ( MC \in MC(_{SQ})Set ) do</td>
</tr>
<tr>
<td>For each ( MC' \in MCB ) do</td>
</tr>
<tr>
<td>For each strategy ( \in { \text{seq, iter, inc, conc, alter, merge, generalize} } ) do</td>
</tr>
<tr>
<td>// possibleOperation and createCompositeMethodComponent follow</td>
</tr>
<tr>
<td>// Table 3 rules</td>
</tr>
<tr>
<td>If possibleOperation (MC, MC', strategy) then</td>
</tr>
<tr>
<td>newMC = createCompositeMethodComponent(MC, MC', strategy)</td>
</tr>
<tr>
<td>If (checkSituationalCharacteristics(newMC, MC(_{SQ})) = True) then</td>
</tr>
<tr>
<td>MC(<em>{SQ})SetTemp = MC(</em>{SQ})SetTemp ( \cup ) newMC</td>
</tr>
<tr>
<td>End if</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td>Delete MC from MC(_{SQ})Set</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td>MC(<em>{SQ})Set = MC(</em>{SQ})Set ( \cup ) MC(_{SQ})SetTemp</td>
</tr>
<tr>
<td>Until MC(_{SQ})SetTemp = ( \emptyset ) or the number of iterations reaches an uppermost limit</td>
</tr>
<tr>
<td>Return MC(_{SQ})Set</td>
</tr>
</tbody>
</table>

EVALUATING THE AUTOMATIC CREATION OF SITUATIONAL METHODS

In order to evaluate it, the proposed approach has been implemented as a tool called ADOM-ME. This tool enables the creation, modification, retrieval, and composition of method components that are stored in a method base. The experimental method base consisted of 244 structural and behavioral method components, organized in 15 OPM models, which were derived from four well known methods, namely XP (Beck and Andres, 2004), Scrum (Schwaber and Beedle, 2002), RUP (Kruchten, 2000), and OPF (Firesmith and Henderson-Sellers, 2001). The selection of these methods was not arbitrary: XP and Scrum represented "agile" methods, while RUP and OPF represented "traditional" ones. The method components were taken from different development phases, including business modeling, requirements analysis, software design, and software implementation. Having analyzed the characteristics
of this method base, we set the different weights and thresholds as described in Table 4; the reasons for setting these values are presented in the fourth column.

Table 4. The values assigned to the different weights and thresholds for the experiment

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Reasons for setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_{\text{ling}})</td>
<td>The linguistic similarity weight</td>
<td>0.15</td>
<td>The weight of the linguistic similarity was reduced with respect to the other weights, due to the variability of the method components in the method base (different development approaches, different development phases).</td>
</tr>
<tr>
<td>(w_{\text{meta}})</td>
<td>The meta-informational similarity weight</td>
<td>0.283</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{str}})</td>
<td>The structural similarity weight</td>
<td>0.283</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{bhv}})</td>
<td>The behavioral similarity weight</td>
<td>0.283</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{exh}})</td>
<td>The exhibition set similarity weight</td>
<td>0.4</td>
<td>The weight of the relation set similarity was reduced with respect to the other weights, since the method components used were rarely structurally related.</td>
</tr>
<tr>
<td>(w_{\text{agg}})</td>
<td>The aggregation set similarity weight</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{rel}})</td>
<td>The relation set similarity weight</td>
<td>0.2</td>
<td>Equal values were assigned to these weights, since no bias was detected.</td>
</tr>
<tr>
<td>(w_{\text{inp}})</td>
<td>The input set similarity weight</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{out}})</td>
<td>The output set similarity weight</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>(w_{\text{tri}})</td>
<td>The trigger set similarity weight</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>(hTh)</td>
<td>The high threshold for defining &quot;very similar&quot; components</td>
<td>0.60</td>
<td>These values were experimentally set, after running different queries on the method base and analyzing the tool's results.</td>
</tr>
<tr>
<td>(sTh)</td>
<td>The low threshold for defining &quot;similar enough&quot; components</td>
<td>0.55</td>
<td></td>
</tr>
</tbody>
</table>

We compared the results received from the ADOM-ME tool with the recommendations of humans for the same different situation queries in order to evaluate our retrieval and composition considerations. Due to difficulties in conducting such a comparison in industrial settings, because it would require considerable time, effort, and resources (Carver, Jaccheri, Morasca, and Shull, 2003), the human recommenders who participated in this comparison were a dozen advanced undergraduate (final year) and graduate information systems students at the University of Haifa, Israel, who took a seminar in method engineering in the academic year 2008-2009. The students were familiar with method engineering and situational method engineering concepts and activities, as well as with OPM. We further claim that the academic and industrial background of these students render them comparable to junior method engineers. The students were familiar with all the method components in the method base and had worked with them during the semester on different occasions. The rest of this section elaborates on the comparison settings, results, conclusions, and validity threats.
Comparison Settings

The tasks were divided into three categories, each of which consisted of two to four representative tasks. The first category consisted of comprehension tasks, which required the students to understand the intentions of the models of the different method components. The students were asked to justify the selection of specific method components for given situation queries. The second category consisted of retrieval tasks, which required the students to suggest suitable method components (from the method base) for given situation queries, and justify their choices. Finally, the third category consisted of composition tasks, which required the students to suggest appropriate compositions for sets of retrieved method components and given situation queries. The full questionnaire is provided in ADOM-ME web site (http://mis.hevra.haifa.ac.il/~iris/research/ME/). Table 5 details the classification of the different tasks in that questionnaire into the three aforementioned categories.

Table 5. Classification of the different questionnaire tasks into comprehension, retrieval, and composition categories

<table>
<thead>
<tr>
<th>Part</th>
<th>Question no.</th>
<th>Category</th>
<th>Question format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>Comprehension</td>
<td>For each listed method component, explain the reasons for its selection in the context of the given situation. Indicate all possible reasons.</td>
<td></td>
</tr>
<tr>
<td>A 2</td>
<td>Composition</td>
<td>Identify pairs of elements relevant to both &quot;Requirements Extraction&quot; and &quot;Manage User Stories&quot; that in your opinion indicate the relationships between these method components. For each pair, briefly explain the relationships and your reasons for composition.</td>
<td></td>
</tr>
<tr>
<td>A 3</td>
<td>Composition</td>
<td>Rank the level of suitability of the two method components to the situation in hand (1 indicates &quot;not suitable at all&quot; and 5 indicates &quot;very suitable&quot;). The objective of this question is to attain additional insights into questions A1 and A2.</td>
<td></td>
</tr>
<tr>
<td>B 1</td>
<td>Retrieval + Composition</td>
<td>Identify and mark the words in the given text that may influence the retrieval of relevant method components. The objective of this question is to aid situation query definition (for questions B2 and B3).</td>
<td></td>
</tr>
<tr>
<td>B 2</td>
<td>Retrieval</td>
<td>Identify the method components (from the method base) that are suitable for the given situation. For each component, indicate the reasons for your selection.</td>
<td></td>
</tr>
<tr>
<td>B 3</td>
<td>Composition</td>
<td>Which of the method components you retrieved (in question B2) should be incrementally or iteratively composed? Explain your choice.</td>
<td></td>
</tr>
<tr>
<td>C 1</td>
<td>Comprehension</td>
<td>Indicate the level of suitability of each listed component for the situation in hand (scale 1-5), and give the factors that you think led to its selection.</td>
<td></td>
</tr>
<tr>
<td>C 2</td>
<td>Composition</td>
<td>Suggest compositions of the components retrieved for question C1 that are suitable to the described situation. Refer to sequential and concurrent compositions only.</td>
<td></td>
</tr>
<tr>
<td>C 3</td>
<td>Retrieval + Composition</td>
<td>The situation has been changed &lt;description of the change&gt;. State which components you would pick in order to support this change. How would you integrate these components with the prior components? Refer to sequential and concurrent compositions only.</td>
<td></td>
</tr>
</tbody>
</table>
So that the results of the automatic tool would not influence the human recommenders (i.e., the students), the latter were not provided with the tool outcomes, but rather were required to make their suggestions independently, and justify each suggestion. Since this process is time-consuming, error-prone, and influenced by the students' knowledge and understanding of the development methods used, the tasks were very concrete and the students were given unlimited time to conclude their tasks.

Independently, we modeled each situation query related to these tasks and ran the ADOM-ME tool to obtain its suggestions of suitable method components or compositions. We limited the number of the iterations of the tool, so that the tool outcome would be comparable with the results of the manual processing of the students.

**Comparison Results**

In each question, the answers given by the students were sorted according to the number of students who provided them and the number of supplied justifications, whereas the outputs of ADOM-ME were presented as an ordered list of method components sorted according to their descending general similarity with the situation query. We then compared pairs of relevant student and tool responses, classifying each pair into complete match, partial match, or mismatch. A pair was considered as a 'complete match' when both its location in the ordered lists of answers and its justifications were alike (we used the different similarity metrics as "justifications" of the tool); the pair of responses was considered as 'partial mismatch' when either its location in the ordered lists of answers or its justifications were different; otherwise, i.e., when both its location in the ordered lists of answers and justifications were different, it was considered as 'mismatch'. Student or tool responses that had no counterpart were classified as mismatches. Table 6 summarizes the comparison results in each category, namely comprehension, retrieval, and composition, along with possible reasons for differences. These differences are elaborated next.
Table 6. The comparison results

<table>
<thead>
<tr>
<th>Category</th>
<th>Comparison Results</th>
<th>Reasons for Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehension</td>
<td>complete match</td>
<td>68%</td>
</tr>
<tr>
<td>partial match</td>
<td>9%</td>
<td></td>
</tr>
<tr>
<td>mismatch</td>
<td>23%</td>
<td></td>
</tr>
<tr>
<td>Retrieval</td>
<td>complete match</td>
<td>52%</td>
</tr>
<tr>
<td>partial match</td>
<td>11%</td>
<td></td>
</tr>
<tr>
<td>mismatch</td>
<td>37%</td>
<td></td>
</tr>
<tr>
<td>Composition</td>
<td>complete match</td>
<td>61%</td>
</tr>
<tr>
<td>partial match</td>
<td>11%</td>
<td></td>
</tr>
<tr>
<td>mismatch</td>
<td>28%</td>
<td></td>
</tr>
</tbody>
</table>

**Comprehension Results.** In 77% of the cases, the students correctly comprehended the method component models, providing responses that were similar to those of the tool. Only in 23% of the results was a complete mismatch between the answers of the students and of the tool detected. An analysis of the students' responses and the tool's outputs revealed four main reasons for these mismatches. First, the level of detail of the models in the method base influenced the performance of the students. The "Requirement Extraction" process, for example, had a very detailed model, which spanned across three different diagrams. As a result, the general similarity that was calculated by the tool between this method component and the situation query was relatively low, but the students ranked this method component high due to the similarity between its behavioral aspects and the given situation. This outcome may indicate that the importance of behavioral similarity in the creation of situational methods is greater than that of the structural aspects, justifying a reduction in the structural weight \( w_{str} \) with respect to the behavioral weight \( w_{bvh} \).

Secondly, the students referred to the behavioral similarity between structural method components, especially those of producers. They took into consideration the behavioral
components that are triggered by the producers, while measuring the similarity between the producers. This outcome may raise a question concerning our decision to refer to behavioral aspects of behavioral components only, and thus suggests a subject for future research.

Thirdly, we detected differences that we believe originate in the differences in the nature of the work (manual vs. automatic). While the students had to establish their decisions manually in a large method base, the tool did so automatically and systematically, taking into consideration all the method components in the method base, and thus its results were more consistent and more explicable.

Finally, in a few cases, we found different justifications for the same decision, e.g., the tool made recommendations mainly according to linguistic considerations while the students focused on behavioral justifications. The reason for this outcome may be the fact that English was not the first language of the students and they could not recognize the linguistic similarity between the method component names in these cases.

Despite these differences, when the students were asked to give an overall recommendation (question 3 in Part A), they reached conclusions similar to those of the tool.

**Retrieval Results.** In 63% of the cases, the tool and the students retrieved quite similar method components or provided similar justifications for their selection. However, in 37% of the cases, mismatches were found. The higher percentages of mismatches in this category may be attributed mainly to differences in the nature of the work (manual vs. automatic). The tool retrieved five suitable method components that were not mentioned in the students' responses at all. The linguistic and behavioral similarity metrics of these method components justify their retrieval for the given situation queries. The students, who were required to search the entire method base manually, seemed to miss these method components: two out of the five method components were sub-processes of method components that had already been discussed in another part of the questionnaire, while two other method components handled requirements that satisfied only a small part of the goal of the situation query. Another finding is that the students relied on previous knowledge that was not explicitly specified in the
models and thus could not be used by the tool. This finding points to the importance of including detailed models in the method base in order to obtain relevant situational methods.

**Composition Results.** In 72% of the cases, the tool and the students recommended similar compositions, while 28% of the cases were classified as mismatches. An analysis of the justifications of the tool and the students showed that the differences in ranking compositions that involve work products occurred when the linguistic aspect of these work products reflected no similarity and the specification of the method components spanned across different diagrams. This outcome is in line with our observation that throughout the questionnaire the students searched mainly for compositions of behavioral method components and only merged similar work products that they encountered.

The main reasons for the differences in composing work units are the detail level of the models and the number of similar associated work products and producers. According to our analysis of incremental vs. iterative compositions (Question 2 in Part B of the questionnaire), more than half of the students' recommendations on incremental compositions (6 out of 10) were in line with the rule implemented for incremental compositions in the tool. Two out of the four recommendations that did not follow this rule were based on previous knowledge that was not explicitly specified in the models, whereas each one of the other two method components was retrieved by only one student. Regarding iterative compositions, the students based their recommendations mainly on linguistic aspects (expressed in the method component names or roles), rather than on the method components structure. While this strategy may work well on pre-defined method components in a relatively small method base, it may be problematic in the general case. Nevertheless, this result raises the question whether linguistic aspects should be examined in addition to the iterative composition rule – an issue that should be investigated in the future.

An examination of the recommendations of the student and the tool concerning sequential vs. concurrent compositions (Questions 1 and 3 in Part C) revealed that almost all the recommendations concerning the same method components yielded the same composition
operation. The two exceptions were method components that could be concurrently composed, but (some of) the students tried ordering them into a sequential composition, utilizing their "previous knowledge." In addition, the tool found one to two compositions that were not found by the students, probably due to the manual nature of their work, whereas the students suggested several compositions that were not detected by the tool, based on their "previous knowledge."

**General Conclusions and Threats to Validity**

Based on the comparison above, ignoring differences that we believe were due to the manual work of the students, we can draw four general conclusions. First, the different similarity metrics we have identified for comparing method components, namely, linguistic, meta-informational, structural, and behavioral similarities, correlated in most cases with the students' selection and justification in the same situations. Differences were discovered mainly when: (1) students used previous knowledge that was not explicitly modeled in the method base, and (2) students referred to behavioral aspects while evaluating the similarity between structural method components, especially producers. Secondly, the rules for sequential, concurrent, and increment composition were in line with the students' recommendations. Nevertheless, the students offered more compositions than were suggested by the tool, following a more flexible policy for composition, based mainly on their previous knowledge. Thirdly, the students recommended iterative composition, considering linguistic aspects alone. This may be due to a confusion between incremental and iterative compositions and because they did not have a clear criterion for distinguishing between these types of composition. Finally, the students seem to comprehend the method components model.

The main threats to the validity of the results can be divided into threats to internal, construct, external, and conclusion validity (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén, 2000). Regarding internal validity, we compared the manual work of humans (students) with the automatic and systemic work of ADOM-ME. It is well known that manual work is more time-consuming and error-prone, and less consistent. Thus, we took the following actions: (1)
the scope of each task was limited and the goal was very concrete; (2) we limited the number of iterations of the tool, preventing it from examining complicated cases (such as compositions of three method components); and (3) we required justifications for each selection (of a method component or a composition) and we used this information in addition to the ranking.

The representativeness of the method components can also be questioned. We chose four well-known methods, two of which are classified as 'agile' (XP and Scrum) and the others as 'traditional' (RUP and OPF). From each method, we selected both structural and behavioral method components that had been developed by different individuals in various levels of detail. Nevertheless, further evaluations of additional method components and approaches have to be made.

The main external threat to validity stems from the type of subject who performed the tasks. In our case, the subjects were advanced undergraduate (final year) and graduate information systems students who were familiar with method engineering and situational method engineering. However, as already noted, we claim that their academic and industrial background makes them comparable to junior method engineers. Furthermore, we made an effort to ensure that staff members would accompany the students throughout the seminar course and provide detailed guidance for their tasks. Nevertheless, an additional evaluation of the approach is required; in particular, method engineers should be requested to provide feedback concerning the tool's outcome for the different situations.

Regarding construct threats to validity, the tasks in this comparison were objective, such as retrieving and composing method components according to pre-defined rules, and subjective, such as ranking. However, the subjects were requested to justify each subjective task; their ranking correlated with the provided justifications.

Finally, the small number of participants in the comparison (about a dozen), which prevented us from carrying out a statistical analysis, may raise concerns about the validity of the conclusions. To overcome this concern, we provide the raw results, as well as detailed...
RELATED WORK

Retrieving method components and composing them into situational development methods are time-consuming, error-prone, and subjective tasks. To make these tasks more systemic, different heuristic methods have been proposed for determining similarity metrics between method components. These approaches generally measure the closeness between entities of different conceptual schemas by evaluating the common properties and links with other entities. Ralytė and Rolland (2001), for example, define semantic and structural criteria for measuring the similarity between method components. The similarity between product-related components is measured using name affinity metrics and links calculation. The process models of the method components are represented as maps that provide a non-deterministic ordering of intentions and strategies. When measuring the similarity between these maps, the semantic affinity and structural similarity between intentions and sections (i.e., intention-strategy-intention edges) are taken into consideration. Further, Ralytė and Rolland suggest two strategies for assembling method components: association and integration. Association is applicable for method components that correspond to different functionalities and do not have common elements, while integration is suitable for method components that have similar engineering goals, but provide different ways to satisfy them. In addition to these strategies, extension-based and paradigm-based operations can be performed. In comparison to this work, our approach refers to behavioral and meta-informational considerations when retrieving relevant method components, and allows additional composition possibilities, namely concurrent, iterative, and incremental compositions.

The Situation-Success-Scenario (S3) model (Harmsen, Lubbers, and Wijers, 1995) lists aggregated situation factors, performance indicators, and scenario aspects. The method components are then selected in four steps: (1) determination of the project goal, (2) determination of a preliminary scenario, (3) adaptation of the preliminary scenario, resulting
in a project scenario, and (4) selection of the method components containing the aspects of the project scenario. All tasks are performed manually by method engineers or project managers.

Mirbel (2006) provides ways to qualify each method component using meaningful keywords for the critical organizational, technical, and human-related aspects. This is done in order to allow each project to share its best practices with other projects without imposing a unique organization-wide method on all of them. Similarity metrics are proposed for measuring the distances between a given situation, which is specified as a set of at least one pertinent keyword and forbidden keywords, and a method component reuse context, depicting the situations to which the method component is applicable.

Kornyshova et al. (2007) introduced multi-criteria techniques for improved guidance in the retrieval of method components. These techniques allow the selected method components to be prioritized according to multiple criteria in order to guide the method engineers in the final selection process. As opposed to mono-criterion approaches, this approach allows a more in-depth analysis of problems that takes various aspects into consideration. Two groups of techniques are used: outranking and weighting. Outranking techniques are inspired by the theory of social choice and serve as a means for indicating the degree of dominance of one alternative over the other. Weighting techniques assign weights to the different decision criteria. A project characteristics typology is suggested in order to identify all the relevant critical aspects. The final selection is then realized using similarity metrics that measure the distances between different work products and various work units. The approach does not take into consideration the structure and behavior of the method components.

Pirro (2009) compares several approaches for assessing similarity, according to the source of information they exploit. Ontology-based approaches (e.g., Rada, Mili, Bicknell, and Blettner, 1989; Hirst and St-Onge, 1998) assess semantic similarity by counting the number of nodes or edges separating two concepts. These approaches suffer from the limitation that, in order to perform properly, they require consistent and rich ontologies. Information theory approaches (e.g., Lin, 1998; Rodríguez and Egenhofer, 2002) exploit the notion of information content,
defined as a measure of the informativeness of concepts and computed by counting the occurrence of words in large corpora. The drawbacks here are that these approaches require time-consuming analysis of corpora, and that Information Content values usually depend on the kind of corpora being considered. Hybrid approaches (e.g., Li, Bandar, and McLean, 2003) combine multiple information sources. They typically require some "configuration knobs" to be adjusted (e.g., weights used to set the contribution of each information source).

**SUMMARY AND FUTURE WORK**

As the variety of development projects increases, so does the importance of approaches that support the retrieval and composition of method components for use in given situations. However, at present most approaches in this area support manual work in which retrieval is basically structural and composition is usually sequential. Furthermore, the resultant situational methods are not recorded, managed, and potentially cataloged for future 'similar' situations. The objective of the approach suggested here is to increase the spectrum of semi-automatically created situational methods, taking behavioral aspects into consideration, and introducing additional assembling operations. The similarity between the method components is measured according to linguistic, meta-informational, structural, and behavioral considerations. Composition is enabled using two structural operations (merging and generalization) and five behavioral operations (sequential, concurrent, incremental, iterative, and alternative compositions). The resultant situational methods are inserted into the method base for future use in similar situations. Nevertheless, the creation of situational methods is not a completely automatic process and may require the involvement of method engineers to bridge the gaps between method components that are only similar enough, but not identical.

We have developed an automatic tool for the creation of situational methods and compared the outcomes of the tool with the evaluation of information systems students. The analysis of the results of this comparison highlights the importance of behavioral aspects in the retrieval and composition of method components, especially when different development paradigms are involved. In such cases, the vocabulary used for modeling the method components is
diverse, the linguistic similarity value may be low, and the behavioral and structural aspects become dominant. For the purpose of retrieving reasonable method components and compositions, it is important to detail the structure and behavior of method components.

The proposed approach is adaptable and can be extended. In particular, the weights of the different similarity metrics, as well as the different thresholds, can be modified according to the diversity of the available method components. Nevertheless, the determination of the different parameters (weights and thresholds) needs further investigation. Moreover, guidelines for the determination process of these parameters should be developed, and the involvement of a team of method engineers in this process should be considered.

Further research is required to investigate the different similarity metrics and composition operations, and to consider new kinds of similarity and composition operations. In particular, the following questions can be examined. Should behavioral aspects be taken into consideration when measuring the similarity between structural method components, such as producers and work products? Should linguistic aspects be taken into consideration when applying iterative (and other) composition operations? Should the indirect context of method components be taken into consideration for retrieval and composition purposes? In addition, experimental studies of the approach in industrial settings are required. This way, the tool outcome can be examined by experienced method engineers and their feedback can be used to improve the tool and its framework.

REFERENCES


Ralyté, S. Brinkkemper, & B. Henderson-Sellers (Eds.), *Situational Method Engineering: Fundamentals and Experiences* (pp. 79-93). Germany: Springer.


Author Biography

Anat Aharoni is a faculty member at the department of Information Systems Engineering, The School of Engineering, Kinneret College, Israel and a lecturer at the University of Haifa, Israel. She received her BSc degree in Computer Science from the Tel Aviv University, Israel in 1989. She obtained an MSc degree in 2001 from the department of education in technology and science, Technion, Israel Institute of Technology, and a PhD in 2010 in information systems, from the University of Haifa, Israel. Her research interests include method engineering and methodologies for information systems development. She had worked as a programmer and project manager in different software companies focusing on development of information systems.

Iris Reinhartz-Berger is a faculty member in the Department of Information Systems, University of Haifa, Israel. She received her BSc degree in applied mathematics and computer science from the Technion, Israel Institute of Technology in 1994. She was awarded an MSc degree in 1999 and a PhD in 2003 in information management engineering from the Technion, Israel Institute of Technology. Her research interests include conceptual modeling, analysis and design of information systems, software product line engineering, and method engineering. Her work has been published in international journals and conferences and she has chaired a series of Domain Engineering workshops and presented tutorials on domain engineering and variability management.