Scientia Iranica B (2015) 22(4), 1616–1624

SCIENTIA IRANICA

Sharif University of Technology
Scientia Iranica
Transactions B: Mechanical Engineering
www.scientiairanica.com

Meta software engineering for information system development projects

D. Samadhiya* **and W.C. Chang**

Department of Technology Management, Chung Hua University, 707, Sec2, Wufu Rd, Taiwan 30012.

Received 22 May 2014; received in revised form 27 October 2014; accepted 9 May 2015

KEYWORDS
Process engineering;
Structural method;
Process functionality.

Abstract. There is increasing demand for sophisticated software engineering processes in today's software systems. In this regard, however, there is a multitude of different processes and each has various advantages and disadvantages some of which relate to the problem domain or in the context of development. Computer software development processes have to pass through many scenarios to be completed. There are many ways to solve a single problem in software development. Sometimes, in structural engineering, the developer is not able to decide which process will suit a particular problem. It can be said that selecting a suitable process is a big issue in structural process software engineering, and the problem of selecting a good candidate method is a big issue in structural process engineering. The solution of such kinds of problem can be found in the work to be done and the task to be performed by the operational process rather than by the process structure. This paper introduces the notion of process operationality and proposes 'process architecture' to represent this operationality. Thus, Structural Process Engineering (SPE) becomes Operational Process Engineering (OPE). Operationally close process architecture is selected, adapted, enhanced, and restricted as needed. The task of construction consists of putting together process features and structuring the new process.

1. Introduction

Nowadays, information systems are the basis of many activities in the real world, and due to requirements, the complexity of these information system based systems is increasing. On the other hand, development time is reducing and new processes are being introduced constantly. As a consequence, the traditional rigid IS engineering processes are inadequate to provide the necessary support in new IS developments. New methods that are more flexible and better adapted to the situation of every IS development project must be constructed. It is said that "Process engineering in the field of information systems is the discipline to construct new processes from existing processes" [1]. These focus on the design, construction and evaluation of processes, techniques and support tools for information system development [2]. Numerous development processes, based on a variety of paradigms, have been proposed over the years. Of these, very few have been successfully applied to the development of computer based systems.

Since their introduction, various life cycle models and specific supporting techniques have played an important role in building software systems [3]. More recently, the topic of software processes has received increased attention from the software community. A software design approach called "Evolution of Software Processes" is based on the emerging view that software processes - like software - also need to evolve, lest they become obsolete [4,5]. The aim of this evolution is to fulfill the needs of the people who perform the process,

*. *Corresponding author.*
E-mail addresses: samadhiya.durgesh@gmail.com (D. Samadhiya); earnest@chu.edu.tw (W.C. Chang)

and the developmental and organizational goals to be achieved. Another recent software design paradigm that can be seen as a generalization of software process evolution is process engineering. While there is a great overlapping of process engineering and process evolution activities, there are also some important divergences. Basically, process evolution is oriented more towards the improvement of existing processes and process engineering more towards the construction of new methods or processes.

Ralyte suggests that process engineering is facilitated if the goal of the process can be determined. In this regard, the following questions have been raised [6]:

- How can assurances be provided that the process to be enhanced, extended, or restricted is a good candidate process?
- What are the chances that at the process engineering intention stage, the process will have to be discarded because its adaptation is very difficult?
- Should not further exploratory work be undertaken before committing to setting up process adaptation intentions?

The solution to these questions is in structural meta software engineering, but, the problem still arises that no process is best in all the structures.

Figure 1 defines a process re-engineering process model that provides guidelines to re-engineer an existing information system development process into a reusable process. Figure 1 summarizes our process re-engineering approach. In this paper, Section 1 includes the introduction of the theme. Sections 2 and 3 explain the brief terminology of structural meta software engineering and operational meta software engineering. Section 4 illustrates the motivation and contribution of this paper, and Section 5 shows the preliminary results of this contribution. The conclusions are found in Section 6 of this paper.

2. Structural meta software engineering

Process Engineering (PE) and Structural Meta Software Engineering (SMSE) focus on formalizing the use of processes for systems development. The broader term, process engineering, is defined as an engineering discipline that designs, constructs and adapts processes, techniques and tools for systems development; a definition analogous to the IEEE definition of software engineering [7]. In the real world, many information systems development processes exist, but no method is best for all situations. Structural meta software engineering has been proposed for developing or tailoring information system developing processes for specific structural projects [8]. Structural meta software engineering is "directed towards the controlled, formal and computer-assisted construction of structural process out of process fragments" [9]. A *structural process* is an information system engineering process tailored and tuned to a particular structure. Structural processes are engineered in a formal and computer-assisted manner out of standardized and proven building blocks stored in an electronic data base. These building blocks are called process storage and a process storage is a description of an information system engineering process, or any coherent part thereof [10,11].

Figure 2 shows a structural meta software engineering process. In the introduction of process engineering, we discussed the development towards standardized information system engineering processes. Despite various attempts regarding the "unified" or "universal" process, it is concluded that there is no process which is best in all situations [12-16]. To anticipate this problem, various approaches have been proposed, which are positioned in the so-called "Struc-

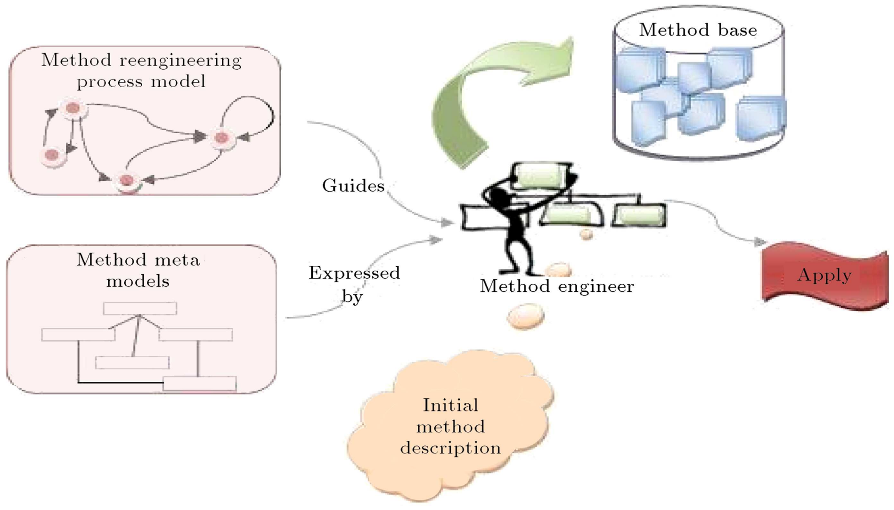

Figure 1. Process engineering approach.

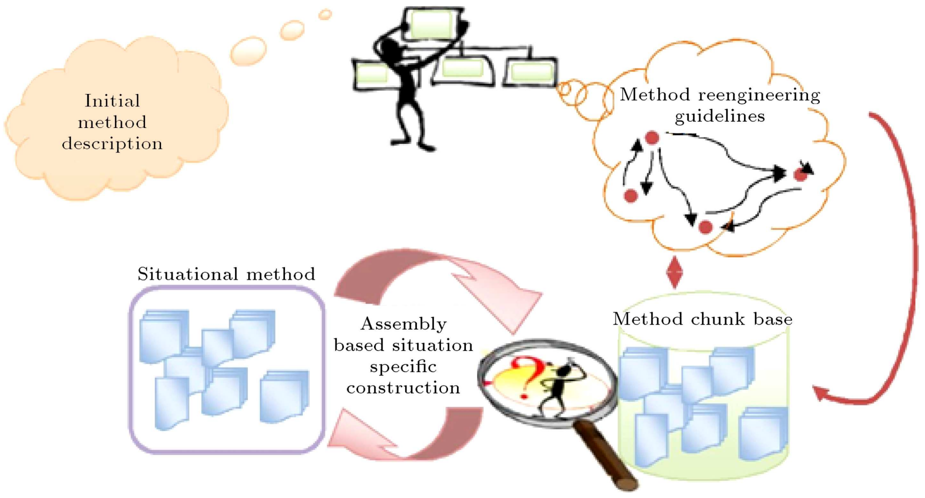
Initial method description
Method reengineering guidelines
Situational method
Assembly based situation specific construction
Method chunk base

Figure 2. Structural meta software engineering process.

tural Process Spectrum" [17]. Despite the large number of proposals that exist, there is some dissatisfaction with the notion of structure. Bucher is concerned about the poor understanding of the notion of a structure [18], so, there is a need to find a way to reduce the number of possible situations [19].

3. Motivation and contribution

As previously mentioned, Ralyte suggests that process engineering is facilitated if the intention of the process can be determined. In this regard, the following questions have been raised [20]:

- How can assurances be provided that the process to be enhanced, extended, or restricted is a good best possible process?
- What are the chances that at the process engineering intention stage, the process will have to be discarded because its adaptation is very difficult?
- Should not some further exploratory work be undertaken before committing to setting up process adaptation goals?

The solution of these questions can be found in the work to be undertaken, and the task to be performed rather than the structure of a process itself. Every computer software process having a full cycle consisting of requirements, design and construction engineering and current state of the art in structural process engineering addresses the construction engineering phase. The other two stages help us to undertake further 'exploratory work' referred to by Ralyte. At the design stage, we introduce the notion of process operationality and propose 'process architecture' as an abstraction of this operationality. Thus, process engineering becomes Operational Meta Software Engineering (OMSE). Operationally close process architecture is selected, adapted, enhanced, and restricted as needed. The task of construction handles the putting together of process features and of structuring the process. Thus, we see a difference between structural process engineering and operational process engineering. Last, but not least, it is necessary to explain requirement engineering, which is upstream to design engineering. Here, we introduce the notion of a process goal. Once processes with similar goals to those being engineered are found, a menu of processes to be adapted, enhanced, and restricted is determined. This is further refined in the design stage, where architecture matching occurs. Again, a residue of processes is found, and, at this stage, the architecture of the new process emerges as a set of connected functions. Finally, this architecture is engineered from building blocks taken from the residue.

It can be noted that progressive selection in the requirements and design stages include:

- The potential to assure that the method to be enhanced, extended, or restricted is a good possible process;
- Rejection of inappropriate processes; those having dissimilar goals and dissimilar architectures are rejected before the actual construction stage, which, therefore, reduces the possibility of rejection;
- Enough exploratory work is done before committing to process features.

Since project needs vary with projects, and projects vary in their characteristics, the development of methods may require specific adaptations. Therefore, an engineering technique for this is required [21]. We can now state the aim of the thesis. We wish to move to goal process engineering in order to explore the context of structural process engineering more fully. As a result, process selection for adaptation shall be more appropriate and will assure that structural process engineering is progressing purposefully. It will considerably reduce the chance of process rejection at later stages. For this task, a 3-stage life cycle for goal process engineering is introduced.

In this life cycle, we introduce a process architecture matching phase that corresponds to our view of operational process engineering. Then, the notion of process architecture is explained through a meta-model, and a set of operations is defined that enables architecture matching. The two layers; design and construction engineering, constitute the functional level of process engineering. Once this is developed, we expect to put an intentional level on top of the operational level, which will further raise the abstraction in terms of which process requirements will be expressed.

4. Preliminary results

4.1. Process development life cycle

As shown in Figure 3, we have developed a process development life cycle for development. The requirements for the engineering stage consist of intention matching. First, the goal of the process, To Be, is elicited. The goal matching process uses matching synonyms to identify intentionally similar processes that reside in the process storage. These processes become possible methods for the second stage of this cycle.

In the design engineering stage, the process engineer retrieves the architecture of each possible process from process storage. The subset of these components and inter-relationships that best meets the broad operational needs of the process To-Be is selected. Such selections are made from all possible processes and are synthesized together into the architecture of the desired process. In the construction stage, the architecture is populated with instances of the process features needed in the process.

4.2. Process architecture

We have defined process architecture as an abstraction of the process that identifies its components and inter-relationships to highlight the externally visible operationality of the process. We use the class abstraction as a way of formally defining process architecture, as follows:

$$\text{Process architecture} = \{\text{process}|\text{process performs function F}\}.$$

The name, process architecture, reflects the operation performed by the class of processes abstracted in the architecture.

The process architecture meta-model can be summarized as an architecture implemented as process organization, and this organization shows the features of the process and their inter-relationships. An architecture can be atomic or complex, and architectures can be related to one another by links. These links form a successor or predecessor relationship between architectures. Links are labeled by their execution properties, Urgency and Necessity, respectively.

4.2.1. Process architecture matching process

In this design, the process engineer retrieves the process architecture of each possible process and these possible processes are obtained from the goal level, as shown in Figure 3. The operational needs of this process are matched with the operational expressions of the candidate processes and new desired process architecture is made. In this process, only those that are useful for the architecture should be selected and those that are useless should be refused. The following operations have been proposed to do this:

i) Given a named architecture, rename it;

ii) Create a new architecture;

iii) Delete an existing architecture;

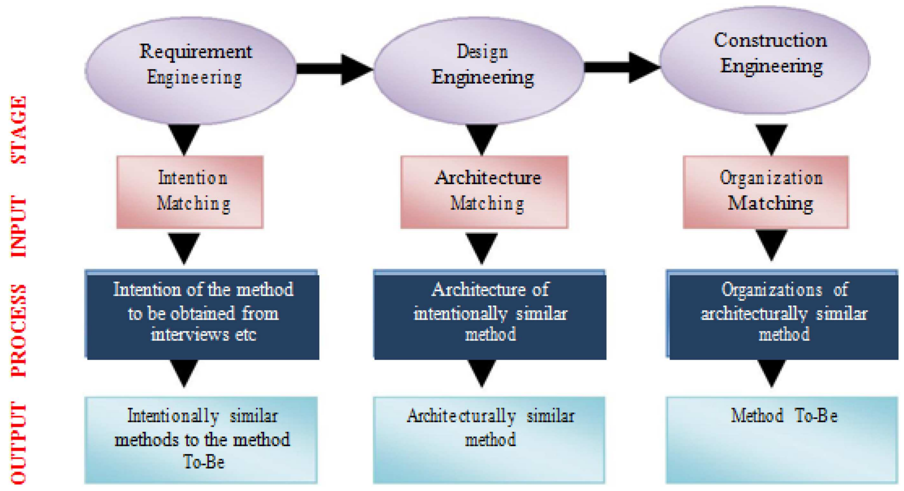
STAGE
INPUT
PROCESS
OUTPUT
Requirement Engineering
Design Engineering
Construction Engineering
Intention Matching
Architecture Matching
Organization Matching
Intention of the method to be obtained from interviews etc
Architecture of intentionally similar method
Organizations of architecturally similar method
Intentionally similar methods to the method To-Be
Architecturally similar method
Method To-Be

Figure 3. The process development life cycle.

iv) Nest, N architectures within another one;

v) Un-nest architectures so that a nested architecture becomes visible at a higher level of nesting;

vi) Change a link type;

vii) Make a sequence of architectures by introducing an edge between them and defining their link type. Link Type (LT) is used to elaborate the nature of the relationship between operational processes. It consists of two properties, Urgency and Necessity.

viii) Eliminate a sequence.

4.2.2. Process architecture meta-model

For better understanding and to make the notion of a process framework more precise, we have developed a meta-model for it that identifies the key concepts of a process framework and their inter relationships. It also forms a basis for deriving the graphical representation of a process framework in this model. One framework is related to other frameworks, and the relationship between frameworks is defined in this meta-model. The link type is an attribute of this relationship, which takes on values from the set {IM, IC, DM, DC} that is: 'Immediate Must', 'Immediate Can', 'Deferred Must', and 'Deferred Can'. We have shown that a process framework can be implemented as one or more process organizations. We found an 1:N relationship between a process framework and process organisation. Table 1 shows all types of link.

4.3. Operational process engineering

It is now time to explain the difference between structural meta software engineering and operational meta software engineering, as proposed. In the fragment based structural meta software engineering proposal [22], there are two fundamental elements:

a) Products and their structures;

b) Procedures and their execution order to develop the products.

Products and their structure show that the interest is in the structure of the products. Similarly, since the structure of a process is largely determined by the order of execution, the interest is in the process structure. Therefore, we can conclude that structural process engineering is centered round the structural aspects of processes. This focus on engineering the structure of processes de-emphasizes what the process does, and what task it is good for. In fact, determination of whether or not the process structure can carry out the project task at hand is based on the experience of the process engineer.

Operational process engineering puts the process structure subordinate to process operationality. OMSE asks for an explicit determination and representation of process operationality in the form of process architectures. It is only after the architecture has been built that the issue of process structure is to be considered. In this sense, SMSE occupies the downstream, construction engineering stage of our life cycle.

Table 1. Types of links.

Link type	Urgency	Necessity	Abbreviation
1	Immediate	Must	IM
2	Immediate	Can	IC
3	Deferred	Must	DM
4	Deferred	Can	DC

5. An example of draw an object chart schema

We aim to match the operational processes stored in the process storage with the operational structure, To-Be. The core matching process is described in this section. We start the operational meta software engineering process by (see Figure 4) creating a new operational process using the operation, create (draw an object chart schema). At this moment, the operation consists of a rectangle with no other details available. The name of the created process is entered in the rectangle. The values of the attributes and relationships are as follows:

Name: Draw an object chart schema

Environment: Information systems, software development

Input: Application concepts, list of events

Output: Class, object, associations

Concepts used: Object model, statechart

Related to: NIL

Implemented as: Subprocess

Interacts with: Application engineer/process engineer

Process type: Complex

The process engineer searches the process storage to retrieve the 'Draw Object Model'. We look inside the 'Identify Class' and find two processes, 'Identify Attribute' and 'Identify Service'. The process engineer feels that the 'Identify Class' is not useful but the

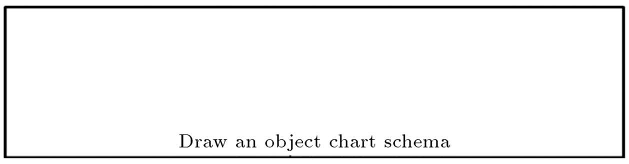

Figure 4. Create (draw) an object chart schema.

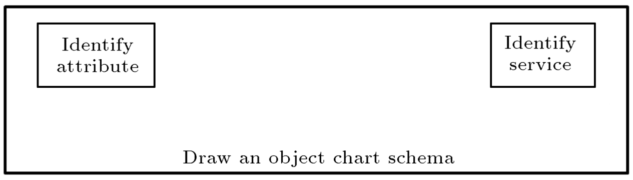

Figure 5. Partial operational process 1.

'Identify Attribute' and 'Identify Service' are. The following sequence of matching operations is applied and the results are shown in Figure 5.

- Un-nest (identify attribute, identify class);
- Un-nest (identify service, identify class).

Now, other processes, i.e. Identify Object and Identify Association, are found to be irrelevant in building the desired process. The process engineer now processes 'Draw a Statechart Schema', as shown in Figure 4. The process engineer applies the following matching operations, one by one:

- Un-nest (identify state, draw statechart schema);
- Un-nest (identify state change, draw statechart schema);
- Un-nest (identify triggers, draw statechart schema),
- Un-nest (cluster states, draw statechart schema).

The result is shown in Figure 6.

Links between operational process segments are set up as follows:

- Sequence (identify state, identify attribute, im);
- Sequence (identify attribute, identify state, dc);
- Sequence (identify state, identify state change, DM);
- Sequence (identify state change, identify state, DC).

At this moment, the operational process of the new process is as shown in Figure 7.

The process engineer now creates self loops to allow iteration. The result is shown in Figure 8.

- Sequence (identify state, identify state, DC);
- Sequence (identify attribute, identify attribute, DC);
- Sequence (identify state change, identify state change, DC).

To identify triggers, we must use the 'Identify Service' after the 'Generate Event'. This is achieved as follows:

Eliminate (generate event, develop transaction, DM);

- Sequence (generate event, identify service, DM);
- Sequence (identify service, develop transaction, DM);
- Sequence (identify state change, identify triggers, DM).

The result is shown in Figure 9.

To complete the integration of the 'Object Model' and 'Statechart' additional operational processes, additional links are defined, as follows:

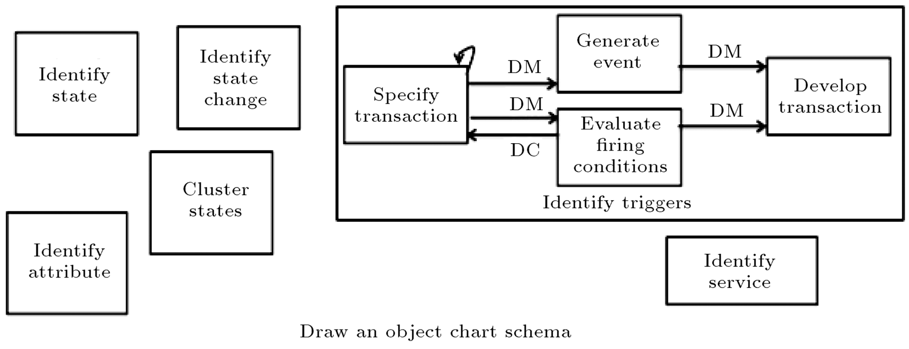

Figure 6. Partial operational process 2.

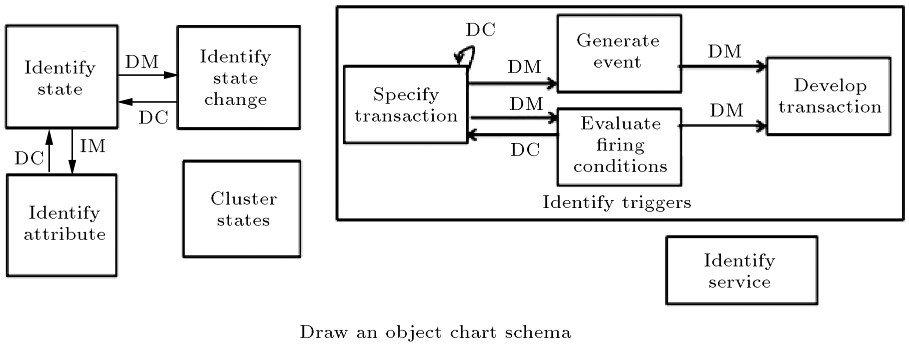

Figure 7. Partial operational process 3.

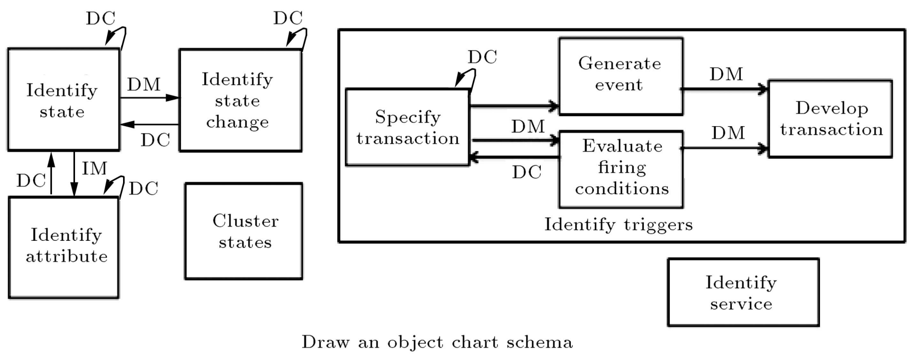

Figure 8. Partial operational process 4.

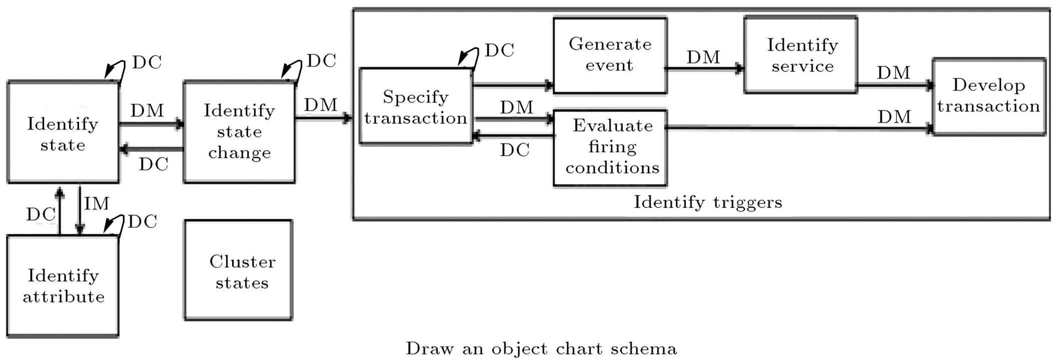

Figure 9. Partial operational process 5.

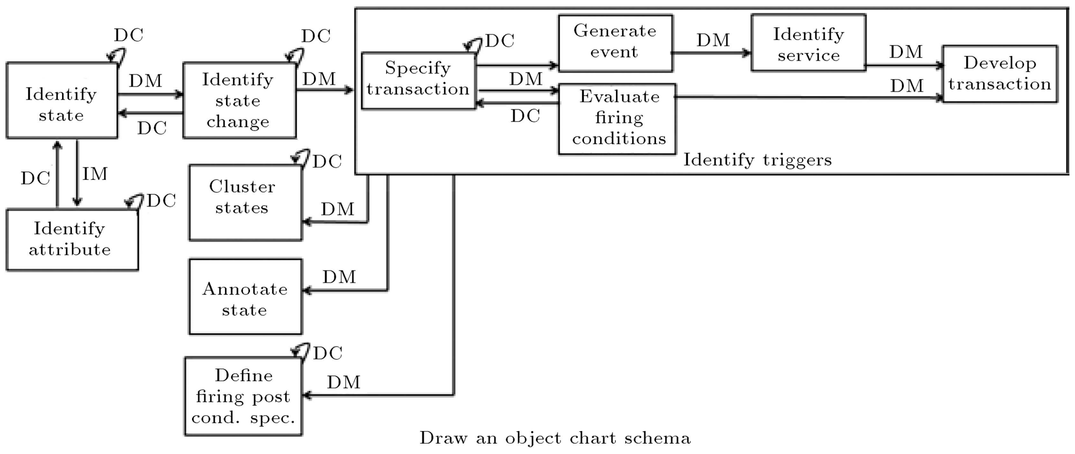

Figure 10. The desired operational process: Draw an object chart schema.

- Sequence (identify triggers, cluster states, DM);
- Create (annotate state);
- Sequence (identify triggers, annotate state, DM);
- Create (define firing post condn spec);
- Sequence (define firing post condn spec, define firing post condn spec, DC);
- Sequence (identify triggers, define firing post condn spec, DM).

In Figure 10, the shaded operational processes show the new operational process needed for introduction to achieve the objective of 'Objectchart'.

6. Conclusion and future work

A process organization represents process features and their interconnections. The interest here is in process concepts, inter-relationships between concepts, constraints, heuristics, guidelines and other such features of a process. It can be seen that process organization represents the structural aspects of processes. Alternatively, it defines the input to be given to a computer aided method engineering tool to engineer/implement the required process. We have selected the generic process model for representation of process organizations.

For the future, in order to finalize construction-design stage interaction, we are also developing a set of operations for performing organization matching. This will allow us to adapt process organizations determined by architecture matching our structural needs.

Thereafter, we propose to develop the goal level. The process goal refers to the goal that the process fulfils. We shall develop the process goal meta-model and provide a precise definition of a goal. We aim to associate a goal with each process and, as for architecture matching, develop the process goal matching operations. Finally, the link between the goal and architecture levels will be defined. Thus, the entire life cycle of Figure 3 will be covered. Once tool support is available, we will experiment with our technique to establish its usefulness.

References

1. Harmsen, F. and Saeki, M. "Comparison of four method engineering languages", In *Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering on Method Engineering: Principles of Method Construction and Tool Support*, Sjaak Brinkkemper et al., Eds., Atlanta, Georgia, United States, pp. 209-231 (Jan. 1996).
2. Brinkkemper, S. "Method engineering: Engineering of information systems development methods and tools", *Journal of Information & Software Technology*, **38**, pp. 275-280 (1996).
3. Royce, W.W., *Managing the Development of Large Software Systems: Concepts and Techniques*, Proceedinas WESCON (1970).
4. Bandinelli, S., Fuggetta, A., Lavazza, L., et al. "Modeling and improving an industrial software process", *IEEE Trans. Soft. Engg*, pp. 440-454 (1995).
5. Bassili, V., Zelkowitz, M., McGarry, F., et al. "SEL's software process - improvement program", *IEEE Software*, **12**, pp. 83-87 (1995).
6. Ralyté, J., Deneckére, R. and Rolland, C., *Towards a Generic Model for Situational Method Engineering, Proc. CAiSE 2003*, J. Eder and M. Missikoff, Eds., LNCS 2681, Springer, pp. 95-110 (2003).
7. Brinkkemper, S., *Personal Email Communication to Authors* (29 Sept. 2006).
8. Brinkkemper, S. "Method engineering: Engineering of information systems development methods and tools", *Information and Software Technology*, pp. 275-280 (1996).
9. Harmsen, F., *Situational Method Engineering*, Moret Ernst & Young (1997).
10. Heym, M. "Method engineering: Specification and integration of development methods for information systems", [in German: Methoden-Engineering: Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme], Dissertation, Hochschule St. Gallen, Switzerland (1993).
11. Heym, M. and Österle, H. "Computer-aided methodology engineering", In *Information and Software Technology*, **35**, pp. 345-354 (1993).
12. Malouin, J.L. and Landry, M. "The mirage of universal methods in systems design", In *Journal of Applied Systems Analysis*, **10**, pp. 47-62 (1983).
13. Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, Reading, MA (1990).
14. Olle, T.W., Hagelstein, J., MacDonald, I.G., Rolland, C., Sol, H.G., van Assche, F.J.M. and Verrijn-Stuart, A.A. *Information Systems Methodologies - A Framework for Understanding*, 2nd Edn., Addison-Wesley (1991);
Vessey, I. and Glass, R.L. "Applications-based methodologies", In *Information Systems Management*, pp. 53-57 (Fall 1994).
15. Purba, S., Sawh, D. and Shah, B. *How to Manage a Successful Software Project - Methodologies, Techniques, Tools*, John Wiley & Sons, New York (1995).
16. Parkinson, J., *60 Minute Software - Strategies for Accelerating the Information Systems Delivery Process*, John Wiley & Sons, New York (1996).
17. Harmsen, F., Brinkkemper, S. and Oei, H. "Situational method engineering for information system projects", In *Methods and Associated Tools for the Information Systems Life Cycle*, T.W. Olle, and A.A. Verrijn Stuart, Eds., *Proceedings of the IFIP WG8.1 Working Conference CRIS'94*, North-Holland, pp. 169-194, Amsterdam (1994).
18. Bucher, T., Klesse, M., Kurpjuweit, S. and Winter, R. "Situational method engineering - On the differentiation of "context" and "project type"", *Situational Method Engineering 2007*, pp. 33-48 (2007).
19. Börner, R. "Applying situational method engineering to the development of service identification methods", *16th Americas Conference on Information Systems*, Lima, Peru, forthcoming, Paper 18, pp. 1-10 (2010).
20. Ralyté, J., Deneckère, R. and Rolland, C. "Towards a generic model for situational method engineering", *Proc. CAiSE 2003*, J. Eder and M. Missikoff, Eds., LNCS 2681, Springer, pp. 95-110 (2003).
21. Anat, A. and Iris, R.B. "Semi-automatic composition of situational methods", *Journal of Database Management*, **22**(4), pp. 1-29 (2011). doi:10.4018/jdm.2011100101
22. Brinkkemper, S., Saeki, M. and Harmsen, F., *Assembly Techniques for Method Engineering, Proc. CAiSE 98*, B. Pernici and C. Hanos, Eds., LNCS 1413, Springer, pp. 381-400.

Biographies

Durgesh Samadhiya obtained a PhD degree from

Chung Hua University, Taiwan. He has published and presented a number of papers in international journals and at conferences. He has made significant contributions to the area of software engineering and networking, and completed many projects in this field.

Wen-Chih Chang is Associate Professor in the Department of Information Management at Chung Hua University, Taiwan. His research interests include e-learning, game-based learning, social network, learning behavior analysis, and institutional research and data analysis.