# Towards Functional Method Engineering

**S.B. Goyal**
Manav Rachna College of Engineering, Faridabad

**Naveen Prakash**
Manav Rachna College of Engineering, Faridabad

**Abstract:** *Different methods are needed for different projects/situations. There is a no single universally applicable methodology to design a new adapted method. Situational Method Engineering, SME promotes the idea of retrieving, adapting, and tailoring fragments to speciic situations for helping method engineers in their efforts to implement a flexible system correctly, on time, and within budget. SME is concerned with the features to be included in methods but does not consider the relationship of these features with the overall task to be performed by the method. This difficulty can be overcome by treating the task as a function, doing method experiencing at the functional level and then engineering the needed features. We refer to this approach as Functional Method Engineering, FME and outline in this paper.*

***Keywords:*** *Method, situation, fragment, method base, method engineering, situational method engineering.*

## 1. Introduction of Method

The 'method' word comes from the Greek 'methodos', i.e. way of investigation. We are dealing with methods for information systems development methods, ISDMs. A number of methods have been developed and successfully deployed for software and information systems development. Examples of such ISDMs are ER Modeling, RUP, DSDM, SSADM, Merise, and object-oriented analysis and design methods alike OMT.

Depending on the nature of the method, it may address either a part or the whole of the traditional System Development Life Cycle. For example, ER modeling addresses only the conceptual design phase whereas SSAD of Yourdon addresses the full life cycles.

Many definitions of a method have been proposed in [1], [13] [22], [12]. According to Brinkkem [1], a method is an approach based on a certain way of thinking, to carry out an Information System (IS) development process consisting of directions and rules structured according to a systematic ordering of development activities and corresponding development methods. Locopoulos [11] has defined a method as a set of specific formalisms and is the working procedure structure to build well-formed instances of specifications. According to Prakash [14], [15] a method is the decision making capability and the mechanism that supports this. A *method* can be considered as a predefined and organized collection of techniques and a set of rules which state by whom, in what order, and in what way the techniques are used by Smolander [21] to achieve or maintain some objectives. A method is a set of processes for a particular objective [17].

Methods were built by individual designers, are monolithic and embody the experience of their designers. For example, ER reflects the beliefs and world view that emphasized entities and their relationships but ignores functions and functional decomposition of SSADM and Merise. Methods are usually described in textbooks and operating manuals giving the step-wise structuring of the development activities and the structural requirements for the products, also called method's deliverables.

## 2. Method Engineering

Early method design was ad hoc and methods were developed based on the experience of their designers. As the number of methods increased, the need for a systematic discipline for engineering methods was felt. This came to be known as Method Engineering, ME. Method engineering is the discipline [2] to design, construct and adapt methods, techniques, and tools for the development of information systems. It can construct methods from scratch, modify and adapt existing methods and assemble given methods or method parts to yield the needed method.

Early researchers searched for a Universal Method, one that could be used to develop any information system. However, it soon became apparent that such a method is an illusion. Rather, [10] methods should be constructed to meet a particular information system needs. For example, a system that is essentially functional needs a functional approach like SSADM, another that is data oriented needs an ER like approach, and others that are a mix of different concepts needs to be constructed with the appropriate mix of concepts integrate together.

This the area of situational method engineering which looks at the project situation, identifies the set of concepts in which it can be represented and then goes on to find methods and method parts that can be brought together in an integrated whole. A central repository called the method base contains the reusable method parts. The method base could be queried and appropriate method parts retrieved. These could then be assembled together to form the desired method. Proposals for querying the method base use descriptors, project contingency factors and multi-criteria search descriptors. Three major proposals exist for situational method engineering: one is based on fragments by Harmsen et al [7], [9] the other on the notion of contexts by Rolland et al [19], [20] and the third is based on decisions proposed by Gupta [6].

The approach of Brinkkemper [3] relies on the experience and knowledge of the method engineer in ensuring well selected components and building the required method. Grundy [4] proposed an integrated facility for carrying out method development from scratch, method modification and method reuse. Gupta [6] proposed an instantiation algorithm that formed the basis for metho~~d development from scratch,~~ by modification and by assembly.

It was explicitly realized in [6] that before starting to fit method parts together, some additional preparatory work needs to be done. It was proposed to have a method development life cycle that, in its first stage, identified method requirements and only thereafter would method parts fitting into these requirements were to be located. The belief was that method requirements were essentially the broad, global structure and purpose of the method concerned. The structure dealt with whether the method was adomic (like ER) or consisted of a number of collaborating ones (like OMT) etc. The purpose was whether the method was transformational (for converting an ER schema intorelational) or constructional (for building a schema).

Ralyte [18] suggests that method engineering is facilitated if the intention of the method can be determined and raises some questions: a) how can assurance be provided that the method to be enhanced, extended, or restricted is a good candidate method? b) What are the chances that at the method engineering intention stage, the method shall have to be discarded because its adaptation is very difficult? c) Should not some more exploratory work be done before committing to setting up method adaptation intentions?

The approach to Functional Method Engineering proposed by us is directed towards answering these questions. Whether method, M is a good candidate for adaptation, enhancement, restriction can be determined if it does similar work to that of the one being engineered. This work is conceptualized in the notion of the function carried out by a method. Further, we believe that the closer the two methods are in this respect, the less are the chances of discarding M at a later stage due to adaptation difficulties.

## 3. Functional Method Engineering

In this section, we start from the differences between SME and FME, then consider the details of FME. We explain the two key notions of Method architecture and Method Organization. Thereafter, we consider the representation of a method for building the ER schema.

Let us bring out the difference between SME and FME being proposed here. As an example consider the fragment based SME proposal [3]. We have two fundamental element a) *product and their structures* b) *procedures and their execution order to develop the products.* It is clear from (a) that interest is the structure of products. Similarly, since the structure of a process is largely determined by the order of execution, interest is in process structure. Therefore, we can conclude that SME is centered on the structural aspects of methods.

*This focus on engineering the structure of methods de-emphasizes what the method does, what task it is good for.* In fact, the determination of whetehr the method structure can carry out the project task at hand is based on the experience of the method engineer. In other words, the method engineer determines the task of the project by some ad-hoc means selects the appropriate method structures and then assembles these together.

FME is based on Method Development Life Cycle, MDLC [5], [16] for method development as shown in Table 1. As illustrated MDLC in [16], **the Requirements**

**Engineering stage** consists of Intention Matching. First, the intention of the method To Be is determined through interviews that focus on eliciting what the method aims at achieving and resulted methods become candidates for the second stage of the MDLC. The second, **Design Engineering stage** of the MDLC considers intentionally similar methods of the Requirements Engineering stage one by one. The architecture of each candidate is retrieved from the method repository. This reveals the main components and their inter-relationships that comprise the method. That subset of these components and inter-relationships is selected which best meets the broad functional needs of the method To-Be. Such selections are made from all the candidate methods and are synthesized together into the architecture of the desired method. It is possible that some method component may have to be introduced if it is not available in any of the selected architectures. **The Construction stage** deals with the details of the architecture of the method To-Be that has been synthesized in the Design Engineering stage. The architecture is populated with instances of the method features needed in the method. These features may be in accordance with the concepts of the fragment, contextual, decisional or any other meta model used.

**Table 1: The Method Development Life Cycle**

| Stage | Process | Input | Output |
|---|---|---|---|
| Requirements Engineering | Intention Matching | Intention of method To Be obtained from Interviews, documents etc. | Intentionally similar methods to the method To be |
| Design Engineering | Architecture Matching | Architectures of intentionally similar methods | Architecturally similar method to the method To Be |
| Construction Engineering | Organization Matching | Architecturally similar method | Situated Method |

FME occupies the last two stages: 1) Design Engineering and 2) Construction Engineering of MDLC in Table 1. It puts method structure subordinate to method functionality. FME asks for an explicit determination and representation of method functionality in the form of method architectures. It is only after the architecture has been built that the issue of method structure is to be considered. In this sense, SME occupies the downstream, construction engineering stage of our life cycle in Table 1.

FME defines Method Architecture as the class of all methods that perform the same function. Method architecture has been defined as an **abstraction that identifies its components and inter-relationships to highlight the externally visible functionality of the method** [16]. An architecture is dependent upon another if a method belonging to it can be enacted after a method of the former is enacted. Thus, dependencies define a successor-predecessor relationship between method architectures. Dependencies show different properties. To capture this, two attributes, urgency and necesity, are associated with each dependency type [16]. Urgency refers to the time at which a method of the dependent architecture is to be enacted. If it is to be enacted immediately after the first, then this attribute takes on the value *Immediate*. If it can be enacted any time, immediately or at any moment, after the first, then urgency takes on the value *Deferred.* Necesity refers to whether or not a method of the dependent architecture is necessarily to be enacted after the first has been enacted. If it is necessary to enact it, then this attribute takes the value *Must* otherwise it has the value *Can.* Table 2 shows the four types of links that can exist between methods. We shall refer to these by their abbreviations shown in the fourth column of Table 2.

**Table 2: Types of Links**

| Type | Urgency | Necessity | Abbreviation |
|---|---|---|---|
| 1 | Immediate | Must | IM |
| 2 | Immediate | Can | IC |
| 3 | Deferred | Must | DM |
| 4 | Deferred | Can | DC |

## 3.1 Example

The architecture of a method to build the ER schema is shown in Fig. 2. Since the function of the method is to be defined, we refer to the architecture of Draw ER Schema. As shown, it is a complex architecture that represents a collection of methods for building ER schemata. It consists of two simpler method architectures, Draw Entity and Draw Relationship respectively. The link types between these are shown. The components of these latter two are also shown in the Fig. 2.

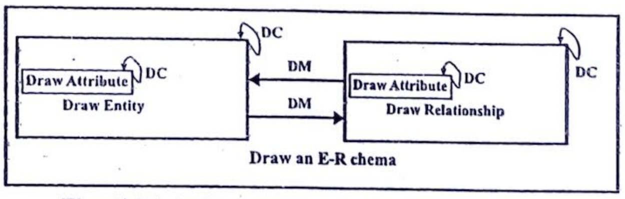

**Fig. 1. Method Architecture of Draw ER Schema**

The self-loops show that it is possible to do the same function many times.

After method architecture, we move to Method Organization, MO [17] **in developing a detailed design of the method.** It is useful in the third stage of the MDLC, i.e. in construction engineering. Each organization is a specification of the features that are to be built in the method. There can be many organizations for a given architecture.

Now, consider MO of the function Draw Attribute in an ER method. Consider a variant of the method that does not allow derived attributes and another which does. This is made precise in the cases in two cases: 1) Both Single: valued/multi-valued attributes are allowed, 2) As above but derived attributes are also allowed. We show the organisations for these cases in Figs. 3 and 4 respectively.

Consider figure Fig. 3, the method starts off by identify the attribute of interest. This is captured by the node <Attribute, Identify>. It is now possible to determine whether the attribute is atomic or composite i.e. the nature of the attribute and captured this with the node <nature, Determine>. The nature of the edge connecting this with <Attributem Identify> is IM as shown. After <nature, Determine> we have two options, if we found that nature of attribute is complex then again we can move to <Attribute, Identify> with the link DC otherwise find the valuation of the attribute i.e. single valued or multi-valued attribute i.e. the Valuation of the attribute. This is captured by the node <valuation, Define> and link type is DM as shown in the figure.

Method organisation shown in Fig. 3 is ~~for the~~ case of both single/ multi-values and derived attributes. This organisation is the same as that in Fig. 2 except that we have one more option in case of base attribute and derived attribute i.e. virtualization of an attributes. This is capture by the node <Virtualisation, Find >. As shown, the link type is DM between <nature, Determine> and <Virtualisation, Find>.

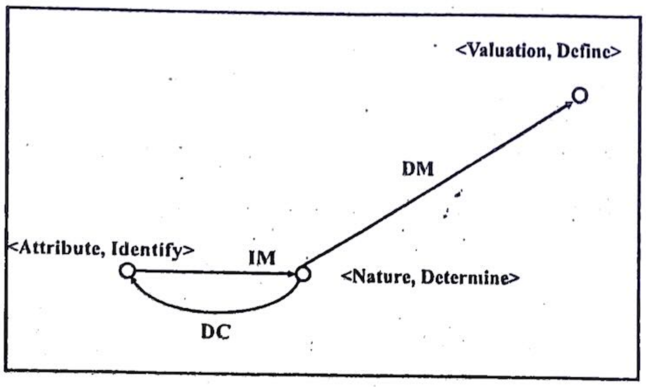

**Fig. 2. Method Organization for case 1.**

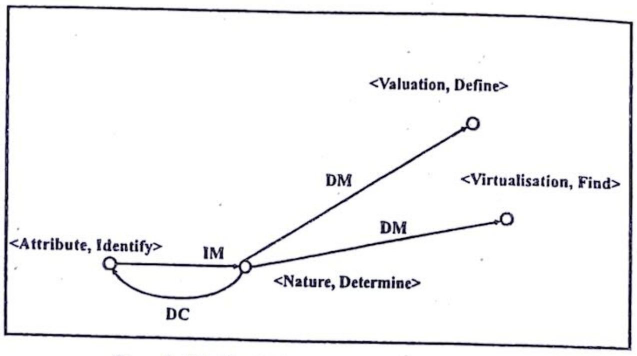

**Fig. 3. Method Organization for case 2.**

It can be seen that MO is an elaboration of MA to determine *the capabilities provided by a method, the interconnections between these,* and *the constraints that are applicable.*

## 4. Comparison with SME

The basic process for engineering architectures and organisations is assembly based. Just as method chunks, fragments etc. can be assembled from re-useable parts, so also architectures and organisations can be assembled together from re-useable components.

Now consider the second row of Table 3 to differentiate between SME and proposed FME. The method base consists of the universe of method knowledge available o the method engineer. In SME approach this knowledge is organized as a flat structure, in terms of modules in a single level. In FME approach, the universe is organized in two levels, the architecture and organisation component levels. Additionally, these two levels are linked by the 'is implemented as'

**Table 3: Comparison between SME and FME**

| Criterion | SME | FME |
|---|---|---|
| Method Knowledge | Chunk, fragments, patterns, method blocks, components | Architecture components Organization components |
| Method Base | One level | two levels for architecture, organization, related by *is implemented as* |
| Selection strategies | One shot selection: from universe of method knowledge | Progressive:<br>1. Architecture selection<br>2. Organisation selection from short-list of selected architectures |
| Construction Process | Assembly | Assembly |
| Applicability | ISDM | ISDM, BPM, Robotics, Project Management etc. |

relationship and a method *architecture* can be implemented as one or more method organization.

Tool support nature for FME is shown in Fig. 4. As can be seen the method reposiroty consists of two base parts, the Architecture and the Organization respectively. These two bases are interlinked. Thus, it is possible to move to the organization once the architecture is identified.

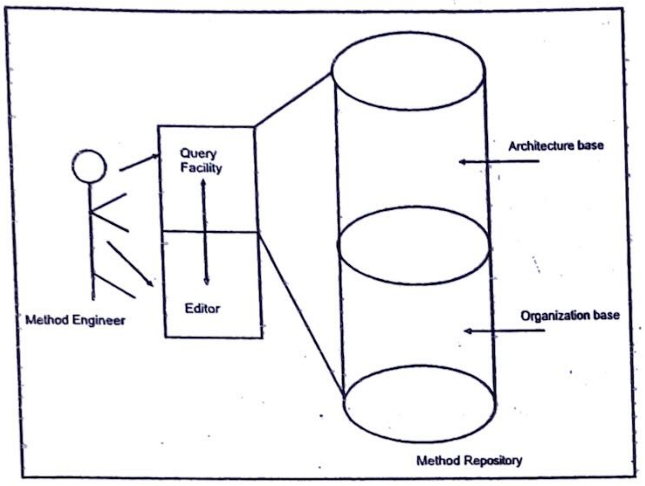

**Fig. 4. FME Tool Sketch**

The Query facility is used by the method engineer to navigate from intention, architecture to organization. The results of the query are displayed in the editor in graphical form. This form is edited to build the required method. The two bases are then populated with the information about the constructed method.

## 4.1 Comparative Example

We bring out the difference between SME and FME with a simple example of method assembly: Object Model of OMT in case of Object-Oriented Analysis/ Design and Harel's Statechart to Objectchart for specifying objectclasses. An Objectchart transitions correspond to the state-changing methods that the class provides and those that it requires of other classes. Object attributes and observer methods annotate Object chart states. Firing and post conditions are used to specify the effect of transitions on class attributes.

### 4.1.1 SME

In the SME approach, the structures of the Statechart and Object model are given. It is assumed that both these are relevant to the project at hand. No effort is made to articulate or to verify if this is indeed so. Thus, we directly preceed to integrate the structural aspects of our two rechniwues.

Statecharts [Bri98] can be seen as an extension of finite state transition diagram and can represent hierarchical decomposition of state: AND decomposition for concurrency, and OR decomposition for state-clustering. The description of the method fragment shown in Fig. 5.

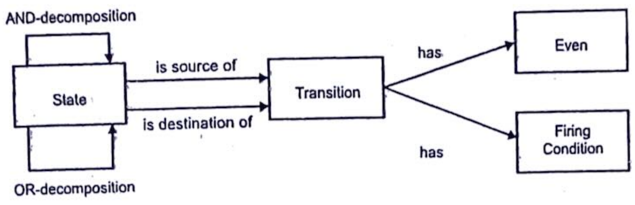

**Fig. 5. Statechart [Bri98]**

The object model identifies the object classes in the system and its conceptual structure is shown in Fig. 6.

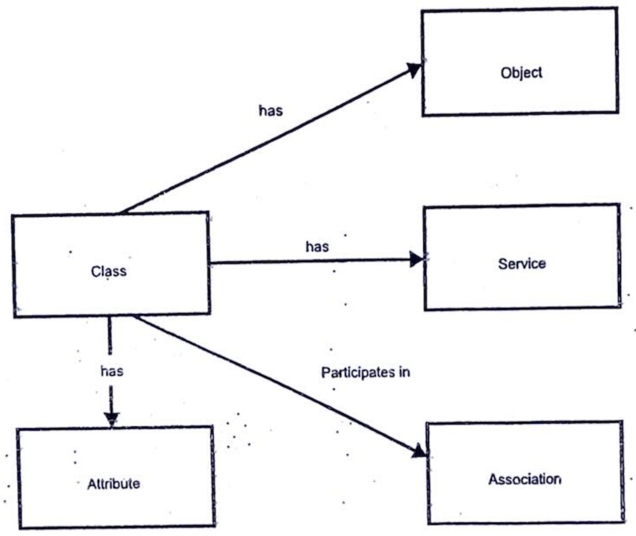

**Fig. 6. Object Model [Bri98]**

The SME assembly process to produce Objectchart by assembling Statechart and Object method fragments in shown in Fig. 7. The method engineer has introduced some new notions to establish relationships between concepts found in Statechart and Object Model. These are, for example, post condition, is annotated with, consists of, and has between Transition and Post condition. A new property, is _hidden?, has also been introduced.

### 4.1.2 FME

In the FME approach, we concentrate on the task rather than structure. Thus, our starting point is not Figs. 5 and &. Rather, we draw an architectural diagram of the Objectchart method as shown in Fig. 8, which

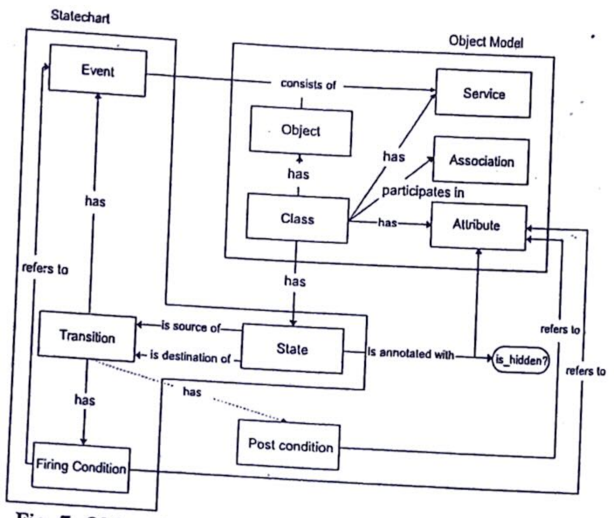

**Fig. 7. Objectchart: Assembly Process of SME in Product Perspective [Bri98]**

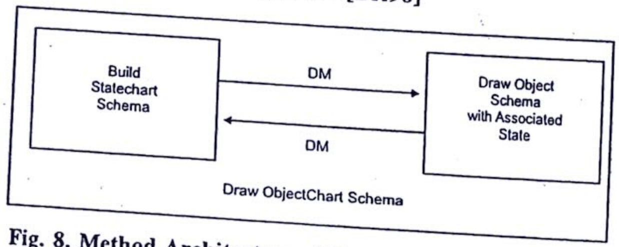

**Fig. 8. Method Architecture of Draw ObjectChart Schema**

consists of two method architectures *Build Statechart Schema* and *Draw Object Schema with Associated State* respectively. The link types between these are shown. The components of these latter are shown in Fig. 11 after applying the FME process.

Now, we can design *Build Statechart Schema* using the FME process as shown in Fig. 9.

After this, we can build method architecture of *Build Object Schema for OMT* in Fig. 10.

To produce Objectchart by applying FME process on Statechart FME process on Statechart and Build Object Schema for OMT is shown in Fig. 11. The method engineer has introduced new method fragment *Identify Fire Operation* between *Specify Transaction* and *Identify Operation* with appropriate links.

In Fig. 11, Resultant of FME process achieved using introduced the new method fragment *Identify Fire Operations* with appropriate link types.

It can be seen from Fig. 11 that FME de-emphasizes the structural concepts of methods and highlights the manner in which the method can be deployed. The structural aspect is, of course, to be

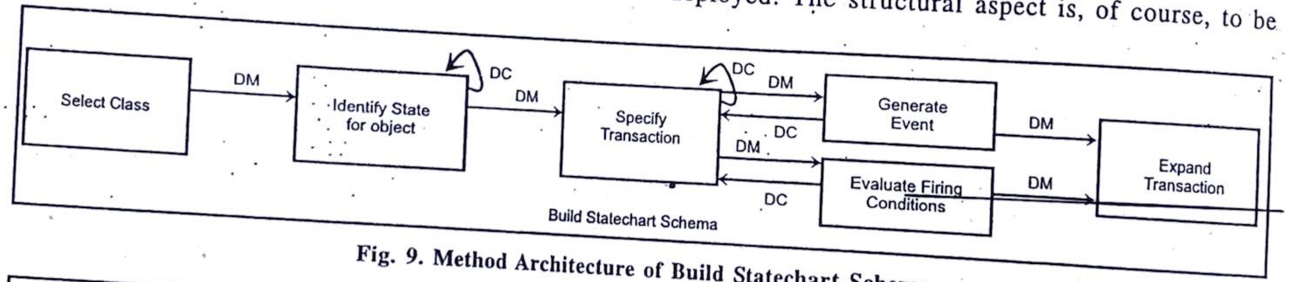

**Fig. 9. Method Architecture of Build Statechart Schema**

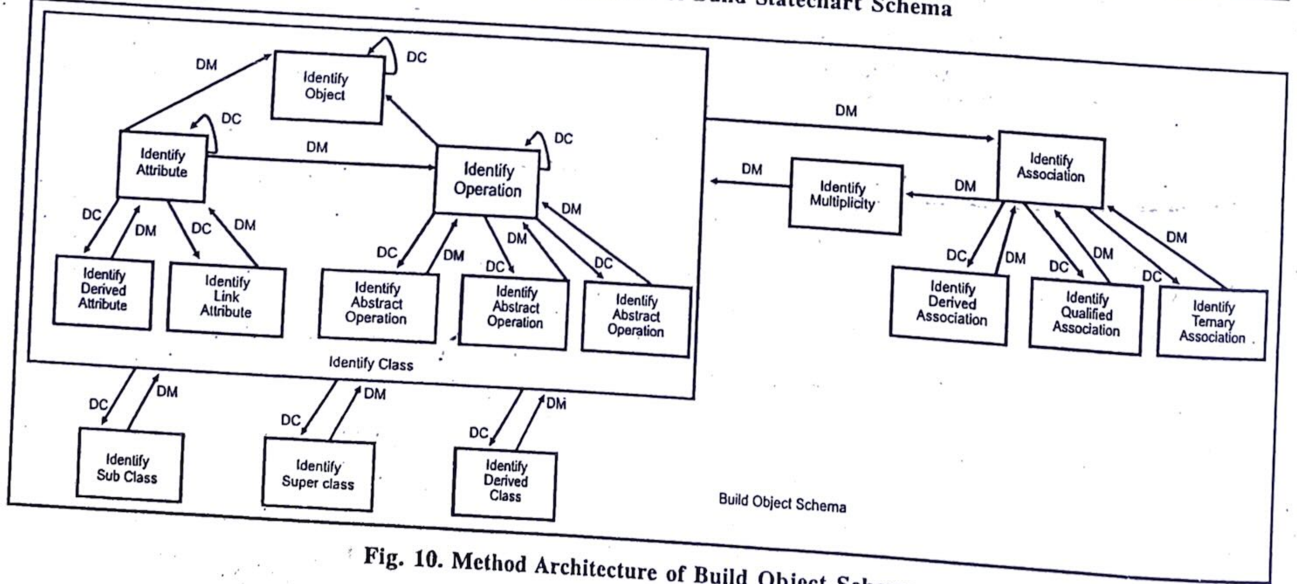

**Fig. 10. Method Architecture of Build Object Schema**

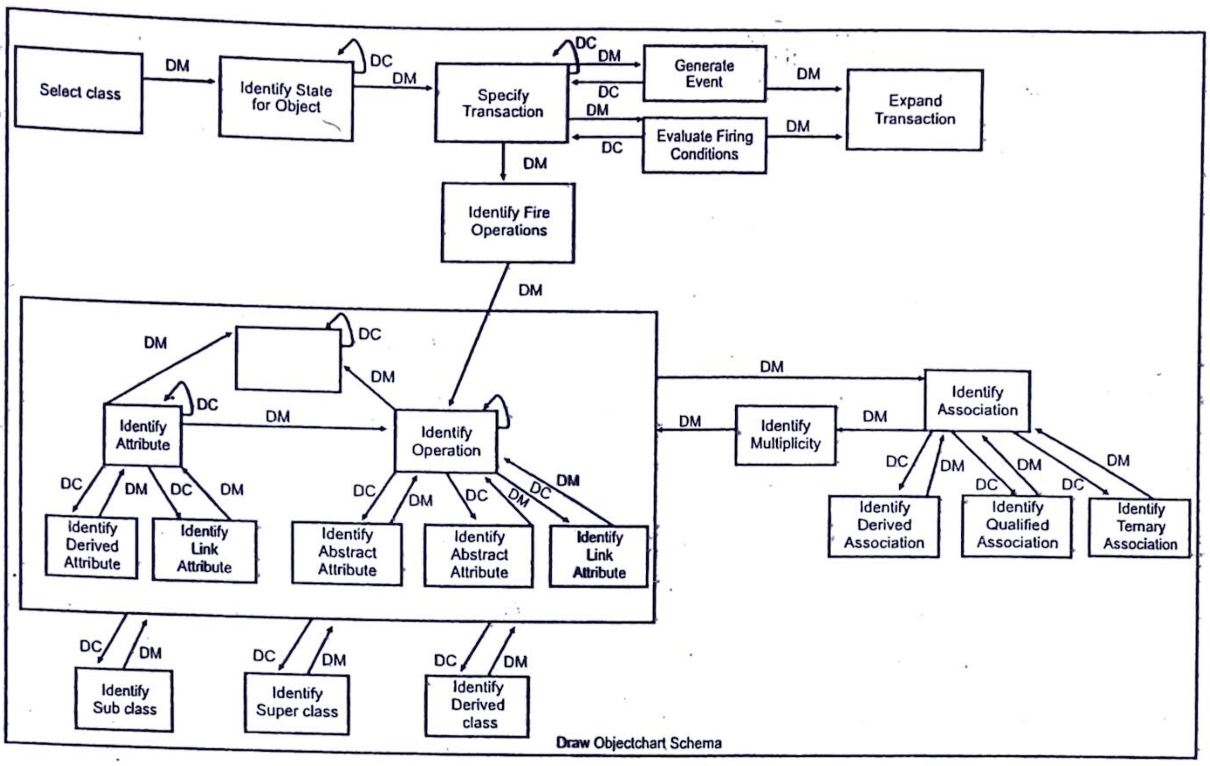

**Fig. 11. Method Architecture to Draw Objectchart Schema**

designed separately but is seen as a secondary task to the main task of specifying the function.

### 4.1.3. SME vs. FME

Our basic premise is that we should stand back from structural concerns of method engineering instead, look for functional method properties that define what the method must do. We represented this though the notion of method architecture. Thus, method engineering now starts off by method architecture engineering. As a result the components of the architecture and their inter-connections, or in other words, the sub-functions and their order of enactment, are the first items to be engineered. This functional method engineering is to be supported through an architecture method base for retrieval and storage of architectures.

Finally, SME approach is structured based instead of functionally to be achieved. In FME approach, we concentrate towards the task to be achieved rather than structure.

## 5. Conclusion and Future Work

The selection of a good candidate method is a big issue in SME. We believe that the solution to this can be found not in the structure of a method itself but in the work to be done, the task to be performed by the method. Functional Method Engineering is providing the solution of selecting a good candidate method from repository. FME uses a ***progressive selection strategy*** which has the capacity to make architecture and organisation selection more specific. As a result from FME, method selection for adaptation shall be more appropriate and give assurance that the SME task is progressing purposefully. *The chance of method rejection at later stages shall be considerably reduced.*

One future work for FME will move away from features and look at methods in a global sense. Our global view is derived from organizing method enginering in three levels, situational, functional and intentional. The last lays down the requirements, the second the functional architecture to meet these needs and the first provides the set of concepts and interrelationships to realize the functional. Progression occurs down the three levels. We will concentrate on functional method. We will show that progression occurs within this level and a stage is reached where method features get identified.

## References

1. Brikkemper S., *Formalisation of Information Systems Modelling, Ph.D. Thesis*, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

2. Brinkkemper, S., Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, 38, pp. 275-280, 1996.

3. Brinkkemper S., Saeki M., Harmsen F., Assembly Techniques for Method Engineering, in Method Engineerng Principles of Method Construction and Tool Support, Brinkkemper, Lyytinen, and Welke (eds.) Chapman and Hall, 45-62, 1996.

4. Grundy J.C. and Venable J.R., Towards an Integrated Environment for Method Engineering, in Method Engineering Principles of Method Construction and Tool Support, Brinkkemper, Lyytinen, and Welke (eds,) Chapman and Hall, 45-62, 1996.

5. S.B., Requirement Engineering Stage for Method Engineering in MDLC, Proceedings International Conference on Data Management (ICDM-2008), 1315-1321, 2008.

6. Gupta D. and Prakash N., Engineering Methods form Method Requirements Sepcifications, Requirements Engineering Journal, 6, 3, 135-136, 2001.

7. Harmsen F., et al, Situational Method Engineering for Information System Project Approaches, in Methods and Associated Tools for the Information Systems Life Cycle, Verrijn-Stuart and Olle (eds.), Elsevier, 169-194, 1994.

8. Harmsen Frank, Sjaak Brinkkemper: Design and Implementation of a Method Base Management System for a Situational CASE Environment, PSEC: 430-438, 1995.

9. Harmsen Frank, Marnix Klooster, Sjaak Brinkkemper, Gerard Wijers: Intranet Facilitated Knowledge Management: A Theory and Tool for Defining Situational Methods. CAiSE 1997: 303-317, 1997.

10. Kumar, K., Welke, R.J., Methodology engineering: a proposal for situation-specific methodology construction. In: *Challenges and Strategies for Research in Systems Development* (eds. W.W. Cotterman, J.A. Senn), John Wiley & Sons.Ltd., pp. 257-269, 1992.

11. [Loucopoulos P., Zicari R., *Conceptual Modelling, Database and CASE,* Wiley (Pub.), 1992.

12. Lyytinen K., Smolander K., Tahvainen V-P, *Modelling CASE Environments in Systems Work,* Conference on Advanced Information Systems Engineering, Sweden, 1989.

13. Prakash N., A Process View of Methodologies, CAiSE 1994, 339-352.

14. Prakash N., *Towards a Formal Definition of a Method,* Requirements Engineering Journal, Vol. 2(1), Springer Verlag, U.K., 1997.

15. Prakash N., *On Method Statics and Dynamics,* Requirements Engineering Journal, Vol. 24 (8), Springer Verlag, U.K., 1999.

16. Prakash N. and Goyal S.B., Towards a Life Cycle for Method Engineering, Proceedings Eleventh International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'07), 27-36.

17. Prakash N and Goyal S.B., Method Architecture for Situational Method Engineering, 2nd IEEE International Conference on Research Challenges in Information Science (RCIS'08), Marrakech, Morocco, ISBN 978-1-4244-1677-6, pp. 325-336, 2008.

18. Ralyte J; Deneckere R., Rolland C:, Towards a Generic Model for Situational Method Engineering, Proc. CAiSE 2003, Eder J. & Missikoff M. (eds.) LNCS 2681, Springer, 95-110, 2003.

19. Rolland Colette, Veronique Plihon: Using Generic Method Chunks to Generate Process Models Fragments. ICRE 1996: 173-182, 1996.

20. Rolland Colette, Challenges in Object Oriented Modelling: From Conceptual Modelling to Requirements Engineering. In Proceedings of OOIS' 1996.

21. Smolander, K., Tahvanainen, V.P., Lyytinen, K., How to Combine Tools and Methods in Practise - a Field Study In: *Lecture Notes in Computer Science, Second Nordic Conference CAiSE '90,* (eds. B. Steinholtz, A.S. lvberg, L. Bergman) Stockholm, Sweden, May, pp. 195-211, 1990.

22. Wynehoop J.D., Russo N.L., *System Development Methodologies: Unanswered Questions and the Research-Practice Gap,* Proc. of 14th ICIS, Orlando, USA, 1993.

□