

Developing Application-Centric Methods

Naveen Prakash¹ , M.P.S. Bhatia²

¹JIIT A10, Sector 62 NOIDA 201307, India
praknav@hotmail.com

²NSIT Sector 3, Dwarka New Delhi, India
mpsghatia@nsit.ac.in

Abstract. Meta-models and Generic models have been built in the area of Information Systems to facilitate the task of system designers. It de-emphasises the view under which applications are conceived in terms of real world data, operations, constraints and processes. We refer to methods that help in constructing such applications as ‘application-centric’ methods. We show here that a Generic Model can handle application-centric methods by identifying application objects and operations as well as process models.

1 Introduction

The generic model [6] has been instantiated as the decisional meta-model [5]. Subsequently, a method engineering technique has been developed in [3]. Thus it can be seen that the generic model/meta model combination is useful in engineering Information Systems Development Methods. For the generic model to be truly generic, it should be possible to apply it in different domains. We have shown in [7] its applicability to domains like Robotics, Information Systems and Project Planning.

We intend to show here that the generic model is helpful in developing methods for application development:

- (a) operations to be provided in the application NOT those provided by the system,
- (b) application objects that these operations act upon NOT information structures,
- (c) application constraints that must be satisfied NOT system constraints,
- (d) the manner in which application operations are used to form application processes as different from information system development processes.

We can use the generic model to represent the (a) Functional aspects of applications: what application object is manipulated by what operation, and (b) Constraints of applications: which function can be performed before/after which other one. Constraints of this type can be looked upon as defining a class of process models that can be built using functional information. The most appropriate process model can then be selected.

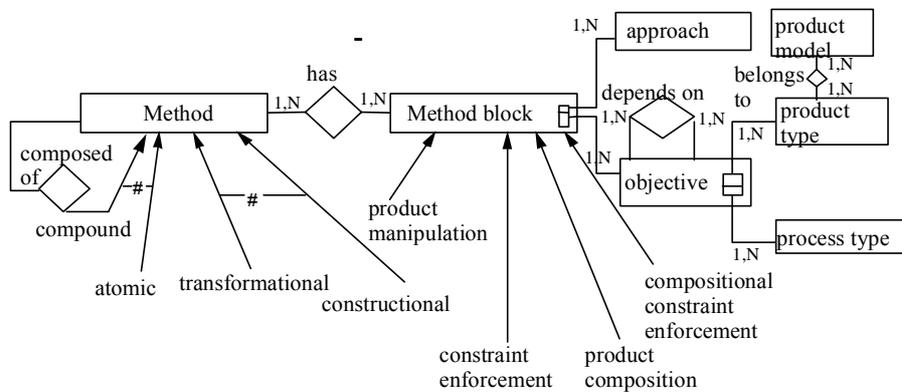


Fig. 1: Generic Method Statics

The Generic Model [6] looks upon a method as having two aspects, statics and dynamics. Here, we consider Generic method statics only (Fig. 1). As shown, product types belong to a product model. For each product type of the product model, the process types that act upon it are associated with it to yield the **objectives** of the method. Thus, an objective is a pair $\langle \text{product type, process type} \rangle$. Fig. 1 shows that a method block is a pair, $\langle \text{objective, approach} \rangle$. The approach tells us the technique that can be used to achieve the objective of the block. Details of the rest of the Generic Model can be found in [6].

Fig. 1 shows that objectives exhibit dependencies among themselves. In [5] there are four kinds of dependencies. The Requirement (REQ) dependency says that an objective O2 **must** be performed after O1 has been performed. The Activate (ACT) dependency says that O2 **can** be performed after O1. Both these have their inverse relationships that are described in [5].

Generic method statics suggests that application methods can be developed by (a) identifying product types, i.e. application object types, (b) process types, i.e. application operations, and (c) dependencies. The first two are used to form application objectives that denote application functionality. These objectives can then be ordered using dependency relationships between them. This ordering produces the set of process models.

2 Developing an Application-Centric Method

In this section we consider building a method for the domain of Passport making. The Passport Office has forms in which an application for passport services can be made. These services may be the issue of a fresh passport, renewal of an expired passport, issue of a duplicate passport, termination of a valid passport. The passport office verifies the details of the requestor and checks that there is no police record that might disallow the issue of the passport. Also, it is determined whether the requestor has to comply with emigration formalities before travelling abroad or not. Thereafter, the

passport is issued. It is possible to add dependents of the passport holder on his/her passport or to delete them from the passport.

There are a number of product types in this application. This set, P, is as follows:

$P = \{ \text{Request, Passport, Expired passport, Duplicate Passport, Dependent} \}$

The set of process types, A, is

$A = \{ \text{Receive, Verify, Refuse, Issue, Renew, Terminate, Add, Delete} \}$

The set of objectives is obtained by taking the cross product, $P \times A$. This can give rise to meaningless objectives. The meaningful ones are selected and the set of objectives O is as follows:

$O = \{ \langle \text{Request, Receive} \rangle, \langle \text{Request, Verify} \rangle, \langle \text{Request, Refuse} \rangle, \langle \text{Passport, Issue} \rangle, \langle \text{Duplicate Passport, Issue} \rangle, \langle \text{Expired Passport, Renew} \rangle, \langle \text{Passport, Terminate} \rangle, \langle \text{Dependent, Add} \rangle, \langle \text{Dependent, delete} \rangle \}$

The Requirement dependencies of these objectives are

$\langle \text{Request, Receive} \rangle \text{ -- Req.} \rightarrow \langle \text{Request, Verify} \rangle$

whereas the Activate dependencies are

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Request, Refuse} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Passport, Issue} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Duplicate Passport, Issue} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Expired Passport, Renew} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Passport, Terminate} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Dependent, Add} \rangle$

$\langle \text{Request, Verify} \rangle \text{ -- Act.} \rightarrow \langle \text{Dependent, Delete} \rangle$

Taking these dependencies into account, the process model as follows is obtained:

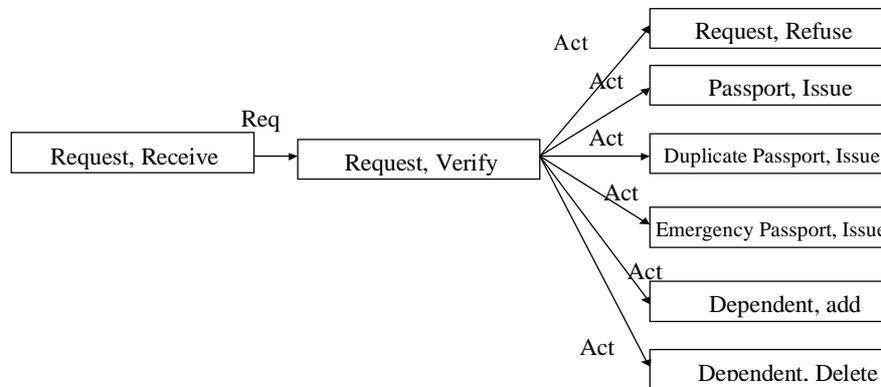


Fig. 2: Passport Process Model

3 Conclusion

Information system development processes have been modelled in the area of **Process Modelling**. The effort in Process Modelling is not oriented towards modelling application processes. As seen here, the construction of such models is made possible

in application centric methods by determining application objectives and ordering constraints. A class of process models is obtained from which any one can be selected for enactment. **Business Process Modelling** deals with the modelling of business processes. Here, business objects and business tasks are determined, and the business model to be followed is arrived at. Application centric methods, in contrast, consider application objects and tasks and the application process model.

Now, consider requirements engineering (RE) methods. **Goal-oriented RE methods** [1,2] over operational goals starting from organisational goals. Operational goals help in discovering system functionality but do not help in discovering which application process type acts on which product type and the constraints that relate the set of objectives together. Further, application process models are not of interest in goal-oriented RE methods. **Scenario based RE methods** [4] help to discover requirements through a study of expected user-system interaction. Again, application centric methods are different from scenario based methods. They do not worry about system responses but focus on stimuli, the application objects that get affected by these stimuli, and ordering of these stimuli for the entire application system to behave as a cohesive whole.

The Generic Model provides a unifying view of an application: the objective represents the functional capability available to the user whereas the Approach represents the manner in which this capability is provided. Thus, the Generic Model suggests both, application-centric and design-centric methods.

References

1. A.I. Anton, Goal based requirements analysis. Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96, pp. 136-144, 1996.
2. A. Dardenne, A. van Lamsweerde, S. Fickas, Goal directed requirements acquisition. Science of Computer Programming, 20(1-2), pp.3-50, April 1993.
3. Gupta D. and Prakash N., Engineering Methods From Requirements Specification, Requirements Engineering Journal, 6, 3, 133-160), 2001.
4. Jacobson, The use case construct in object-oriented software Engineering. In "Scenario-based design: envisioning work and technology in system development", John M. Carroll (ed.), John Wiley and Sons, 309-336, 1995.
5. Prakash N., Towards a Formal Definition of Methods, Requirements Engineering Journal, Springer, 2, 1, 23-50,1997.
6. Prakash N., On Method Statics and Dynamics, Information System Journal, Vol. 24, No.8, 613-637,1999.
7. Prakash N., and Bhatia M.P.S., Generic Models for Engineering Methods of Diverse Domains, Advanced Information Systems Engineering (CAiSE 02), Pidduck, Mylopoulos, Woo and Ozsu (eds.), LNCS 2348, 612-625,2002.