A framework for method integration

Oates, Briony Jane and Bell, Frances

A FRAMEWORK FOR METHOD INTEGRATION

Frances Bell
Department of Applied Sciences
University College Salford
Frederick Road
Salford
M6 6PU

email
frances@uk.ac.ucsalf.scot1

Briony J. Oates
School of Computing & Mathematics
University of Teesside
Borough Road
Middlesbrough
TS1 3BA

B.J.Oates@uk.ac.teesside

Key Words : framework, method, methodology, method integration, reuse, meta-method, model, analysis, evaluation.

Abstract
In the information systems field, there is growing interest in the issue of method integration. The aim of this paper is to begin to build a framework for the understanding and evaluation of work in this area, using a new model of a method to analyse the 'method for method integration' and its resulting products.

Introduction

In the software engineering and information systems development communities there is considerable interest in the issue of 'method integration', that is, discovering ways in which methods can be combined in order to improve the development process and outcome. In this paper we examine current approaches to method integration, and begin to build a framework to support the analysis and evaluation of method integration.

Firstly, we justify the need for a method integration framework by reviewing some of the areas in which method integration is being attempted, and the questions that arise as a result. We define a model for a method, which is then used to define our meta-method for method integration. This meta-method is examined at a high level, identifying five different types of method integration; and by using the model of a method to examine in greater detail the output, or integrated method, for each type of method integration. Finally, we summarise the main points and propose areas of further work.

A recent, detailed analysis of method integration is given by Kronlöf (1993). This work is an outcome of activities carried out in the Esprit Project 2565 ATMOSPHERE, and the book's objective was to contribute to the development of 'method engineering':
'We envisage that in the future the development and customisation of methods will be supported by well-defined meta-methods based on a firm conceptual framework.' Kronlöf (1993)

In this paper, we shall show where we have built on the work of Kronlöf, and where we propose additional concepts of relevance to the study of method integration. We have also been influenced by Brown and McDermid (1992) who explore integration in a (computer-based) software development environment.

The need for a method integration framework
Method integration work has been identified in the following areas:

1. Combining formal and structured methods (eg Polack, Whiston and Mander (1993), Semmens and Allen (1991), Fencott and Lockyer (1993), Larsen, Plat et al (1991)). Formal notation methods such as Z have not been widely adopted, one reason being their intimidating appearance to many developers and users. Method integration work in this area seeks to use the formal notation methods to provide a rigorous foundation for the structured methods (such as Yourdon Systems Method or SSADM), and to improve the usability of the formal notation methods by adding the diagrammatic notations of structured methods.

2. Combining 'hard' and 'soft' methods, for example Multiview (Avison and Wood-Harper (1990)) and COMPACT (CCTA (1989)), which both use a combination of methods drawn from the hard structured methods and from the softer ETHICS (Mumford (1983)) and Soft Systems Methodology (SSM) (Checkland (1981)).
Other examples can be found in recent work attempting to combine SSM with data flow diagrams (eg Merali (1992)) or with data-focused approaches (eg Lewis (1993)).

3. As new versions of methods are released, they tend to increase their repertoire and include other methods within them. For example, the latest version of the Yourdon Systems Method (Yourdon (1993)) includes a type of entity life history analysis, which was not previously part of the method (Yourdon (1989)).

4. There is a recognition that a contingent approach to methods is often necessary (Olle (1991)), choosing and combining methods to suit the particular situation, since there is not one all-purpose method. In Kronlöf's concept of method engineering, a set of methods appropriate to the project is chosen, they are modified as necessary, novel methods are developed to 'bridge the gaps' between the chosen methods, and the modified and developed methods are combined to form a new and more comprehensive method (Kronlöf (1993)). Some methods are deliberately designed as 'toolbox methodologies', allowing the user to choose the appropriate methods from a given selection and combine them into a specific method for the project in question eg Multiview (Avison and Wood-Harper (1990)). Other methods, whilst having a well-defined process model, include a tailoring step eg SSADM (Eva (1992)), where the user adapts the method to suit the project, perhaps deciding on an appropriate subset of models and steps.
In all of the above cases, there are the concepts of choosing appropriate methods for a specific project, and then combining them together in a meaningful way.

5. Reuse of analysis and design products is as important as the reuse of program code (Poulin, Caruso, Hancock (1993)). It is inevitable that systems will have been developed in the past using methods which are no longer used by an organization. A way needs to be found to link the products of the older methods into the organization's current methods.

6. In the construction of Integrated Project Support Environments (IPSEs) and other computer-based software engineering tools, there is interest in tool integration. Kronlöf (1993) points out that tool integration tends to focus on the **implementation** of the integration rather than on the **intention** (or motivation) of the integration. They suggest

that tool suppliers tend to overlook the methodological issues of the engineering process to be supported, and they contend that method integration is a prerequisite for successful tool integration. Brown and McDermid (1992) classify method level integration as the highest level in their classification scheme for tool integration, and point to some current research projects which are examining the way in which this level of integration can be provided in an IPSE.

7. There is increasing interest in integrating software engineering or information systems development methods with methods from other disciplines. For example, Brown and McDermid (1992) point to the need to derive information to support management processes from the technical information produced by software engineers. This implies an integration of management methods with software engineering methods. And Fencott and Hebbron (1994) explore how to combine the traditional methods of safety engineers with those of software engineers.

The above list may not be exhaustive, but it serves to indicate the range of current method integration work.
In order to critically evaluate the method integration work being carried out, we asked the following types of questions: How do we decide which methods to combine? How can we describe the process of method integration? Do we need a meta-method: a method for method integration? How do we evaluate the results of method integration? For example, does the SSADM and Z integration work in SAZ (Polack, Whiston and Mander (1993)) improve upon the Yourdon and Z integration work of Semmens and Allen (1991)? What criteria do we use to answer that question? Where a new method or new version has been developed, is the new method well-integrated? What do we mean by 'well-integrated'? Are some methods more suited to tailoring than others? Why? Do the types of method integration required depend upon the circumstances or context of a project? Is there a contingent approach to method integration?

We concluded that a framework was needed to help in the understanding of method integration :

(i) reactively - where method integration work has been undertaken, to allow analysis and critical evaluation of the method integration work, and

(ii) proactively -- to support the tailoring and combination of methods as required.

Such a framework will assist our understanding by identifying the different types of method integration, the significant features of each type, and criteria for 'good' design of each type. This will then allow us to analyse and evaluate actual instances of method integration work.

Definition of method

Before we can address the issue of method integration, we need to define our concept of a method. Kronlöf (1993) defines a method as having four parts:

1. An underlying model which provides a conceptual representation of the product of a method. It is the class of objects represented, manipulated and analysed by the method. Most system engineering methods in fact incorporate more basic methods (each with their own model) in a 'uses' or 'inherits' hierarchy. As a method increases in repertoire and scope it will have a collection of inherited underlying models. Kronlöf still speaks of a method's single underlying model, even though that 'model' may actually be a set of models.

2. A language, which is the concrete means of describing the product of the method, the user interface to the underlying model of the method. It is used to describe the objects represented, manipulated and analysed by the method, and is the concrete counterpart of the underlying model. As with the model, Kronlöf still speaks of the method's single language, even though that language may in fact be a set of languages.

3. Defined steps and ordering of those steps, the process model of the method.

4. Guidance for applying the method, which typically takes the form of informal text in manuals, handbooks, guides etc.

A method can itself be made up of other methods, each of which has the same four constituent parts.

In the information systems development world (as opposed to software engineering) 'methodology' tends to be used. Olle et al (1991) define a methodology as:
'A methodical approach to information systems planning, analysis, design, construction and evolution.' They also say, 'The concept of modelling is inherent in any information systems methodology.'

Jayaratna (1993) defines a methodology as:

> 'An explicit way of structuring one's own thinking and actions. Methodologies contain models and reflect particular perspectives for thinking about 'reality' based on their embedded philosophical paradigms. A methodology must show 'what' steps to take, 'how' those steps are to be performed and most importantly 'why' [the rationale] the methodology user must follow those steps and in the suggested sequence.'

This definition reflects the fact that not all information systems methodologies are in the scientific or engineering paradigm, and hence their underlying philosophy is of significance.

In this paper we have chosen to use 'method' in place of 'methodology'(if only for the pragmatic reason that 'methodology integration would be too much of a mouthful). We have used Kronlöf's concept of a method, adding 'philosophy' as a fifth constituent part of a method. We also note that Kronlöf describes the physical nature of 'guidance', whereas we feel the logical content is of greater importance. Indeed it is through examining the guidance content that one can often discover the unstated, underlying philosophy of the method designers and their method.

We also feel that the input or initiation trigger of a method, and the final output are of significance. For example, the input to SSM is a 'vague feeling of unease' (Checkland (1981)), in contrast to other methods which take as input a defined problem or a requirements specification. The output of SSM is intended to be consensus about feasible and desirable change: structural, procedural or attitudinal. Consensus about procedural change could be the input to a more engineering-oriented methodology (Miles (1988)). The inputs and outputs of a method are shaped by the underlying philosophy.

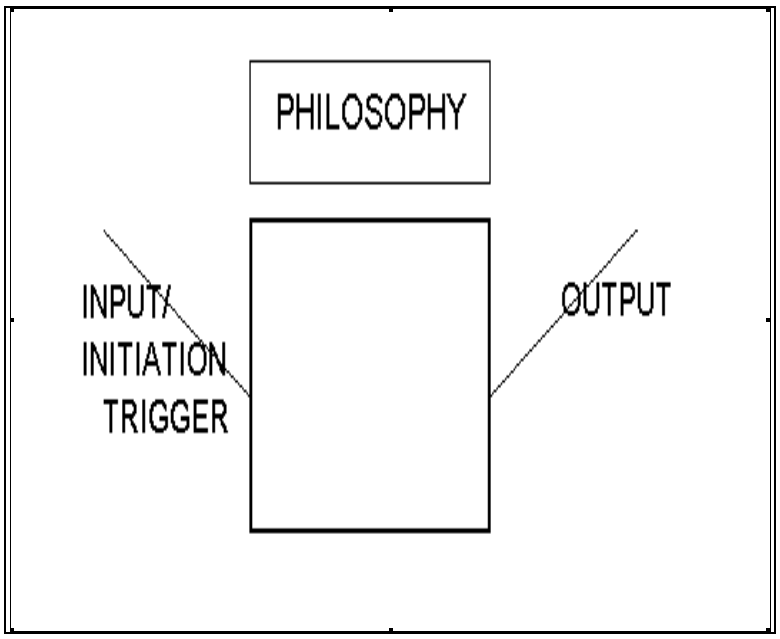

Figure 1

Our concept of a method therefore has seven constituent parts: input, output, model (set), language (set), steps, guidance and underlying philosophy. We have found it convenient to model a method using two levels. At a strategic level, when first considering the use of a method, we see its input, output and underlying philosophy, and the rest of the method can be thought of as a black box (see Figure 1).

When we actually use a method or are otherwise concerned with its detail, we look inside the black box to see the model, language, steps and guidance (see Figure 2).

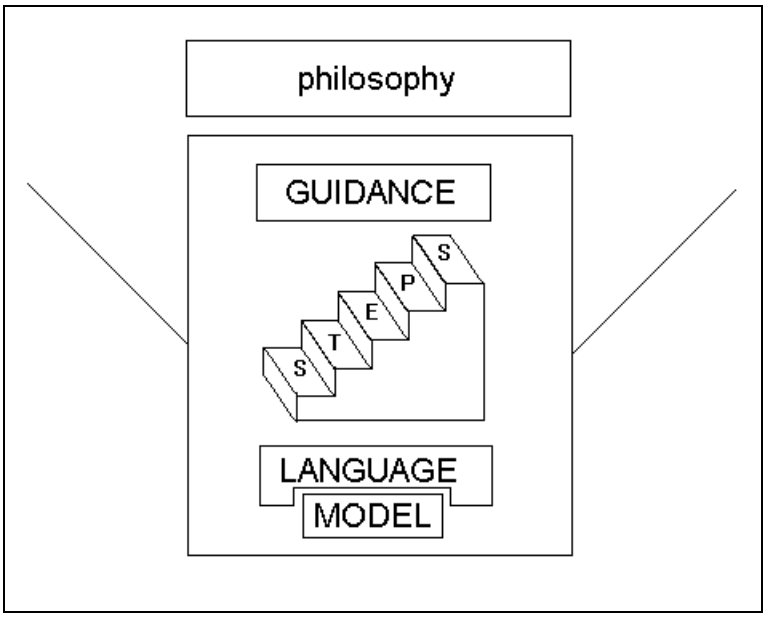

Figure 2

We shall use this concept of a method to structure our outline 'method of method integration' and method integration framework.

Framework for method integration - philosophy and input

The highest level view of our framework uses the model of a method (already defined) to analyse a method for method integration (a meta-method). Firstly, we shall examine the view shown in Figure 1, namely the philosophy, inputs and outputs.

The underlying philosophy of our meta-method is based on the following assumptions:

(i) methods can and must be tailored and/or combined,
(ii) methods can be improved or extended by the incorporation of other methods.
(iii) a theory of method integration is needed as a foundation for tool integration.
(iv) to carry out any of the above successfully, we must be able to
    a) analyse what is done and
    b) critically evaluate what is done.
(v) the study of method integration also increases our knowledge of individual methods.

In order to increase our understanding of method integration, we have identified five broad categories of method integration, classified on the basis of their inputs or initiation triggers: method-oriented, project-oriented, tailoring-oriented, reuse-oriented and tool-oriented. These categories (summarised in Table 1) need not necessarily be discrete ie more than one category may be applied to a particular integration.

Table 1 - Categories of Method Integration

Integration Category	Input / Initiation Trigger	Problem-solver	Output
Method-oriented	2 or more methods (which complement each other)	method designer	a *new* method seeking applications
Project-oriented	project situation requiring the use of 2 or more methods	method consultant / project manager	a situation-specific integration
Tailoring-oriented	project situation requiring the tailoring of a single method	method consultant / project manager	a situation-specific integration
Reuse-oriented	project situation requiring the use of ³1 method **and** the existing products of ³ 1 other method	method consultant / project manager	a situation-specific integration
Tool-oriented	need for two or more tools to be integrated	tool/interface builder	requirements specification for method for integration of tools

The method-oriented category describes the approach of many methods designers over the last fifteen years or so. The method designer perceives that two or more existing methods can complement each other in some way, and aims to produce a new *improved* method. (Of course, the new method may also incorporate additional new ideas and concepts). The project-oriented, tailoring-oriented and reuse-oriented categories have in common that they are likely to be of interest to a method consultant or project manager. Moreover, their outcome is a situation-specific integration ie the integrated method is based on contributing factors (Olle (1991)). The integration may just apply to one specific project or may be reused for other projects with similar contributing factors.

The project-oriented category applies to a project situation where there is a perception that two or more **methods** will be needed, whereas in the reuse-oriented approach the trigger is that there is a need to use existing **products**, in conjunction with a currently used method.

The tailoring-oriented category applies to a project situation and a single method, with the perception that only parts of the method are of relevance.

In the tool-oriented category, there is a need for the actions and outputs of two or more software tools to be integrated. The integration may occur within a software development environment, and so be of interest to the builder of the environment. Alternatively the integration may be externally controlled [simply requiring the outputs of one tool to interface to the other tool (eg Brown and McDermid's 'carrier' or 'lexical' level)] and so be of interest to an interface builder.

At this point, it would seem logical to look inside the black box of our method for method integration (shown by Figure 2), to examine the guidance, steps, model and language. However, this lower-level analysis proves difficult unless we first have a clear understanding of what our method for integration produces ie the *integrated method.*

Framework for method integration - output (the integrated method)

In order to understand the integrated method, the framework uses both levels of the model of a method. The philosophy of the integrated method is inevitably affected by the philosophy of the problem-solver, and will incorporate the philosophies of the participating methods. Hence, we propose that the philosophies of these methods must be examined. Where the philosophies clearly conflict eg one is within an objectivist paradigm and one is within a subjectivist paradigm (Hirschheim and Klein (1989)), the nature of any 'philosophical shift' (Jayaratna(1992)) in the steps of the integrated method must be explained and justified in the guidance.

The inputs to and outputs from the integrated method will be determined, to a large extent, by those of the participating methods. Where the input to a method is not wholly from the problem environment, in many cases it will be an output from one of the partners in the integration. For example, SAZ (the integration of SSADM and Z - Polack, Whiston and Mander (1993)) can be used as a quality audit of part of SSADM. In this case, SSADM commences as usual with a project initiation. The input to SAZ is the chosen Business System Option, produced by SSADM Requirements Analysis. The integration (SAZ) takes place at the Requirements Specification stage of SSADM, involves an error feedback loop, and has as its output an improved SSADM Requirements Specification.

Anticipated features of the guidance, steps, models and languages for each of the categories of integrated methods are summarised in Table 2.

Table 2 - Analysis of Categories of Integrated Methods

Category	Guidance	Steps (expressed as process model)	Model/Language
Method-oriented	detailed	new, detailed process model (using elements of process models of participating methods)	complex area needing further work
Project-oriented	high-level, supplemented by reference to guidance of participating methods	high-level eg project plan, where steps reference documented steps of participating methods	set of models, languages is subset of union of set of models, languages of participating methods
Tailoring-oriented	high-level	high-level elements of process model are a subset of elements of process model of method being tailored	subset of models (set), languages (set) of method being tailored
Reuse-oriented	as for project-oriented except optional reference to guidance of **reused** method	as for project-oriented except that process model of reused method is irrelevant	as for project-oriented except that for the reused method, only the aspects of models, language visible in output are relevant
Tool-oriented	may be high-level for interfacing of discrete tools or enforced as rules or available as help in IPSE	as for project-oriented or computerised process model captured in IPSE	dependent upon intention of integration and level

For method-oriented integrations, the guidance is likely to be detailed and supported by a new process model which details the steps involved in the method. The new process model is likely to use elements of the process models of the participating methods. The possible relationships of the models and languages of the integrated method to those of the participating methods are numerous and are dealt with in Chapter 12 of Kronlöf (1993). We do not propose to deal here with that complex area, except to say that when structured and formal methods are combined, the method integration is essentially a translation between one method's language and another's.

For project-oriented integrations, the guidance is likely to be at a high level only, since it may only ever be relevant to the current project. It is likely to make reference to the guidance of the participating methods. Similarly, the process model will be at a high level only, making reference to the documented steps of the participating methods. The set of models and languages for the integration is a subset of the union of the set of models and languages for the participating methods.

For tailoring-oriented integrations, again the guidance is likely to be at a high level. It will refer to the guidance provided by the method. The process model is also likely to be at a high level, and its elements will be a subset of the elements of the process model of the method being tailored. The model and language will also be a subset of the tailored method's model (set) and language (set).

For reuse-oriented integration, guidance on the reused method is of limited benefit and may not even be available. The guidance for the integrated method is likely to refer only to the guidance of the current method. The process model of the reused method is irrelevant, as are any models and languages that do not appear in the reused method's output.

For tool-oriented integration, where tools are integrated at Brown and McDermid's 'method' level, the guidance should be available as help within the IPSE and the process model should also be incorporated within the IPSE (eg enforcing a sequence of activities). For lower levels of Brown and McDermid's classification, guidance will be required, but could be paper-based, and the process model will not be captured within the IPSE. For all of Brown and McDermid' levels of tool integration, the significant features of the guidance, steps, model and language of the integration depend upon the intention of the tool integration. We suggest that those attempting to integrate tools must first define what type of method integration they are trying to implement.

Conclusion and Future Work

In this paper, we have reviewed some of the current work in method integration; proposed a model for a method; and justified our interest in a framework for understanding method integration. We have begun to build such a framework. The framework uses the proposed model of a method to analyse a 'method of method integration' at the level of inputs, outputs and philosophy. Inputs to and outputs from the method for method integration are examined in more detail by considering five broad categories of method integration. The proposed model of a method is used again, to analyse the output from the method for method integration, since the output is itself a method.

Further work needs to be done on the contents of the black box of the method for method integration. It remains to be seen whether the lower levels of the proposed model are appropriate for analysing such a method, whose concern is the *analysis* of the requirements for an integration and the *design* of an integrated method. As we have seen, the developers may be method designers, project managers, method consultants or tool builders.

> 'The process by which the model is developed combines intuition and judgement based on experience in building similar entities, a set of principles and/or heuristics that guide the way in which the model evolves, a set of criteria that enable quality to be judged, and a process of iteration that ultimately leads to a final design representation.' Pressman (1992)

Pressman's views on software design fundamentals, together with an exploration of the needs of the users, may help us in our consideration of the design part of the method for method integration. Guidance suitable for each of the categories of methods for method integration may emerge eg guidance on how to conduct the tailoring process. It may also be useful to follow the example set by Brown and McDermid (1992), and apply the design criteria of coupling and cohesion to the integrated methods output from methods for method integration. We shall also examine specific examples of method integration, in order to refine and improve the current framework.

References

Avison D.E., Wood-Harper A.T. (1990) 'Multiview. An Exploration in Information Systems Development', Blackwell Scientific.

Brown A.,McDermid J.A. (1992), 'On integration and reuse in a software development environment', Chapter 11, 'Software Engineering Environments', McGraw Hill.

CCTA (1989) 'COMPACT Management Summary, Version 1.1, Issue 1', CCTA.

Checkland P B(1981), 'Systems Thinking, Systems Practice', John Wiley & Sons.

Constantine L.L., Yourdon E. (1979), 'Structured Design', Prentice-Hall.

Crinnion J (1992), 'The Evolutionary Development of Business Systems', IEE Colloquium on Software Prototyping and Evolutionary Development, London.

Downs E., Clare P., Coe I (1992) 'Structured Systems Analysis & Design Method, Application & Context', Prentice-Hall.

Episkopou D.M., Wood-Harper T (1986), 'Towards a Framework to Choose Appropriate IS Approaches', The Computer Journal, Vol.29 No. 3.

Eva M (1992), 'SSADM Version 4: A User's Guide', McGraw Hill.

Fencott P.C., Fox M., Galloway A., Lockyer M.A., O'Brien S.J. (1993), 'Experiences with the integration of structured and formal methods for real-time systems specification', Software Engineering and Its Applications, 6th International Workshop, Paris, Conference Tutorials and Exhibition EC2, Nanterre, France.

Fencott P.C., Hebbron B (1994), 'The application of HAZOPS to requirements models for control systems', Paper accepted for Safecomp 94, Annaheim, California.

Hirscheim R, Klein H.K. (1989), 'Four Paradigms of Information Systems Development', Communications of the ACM, vol 32 no 10, October 1989.

Jayaratna N.(1992), 'Should we link SSM with Information Systems!', Systemist, vol 14 no 3, August 1992.

Jayaratna N.(1993), 'Methodological Challenge for Information Systems', British Computer Society Information Systems Methodologies Specialist Group, Proceedings of Conference on the Theory, Use and Integrative Aspects of Information Systems Methodologies, Edinburgh.

Kronlöf K (1993), 'Method Integration, Concepts and Case Studies', John Wiley & Sons.

Larsen P.G., Plat N.,Pronk K, Toetenel H (1991), 'SVDM: An Integrated Combination of SA and VDM', paper given at Methods Integration Workshop, University of York, June 1991.

Lewis P.J. (1993), 'Culture and IS planning present new challenges for data analysis and modelling', BCS Information Systems Methodologies Specialist Group, Conference Proceedings, Edinburgh.

Merali Y. (1992), 'Analytic Data Flow Diagrams: an Alternative to Physicalism', Systemist, vol 14 no 3, August 1992.

Miles R.K.(1988), 'Combining 'Soft' and 'Hard' Systems Practice: Grafting or Embedding?', Journal of Applied Systems Analysis, vol 15.

Mumford E.(1983), 'Designing Participatively', Manchester Business School.

Olle T.W. et al (1991), 'Information Systems Methodologies - A Framework for Understanding', Addison-Wesley.

Polack F., Whiston M., Mander K. (1993), 'The SAZ Method Version 1.1', University of York.

Poulin J.S., Caruso J.M., Hancock D.R. (1993), 'The business case for reuse', IBM Systems Journal Vol 32 No 4.

Pressman R.S. (1992), 'Software Engineering - A Practitioner's Approach, McGraw Hill.

Semmens L. & Allen P. (1991), 'Using Yourdon and Z: an Approach to Formal Specification', Proc. of 5th Annual Z User Meeting, Oxford, Springer-Verlag.

Sommerville I (1989), 'Software Engineering', Addison Wesley.
Yourdon E (1989), 'Modern Structured Analysis', Prentice-Hall.
Yourdon E (1993), 'Yourdon Systems Method. Model-Driven Systems Development', Prentice-Hall.