LiveMethod: a model and infrastructure for empowering methods use in organizations

Author: Graham McLeod
University of Cape Town Private Bag Rondebosch 7700 South Africa.
Inspired I.T. Consulting, Training & Research Box 384 Howard Place 7450 South Africa
Telephone: +27 21 531 5404 (bus) +27 21 650 2280 (univ) +27 82 578 1834 (cell)
e-mail mcleod@iafrica.com

Words: 3460 excluding abstract

LiveMethod: a model and infrastructure for empowering methods use in organizations

Graham McLeod

University of Cape Town Private Bag Rondebosch 7700 South Africa.
Inspired I.T. Consulting, Training & Research Box 384 Howard Place 7450 South Africa
Telephone: +27 21 531 5404 (bus) +27 21 650 2280 (univ) +27 82 578 1834 (cell)
e-mail mcleod@iafrica.com

Abstract

The effective use of methods in real organizational contexts is frequently hampered by factors including the complexity and size of comprehensive methods, difficulty choosing and applying methods correctly to diverse projects, communicating the necessary method fragments to practitioners in a user friendly way and managing the rapid evolution of the methods to meet changing business and technical require ments.

Based upon earlier work in modeling of methods as an evolving set of inter-related meta data containing definitions of method products, tasks and resources and prototypical tools for the capture, representation, integration, evolution and management of methods, this paper presents a model-based environment in which these concepts are extended to manage the deliverables themselves produced in accordance with the meta descriptions. The proposed approach also embodies the use of intranet technology to replace earlier closed tools as a means of making the methods and their products "live" to project teams and practitioners.

Introduction

Organizations adopt methods and structured planning processes to gain a variety of benefits, including: improved communication, higher quality, improved productivity, better utilization of resources, more consistent delivery, and process management for continuous improvement.

Unfortunately, these benefits are seldom consistently realised. Reasons for this cited by various authors (Avison, 1990; Olle, et al, 1988; Avison & Wood-Harper, 1990/1991, Hackathorn & Karimi 1988, Avison & Fitzgerald, 1988) include the following:

- *Methodologies are difficult to learn*
- *Method suitability is contingent*. i.e. certain methods are appropriate to certain project types, organizational circumstances or culture, project team skills, implementation technologies and other factors
- *Methods lack integration across the lifecycle*. Thus practitioners tackling the full lifecycle are forced to adopt and integrate several approaches
- *Technologies are evolving rapidly*. Practitioners do not have time to become expert in one approach before another one is advocated
- *Where a contingency approach is applied*, the techniques may better suit the project, but many of the *benefits of standardisation may be lost*. Ease of communication across project teams and with management is reduced. Improved performance resulting from increasing experience and tuning of the methodology to the environment and organization is curtailed

Methods engineers (Odell in Brinkkemper, Lyytinen and Welke, 1996) face further problems which include:

- *Lack of standards* for representation of methods, making comparison and integration difficult
- *Tools are lacking* to assist in the evaluation of methods and in rapidly evolving methods to meet changing technologies and requirements without sacrificing integrity and rigour
- *Inadequate* timely *feedback* obtained on effectiveness problems encountered

Consequently, methods engineers require more powerful approaches, techniques and tools to assist them in:

- *Representation* of methods in a comprehensive, coherent and consistent way. Techniques must be competent to handle a wide variety of methods spanning many disciplines
- *Integrating* methods and method fragments from a variety of sources to allow production of comprehensive approaches
- *Measurement* of methods under consideration for inclusion in the portfolio, or produced by the methods engineer
- *Evolution* of methods to improve their quality, efficiency and efficacy

Organizational requirements

In addition to the methods engineering requirements, we can identify further requirements for the success of methods development and usage in organizations. These include:

- *Rapid development* is required to track the fast changing technological environment and rapidly changing business climate
- *Wide, instant deployment* is required to quickly communicate the new approaches to the (often distributed) user community
- *Consistency across work groups and project types* is required to allow a measure of stability and to reap benefits of improved communication and higher skills in subsequent iterations through a process
- *Controlled evolution* so that production teams are insulated from turbulence of market forces and research failures. We thus need ways to accommodate the current official method, while also developing the "next generation" method

Practitioner/project team requirements

To be practical and usable in the pressured project team environment, where the focus is on delivery rather than concepts, methods must meet several criteria:

- *Tailored, manageable subsets* or fragments must be available. A comprehensive method for the whole organization for all project types can become very large. This is intimidating to the practitioner, with the result that methods are avoided or ignored. We need powerful ways to help a practitioner select relevant fragments of the method pertinent to the person or group who will use them and the type of project at hand
- *Participation* of practitioners in methods evolution is vital if we are to obtain their full "buy in" and support
- *Delivery should be seductive*. We need to create an environment and delivery mechanism which is seductive and draws practitioners to it
- *Methods should really add value* and assist in performance of daily activities. They should be a resource to empower staff, not a corporate rule book

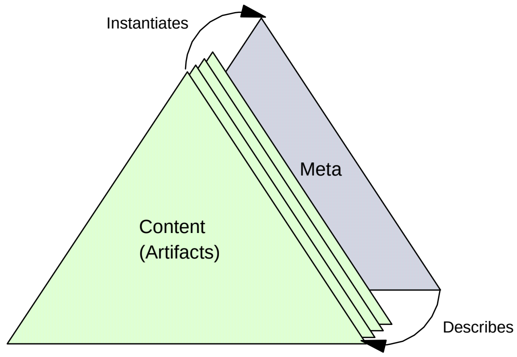
The problems outlined above have been the focus of our research for many years. The author previously developed a generic method model (McLeod, 1993, 1995). This has proven capable in the areas of representation, integration and management of the evolution of the method definition. More recent work has addressed the deployment and organizational issues involved in production use of the concept. Concurrently, a colleague has successfully used intranet technology to describe and disseminate a customized object method within a commercial organization. We have also gained experience in the use of an intranet in the management of many (35) concurrent student system development projects. These experiences, and work on the rapid evolution and dissemination of strategy and guidelines within a dynamic client organization, have led to our current project which uses the previously defined method model as a formal base for holding planning and methods information in a dynamic intranet environment, making method and strategy evolution and use a living widespread daily activity within a corporate environment.

The Model

The model (Figure 1) comprises two major layers: a Meta layer, which contains the method itself as data and provides operations in support of method engineering, and the Content layer, which holds artifacts produced in accordance with the method and provides operations in support of their management and maintenance.

Meta Layer

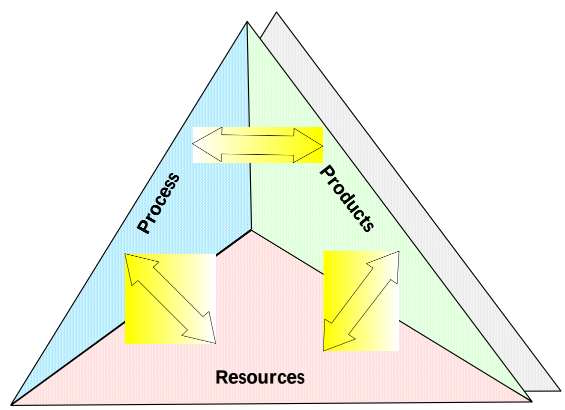
Common aspects of methods were abstracted and distilled to just three generic high level types, viz. Products, Tasks and Resources. It was found that all methods of whatever variety from all disciplines shared these characteristics. We refer to these high level abstractions within our model as *facets*. These are explained further below (Figure 2):



Content and Meta Layers in the Model

Figure 1

- *Products* are tangible artifacts produced by following the method. They include text documents, forms completed, structured record data, graphical models, formally expressed algorithms, rules and derivation formulae. Examples would include:
  - An organization chart
  - An entity relationship model
  - A Jacobson "use case"
  - A data dictionary entry defining a record structure



Inter-related Method Model Facets

Figure 2

- *Tasks* (Process facet) are work units or steps performed in following the method and usually resulting in the production of products. Tasks can be course grained, e.g. a development project, or fine grained, e.g. adding keys to relations in data modeling
- *Resources* are required to perform the tasks. They include two major categories: Human resources (people with their particular skills) and other resources including: Money, time, equipment, software, standards

It was found desirable and useful to decompose these facets in a top down fashion, as well as to have the capability to collate or group lower level items into larger assemblies or groups. Examples of the need to do this include:

- In the *product* facet, a data model might be comprised of a graphical entity model, a data dictionary entry for each entity on the graphical model and a dictionary entry for each data item specified within the entity definitions
- In the *task* facet, we could see a project broken into phases, which in turn are broken into tasks
- For *resources*, we could see a split into human and other resources, with the former split into types and levels and the latter divided into material resources, financial resources and software resources

Thus we require the ability to represent a *hierarchy* within each of the facets of the method model.

In addition to the hierarchical (or composition) relationships, we may also want to show *dependencies*, or network relationships between items in the same facet (Figure 3). This is necessary to define the relationship of the current item to, for example:

- *Input products* required to produce a later assembly or more refined product. E.g. the functional requirements of a system and the data model used as inputs to the production of a prototype interface for the function
- *Destination products* which will use the current item as an input in their production. Example: A business rule which will affect the way a later program design is specified

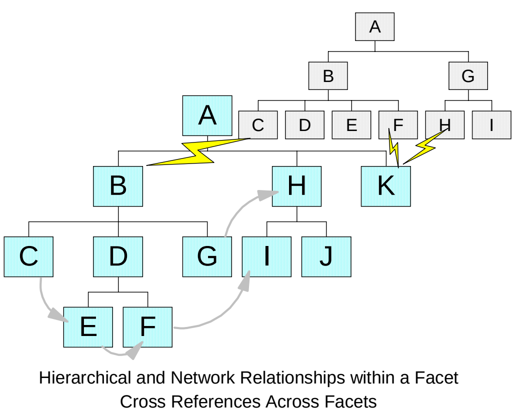


Figure 3

It is further necessary to be able to *cross reference* or inter-relate nodes of different types from the various facets. Examples of this include:

- Linking a task to the product(s) it may produce: a link from the *work* to the *product* facet
- Linking a task to the people assigned to perform it: a link from the *work* to the *resource* facet
- Linking a product to the tasks required to deliver it: a link from the *product* to the *work* facet

These links should be able to be made at arbitrary levels between the facets and to be "many to many" in either direction. E.g. a third level task might be linked to several products at various levels in the product facet; likewise, a single deliverable in the product facet may be linked to several tasks, some of which are summary tasks, in the work facet.

Structure of Nodes in a Facet

The internal structure of a node within a facet includes content variables and meta information. The node must know where it is in the hierarchy and the network, to which other nodes it is related in other facets and its status. In addition, the node must be capable of holding the product representation, or task/resource description as appropriate. Space does not permit a full description of the design solutions in this paper.

Meta information for a node in the method description or content includes:

- *State* of the object. Progress to later stages within a method frequently requires the completion (or advance to a designated state) of a prerequisite task or artifact. An example would be that we want a Data Model in a *normalized* state before proceeding to do physical database design

- *Owner* of the object. Person who should be approached when explanation of, or changes to, the object are required
- *Version*. There may be several versions of an object which are concurrently valid: one could be the current “published” guideline for practitioners, while another is the draft for a methods engineering group
- *Date* last updated
- *Status*, which is an indication of progress from initial concept or personal opinion, through to accepted official policy
- *Security level or category* which will allow the restriction of alteration or viewing rights to sensitive information to certain user communities
- *Relationships* previously mentioned. These will include: parent node, list of child nodes, input nodes, destination nodes and cross references to other facets. Content nodes should have a link to the meta layer, where the techniques and structures necessary to produce this content can be obtained or reviewed. Meta nodes might have a list of instances of artifacts of their type. E.g. the meta node with the definition of what an object model looks like could hold a list of actual object models

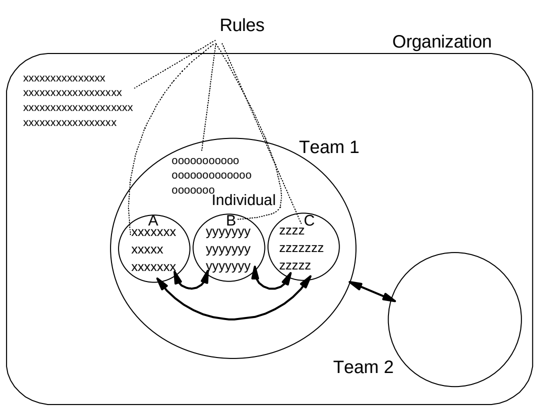
The use of states allows an important methodological innovation much advocated in recent methods, viz. iteration: specified by requiring that a product reach a certain state before a dependency is satisfied. This permits cycling back from later tasks to an earlier one until a required outcome is obtained.

Operations on the Model

To support competent methods engineering, we require a variety of operations on the model. Obviously, we require the normal operations to capture the method definition, view it, modify it, and where required, print it. We will not discuss these, but rather concentrate on additional operations that add value to the methods engineering process.

Various operations can be usefully performed on the completed model. These include

- *Clustering* of method tasks and rules related to a deliverable. This is analogous to the packaging of methods with the instance variables in an object design, or the grouping of rules with affected data in frame based expert systems. It has the advantage of allowing us to determine useful and small method fragments which can be easily learnt and applied by organizational units (Figure 4)



Rules can be clustered around areas where they are relevant

Figure 4

- *Automated extract of relevant method components for a resource type*. This can allow a person operating in a certain role to quickly find and use information relevant to their position, without being overwhelmed by the greater method, but still remaining compatible with the corporate approach and the techniques used by colleagues performing different tasks
- *Customized project plans* which are consistent with the corporate policies, strategies and goals and which take into account all necessary dependencies, resources and requirements to be comprehensive can be generated from the method definition (Figure 5). This is accomplished by selecting the required products for the current product then recursively identifying their sub-components and the prerequisites to these. Once this is done for the

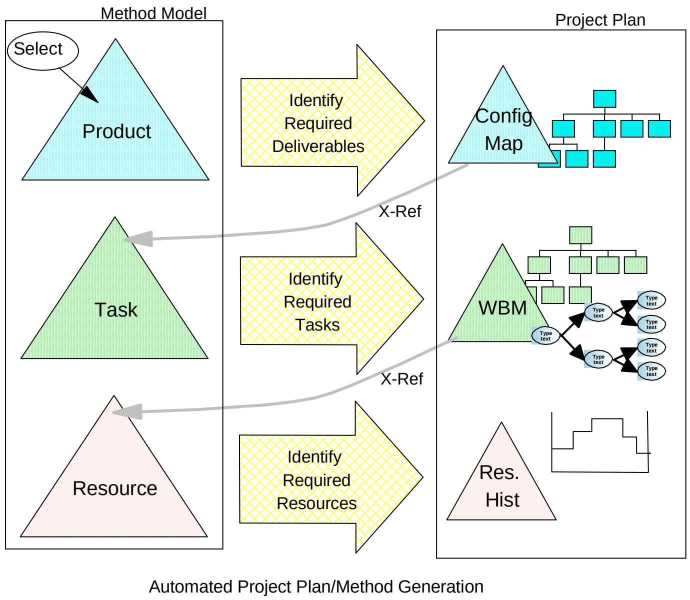


Figure 5

product facet, the cross references are used to identify related activities in the work facet. This initial set is used to perform a similar recursion there. Finally, cross references again determine the related set of resources, where a final recursion is carried out. From this process, we can obtain

- a draft *configuration map* for the artifacts which the project will produce
- a *work breakdown structure* for the tasks involved
- a *list of resource types* and their assignment to tasks

These can be used by the project manager to generate the initial project plan, and subsequently to track the project by replacing generics with specific information and actuals as events unfold

- *Evaluation of method complexity* can be derived using the Method Points technique developed by the author (McLeod, 1997). If the method is expressed in the model, the count could be done automatically

Production Strength Requirements

Two prototype tools were built to prove the viability of the method model. The first (McLeod, 1995) was successful in proving the capabilities for representation, supporting methods integration and evolving methods, as well as the generation mentioned above. The second concentrated on using a similar model for the management of actual project deliverables and supporting project designers.

Based on these and experience in industry, we saw the need to satisfy a variety of characteristics in evolving the concepts to a production strength environment. We also wanted to support distributed workgroups. Criteria thus included:

- *Openness*. The tool should be able to interoperate with a wide range of products including browsers, databases, spreadsheets, graphics packages, project management software, etc.
- *Ease of access.* Little effort or retraining should be required on the part of a developer or practitioner to access or use the system. It should thus make use of common graphical user interface standards, be relatively intuitive to use and integrate well into a desktop environment
- *Ubiquitous*. If we wish to gain widespread methods usage across an organization, then the method information must be universally available
- *Affordable*. If we plan to make the facilities ubiquitous, they must be inexpensive
- *User friendly*
- *Connect vision, strategy, architecture, products, discourse.* This is a need identified from our work with dynamic organizations where paper strategy formulation cannot keep up with business activity. There is a need to keep the vision, strategy, architecture, product choices and discussion regarding all of the above in a form which can be rapidly evolved, communicated and richly interrelated

- *Maintenance* needs to be manageable and, as far as possible, automated. If the burden is too onerous, it will simply not be done in the face of other business pressures. We must find ways in which to make the maintenance as natural and transparent as possible
- *Distributed*. The players involved in strategy formulation, planning and project work are frequently distributed geographically as well as temporally. The infrastructure must thus provide remote access and rapid application
- *Multiple authors* should be supported, since strategy formulation and methods evolution can be widely distributed in the modern organization. *Ownership* of various fragments of the meta and content layers must be identifiable and manageable for individuals and groups
- A powerful realtime or *asynchronous communication mechanism* is required which will allow wider participation in the formulation of strategy, development of methods and techniques and in the use of these in support of organizational goals
- *Change and impact management*. It must be possible to estimate the impact of change and to manage change to both meta and content information
- *Quality awareness and continuous improvement* must be supported and facilitated so that the plans and methods can be evolved and continuously improved in the light of new theories, technologies and practical experience
- *Innovation and reengineering* should be supported in a managed, sustainable way. Without proper impact assessment, these can be destructive. Properly managed, they can provide vital advantage to an organization
- *Security levels* must be supported to prevent unauthorised alteration or access to sensitive or critical information

Potential of Intranet Technology

Looking at the above criteria, it is obvious that this technology has several very distinct advantages and attractions for our purpose (Telleen, 1996):

- *Portability* across a variety of hardware, operating system and browser environments
- *Graphical support* is extensive and competent
- *Ubiquitous availability* can be achieved relatively inexpensively
- The technology is inherently *distributed*
- The need for a communication medium is easily supported by integration with e-mail and could in future be extended to include rich interactions such as video conferencing, audio annotations and the like
- *Automated routing and workflow,* while not inherent characteristics of the current technology, are offered by some products (e.g. Lotus Notes/Domino) and could be achieved with relatively little effort in the absence of these products

Pilot Project

A pilot project was undertaken with a large cellular telecommunications corporation with a focus on the collation, integration and dissemination of strategy and planning information. This will later expand to architecture work and project deliverables. In a matter of weeks, existing strategy documentation and deliverables (budgets, plans, models, presentations, group charters etc.) was gathered, interviews were conducted and the information was merged into an intranet form under the guidance of the methods model. Meta and content layers were created with the latter admittedly containing somewhat fragmented representation, but representative of the state of available artifacts within the corporation. Meta content is still lacking at this stage as well, but is being added to guide further strategy work.

A rich navigation metaphor was created (see Figure 6) which allowed users to browse, connecting top level strategy with planning groups, with initiatives and plans. They could access source documents and relate from a technical architecture for a project (say Data Warehousing) to the business drivers that this would support. A sample screen from the pilot implementation is shown. The left hand area provides familiar web navigation tools, which support moving in the hierarchy. The right hand toolbar provides navigation through the network links (e.g. to sources and destinations) as well as to other facets, and to feedback tools, e.g. Comment. Icons are also provided to navigate between content and meta layer (How To) and to retrieve previous comments and discourse (Issues).

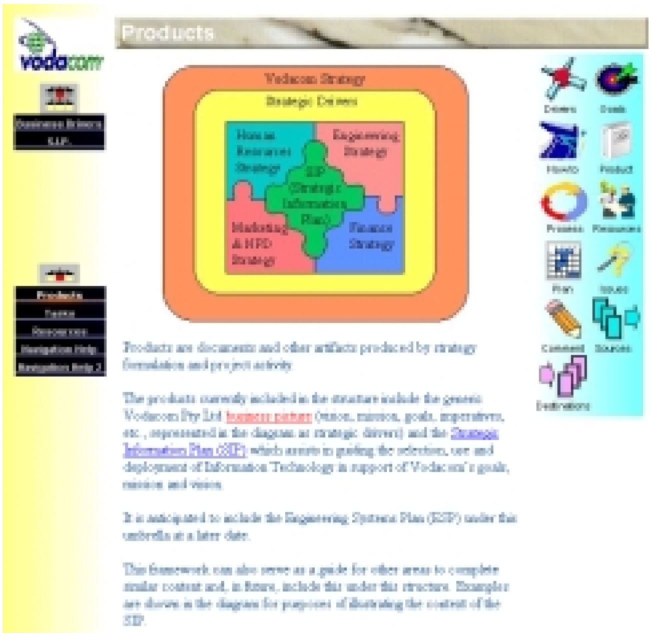


Figure 6

Things Learnt

Initial response from the still limited user community has been extremely favourable. People have become aware of previously unknown strategies and have become involved in debate as to whether they are appropriate, applicable, official etc.

Already, a wider community is participating in the formulation of ideas, policy and planning. Areas of overlapping work and responsibility have been identified, leading to collaboration rather than political battles.

A paper document (the Strategic Information Plan) which was previously confessed to be “an event” created by a few individuals, has now become a process and alive for several work groups. Others at project level not previously aware of such guidance have begun consulting it as a resource.

Placing content and meta into the same structures has been very valuable in highlighting gaps and holes in the corporate strategies in place. Links to meta content provide guidance in how to do something about these holes and what the deliverables could look like when this is done.

Maintenance of the content in a static authoring tool would be prohibitive and we are now engaged on a project to deliver a dynamic environment based on a database and server side processing to manage the site content and meta information consistently in accordance with the method model. We are also looking at Java on the client to provide editing facilities for content directly to authorised users. Further innovations include the unique numbering of all pages, the marking of pages with security levels (mapped to user authorities) and the identification of the status of pages (initial, working, draft, official). More intuitive hierarchy navigation (*ala* Microsoft™ Explorer) is being investigated. In future, we plan to move to a CORBA based client/server interaction and an object database on the server.

We believe that the infrastructure created can have far reaching implications for successful formulation, evolution, adoption and use of strategy and methods within a large corporate context. The environment created also appears to be a viable tool for sustainable business reengineering, but we have yet to prove that.

Key References

Avison DE, Fitzgerald BA, 1995. **Information Systems Development: Methodologies, Techniques and Tools. Second Edition.** McGraw-Hill, 1995

Avison DE, Wood-harper AT, 1990. **Multiview: An Exploration in Information Systems Development,** Blackwell Scientific Publishers, Oxford 1990

Brinkkemper S, Lyytinen K & Welk J (Editors), 1996, **Method Engineering: Principles of method construction and tool support**, Chapman & Hall, 1996

Hackathorn R D, Karimi J 1988, **A Framework for Comparing Information Engineering Methods**, MIS Quarterley June 1988 pp 203-220

Jayaratna N, 1994. **Understanding and Evaluating Methodologies. NIMSAD a Systemic Framework**, McGraw-Hill, 1994.

Jones L H, Kydd C T 1988, **An Information Processing Framework for Understanding Success & Failure of MIS Methodologies**, Information & Management Vol 15 1988 pp 263-271

McLeod G, 1993. **A generic method model for the Representation, Comparison, Integration and Management of Methods**, Proceedings: 1st International Conference on Information Systems, Henley, U.K., 1993.

McLeod G, 1995. **MetaTool: A tool for the engineering and management of methods**, Proceedings: 1st International conference on Meta-CASE. Sunderland, U.K., Jan 1995.

McLeod G, 1997. **Method Points: Towards a metric for method complexity**, Proceedings 2nd International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'97), August 1997, Barcelona

Telleen, Steven L, 1996. **IntraNet Methodology, Intranets and Adaptive Innovation, Intranet Architecture**, Intranet Solutions, Amdahl Corporation, 1996.