A Method Engineering Workbench on the EVA Platform

Graham McLeod

inspired.org, South Africa & University of Duisburg-Essen, Germany
graham@inspired.org

Abstract. Methods should assist organisations to produce desirable results more effectively, efficiently and reliably. Since method effectiveness is contingent on goals, setting, skills and other factors, method engineering is a critical activity: yet it is not well supported by tools in industry. This paper discusses an approach to and experience in creating a method engineering and deployment environment using a commercial enterprise modelling and knowledge management platform, viz.. Enterprise Value Architect (EVA).

The approach involves a competent meta model to describe methods in a generic way. Uniquely, it incorporates goals to shape method development and facilitate method tailoring for purpose, project and practitioner. The platform incorporates a unique "one page portal" for method exploration and tailoring. Quality assurance is emphasised to ensure (i) practitioners appreciate the risks of omitting method elements (ii) project designers, practitioners and reviewers share a common understanding of what is expected and how it will be validated. Challenges in organisational implementation are discussed.

Keywords: Methods Engineering, Methods Workbench, Meta Model

1. Introduction

Organisations adopt methods to produce desirable results more effectively, efficiently and reliably. Methods are intended to guide practitioners to improve quality and productivity and reduce risk. They should also assist in achieving higher consistency across projects and in aiding less experienced resources to perform competently. Practices such as the Capability Maturity Model (CMM) [1, 2] for software development rely upon repeatable processes and their continual refinement to achieve higher levels of maturity, productivity and quality.

Unfortunately, methods are contingent: i.e. they must be suitable or adapted to the situation at hand, the goals of the project, the available skills and resources, tools and technologies and the culture of the organisation, amongst other factors [3, 4, 5]. A different method would be required, for example, to develop a safety critical embedded system (e.g. Air Traffic Control Support) versus a commercial end user web portal. In the case of the latter, intuitiveness of the interface and rapid delivery may be paramount. In the case of the former, safety and data quality are non-negotiable.

Method engineering is often necessary to define, adapt, or combine methods to achieve the required result [6, 7]. This is a highly skilled role that requires a deep understanding of the discipline, the methods, techniques and tools, and the principles, philosophy and cultural issues. It is unfortunately also not well supported by tools in typical industrial settings.

2. Research Problem

As consultants, we were asked to analyse and critique the methods in use for system delivery, package implementation, proofs of concept and technology roll out at a large financial/assurance group. This is a very mature organisation which has exploited information and communications technology extensively for six decades. They have a development group (in the business unit in which we were engaged) of approximately 200 staff. These are augmented by a roughly equal number of strategic partner staff, many of whom are offshore.

Problems being experienced included: poor delivery, poor quality, push back from developers adhering to "agile" approaches [8, 9, 10] against formal use of methods, poor documentation of delivered solutions (leading to problematic and expensive maintenance). In addition, there was poor organisational collaboration across strategic initiatives (business change), programme and project management, and the development organisation.

We conducted an investigation into the existing methods in use. This involved collecting all the relevant lifecycles, method documentation and thorough analysis of the methods documentation contained in a

commercial wiki system (Confluence). We also interviewed management, quality assurance and architecture staff, and representative project sponsors and development / test resources. We captured the method structure into a model and repository to analyse its coverage of lifecycles and concerns and the similarities and differences between various project types and styles of development (primarily “waterfall” and “agile”). This led to extensive findings and recommendations, including:

2.1 Findings

- Methods were extensively (and sometimes conflictingly) documented in the wiki, Ms Office documents and Ms Sharepoint
- Since definition was document based, rather than model based, there were numerous gaps and inconsistencies, as well as redundancies
- The reason why various artefacts were required / produced was not clear
- There were a great many artefacts required of the average project (typically over 100) and many were perceived by the development community as overhead or not adding value (“make work”)
- There was a lack of integration and consistency in definition of lifecycles across responsible business areas including: business change, project management office, development organisation, “business as usual” maintenance team and operations
- It was not clear exactly what was expected for a given type of project at what stages in the lifecycle, leading to conflicts between management, architects and developers
- Many artefacts required were documents, rather than models. This led to imprecise definitions and large volumes of unstructured text, non-standard spreadsheets and diagrams, in turn negatively impacting consistency, quality, maintenance, knowledge management and reuse
- Many tasks were performed at a project level that should rather be done once consistently at a corporate or divisional level e.g. Definition of Domain Model, Choice of Architecture Style or Technology
- Tooling was not prescribed for the production of many artefacts, leading to inconsistencies, incompatibilities and loss of information
- Due to the above factors, method use was inconsistent, leading to an inability to measure performance or implement any continuous improvement towards higher maturity. This was exacerbated when external contractors or outsource partners were involved, since they often “brought their own method with them”

2.2 Recommendations

A large body of recommendations was put forward, grouped into the following major themes:

- Goal Orientation
- Consolidate and Integrate Life Cycles
- Rationalise Artefacts, Update Techniques, Integrate Quality Management Activities
- Organisation and Culture, Move from Document Oriented to Model Driven, Accountability
- Goal derived Metrics and Continuous Improvement
- Implementation Approach

After presentation of recommendations, we were asked by senior management to assist in engineering new methods that would address the issues and improve delivery.

3. Approach

Based upon success we have had internal to our consulting organisation, and in consultation with our sponsor and the client organisation, we decided to follow a goal driven approach to the method engineering. This was to facilitate the following:

1. Reach consensus in the organisation on what should be achieved
2. Identify artefacts that would assist in achieving the various sub-goals
3. Devise lifecycles that would achieve the higher level goals, group the artefacts sensibly depending upon dependencies and address the different styles of project management
4. Reduce the “bulk” of the methods by eliminating redundancy, any artefacts that did not contribute to the goals and duplication across project types

5. Ensure quality by ensuring that all relevant concerns were addressed properly; that all required goals were achieved by a given review point and that all players were "on the same page" in terms of expectation, content, purpose, format and review process

4. Meta Model

A meta model previously developed by the author [11, 12], and employed in our practice for many years already, was adapted by adding principles and goals. We also found it useful to add Discipline, Concerns and Logical Artefact Types to facilitate the analysis of the existing methods. The meta model includes the following concepts, *inter-alia*:

Principles are guiding statements derived from organisation values and strategic intent.

Goals begin with a high level mission to the organisation's customers, and are decomposed into more detailed goals to achieve the overall mission. They take the form of statements including a verb, object and (usually) a qualifying clause. Detailed goals can reflect objectives with a metric. The latter can be recorded for current performance, industry benchmark and desired performance by a designated date (target).

Artefact Types define the kinds of artefacts / deliverables that project work will produce. We had a principle to move the organisation towards more use of models and less of documents to improve conciseness, rigour and reuse of assets. Artefacts can be composed into larger ones. E.g. Requirements may include Functional Requirements, Data Requirements, Non-Functional Requirements. The latter may include various facets, such as Performance, Security, Reliability etc.

Logical Artefact Types were introduced for analysis purposes. They reflect the fact that the same logical purpose can be served by multiple physical Artefact Types. E.g. The logical requirement for a Function Model can be satisfied by the Artefact Type "Function Decomposition Model" or "Use Case Catalogue". Logical Artefacts help to identify redundant or alternate Artefact Types. They are not show in the practitioner views of methods to avoid confusion.

Task Types define the kinds of tasks that project staff will perform in applying the method. These normally produce artefacts.

Risks are dangers that we can face because of omitting certain Artefacts/Tasks or during the execution of a given Task Type.

Quality Checks define how we will verify the presence and correctness of Artefacts. We integrated these with Method Elements as attributes rather than as a separate meta model concept.

Techniques define ways in which a given Task Type can be performed. E.g. The Task Type "Data Modelling" could be performed using the Technique "Normalisation" or "Entity Relationship Modelling".

Tools are things that can help us to perform a Task or produce an Artefact. Most often they are software tools. E.g. The Task Type "Test Component" could make use of the Tool "Regression Test Suite".

Resource Types document kinds of resources that are needed to complete Tasks and Artefacts. They include Job Roles and other resources, such as Reference Models and Reusable Components

Project Types group sets of goals that a particular project profile should achieve. E,g. in broad terms, a Solution Delivery Project should include the Goals "Deliver Working Software" and "Deliver Documentation for Sustainable Solution".

Lifecycle Stage is a way of grouping Artefact Types and Task Types into logical stages of a project and associating content generation tasks and outputs with review and quality assurance activity.

Selected abstractions (e.g. Method Element) allow generalising important relationships, such as hierarchies and dependencies within subtypes.

Note that the meta model also includes coverage to manage the instances of the above concepts relevant to or created by actual projects. These include things like Artefact (and its sub-types), Task, Person (an instance of Role) and State and Metrics. Most of these are grouped in the lower right of Figure 1. The model can thus manage actual project details in association with the methods definition. This leverages the EVA capabilities to store both structured (and if desired, graphically represented) models and documents as well as individual objects (instances of Concepts addressing Concerns) and lists, forms, matrices or visualisations related to these.

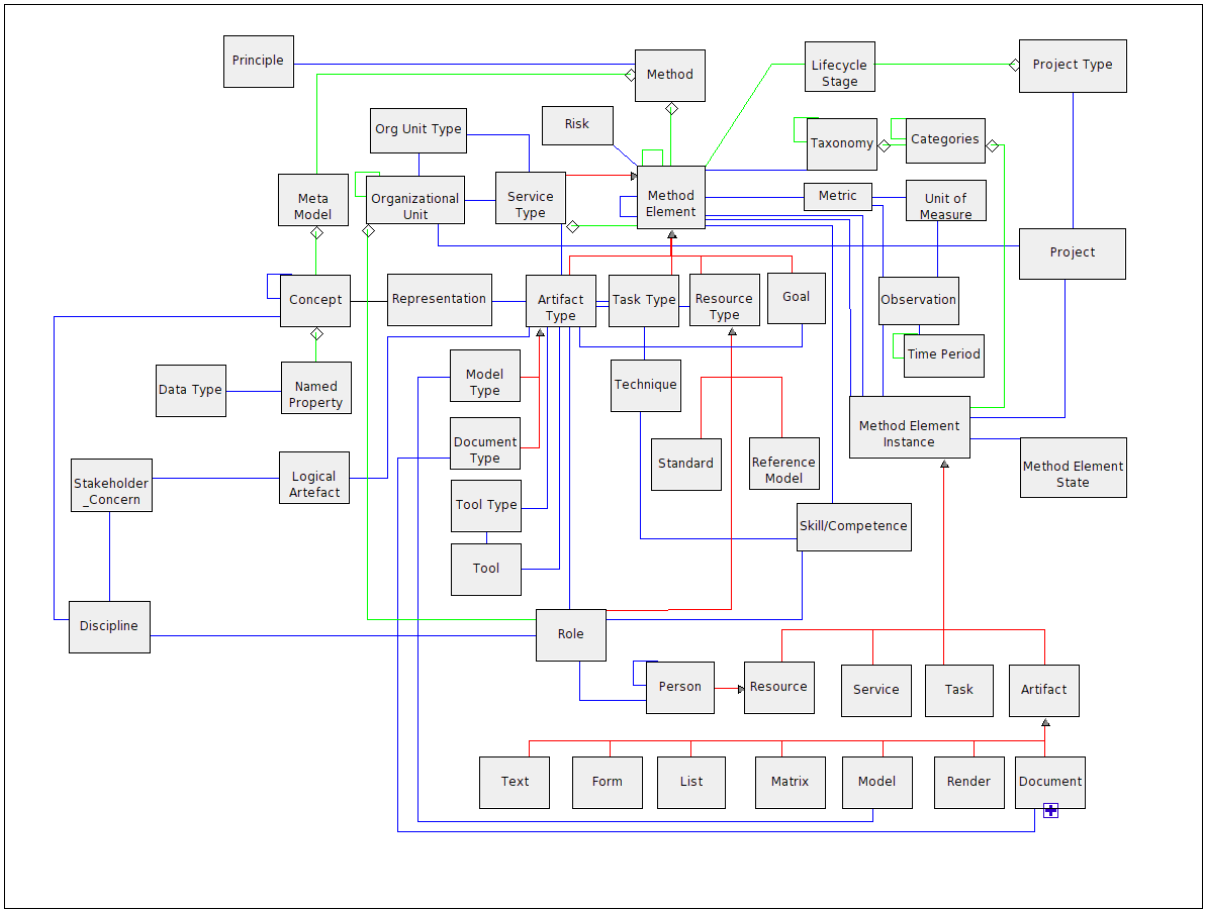

Figure 1 - Methods Engineering and Artefact Management Meta Model

5. Algorithms

A few key algorithms are employed to generate unique method configurations. These have their roots in earlier work by the author [13, 14] but were adapted for the enhanced meta model.

A1. Project Type is selected which leads to selection of Goals, associated Artefact Types, associated Task Types, Associated Resource Types. In each facet, the selection includes recursive addition of descendants within the hierarchy, as well as cross referencing to other facets (and where appropriate, recursion there too). The recursion algorithms must detect and report situations where users may have defined cyclical relationships (e.g. an Artefact Type has a child which becomes its parent).

A2. A Lifecycle Stage can be selected which will select relevant Goals, Artefact Types, Task Types, Resource Types. This is similar to Project Type but with a more limited scope.

A3. An Artefact Type can be selected. This will select any component children, associated Task Types and Resource Types.

A4. A Resource Type can be selected which will select associated Artefact Types and Task Types.

A5. Any method related concept can be selected and "documented". This involves retrieving the detailed properties of the item as well as items directly related to it and their properties and rendering these in a hyperlinked report.

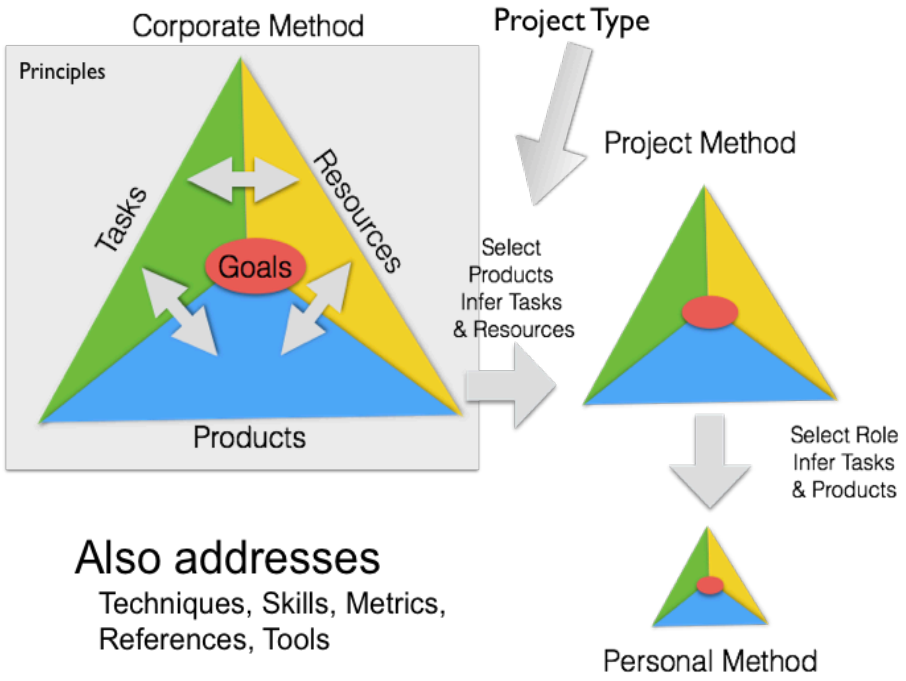

Figure 2 - Method Model Facilitates Specialisation and Fragment Generation

6. Platform

The Enterprise Value Architect (EVA) platform from inspired.org [15, 16] allows user definition of meta models. We use it extensively in our consulting practice and it was already in use in the client organisation for enterprise architecture and application portfolio management. The platform had an earlier implementation of method engineering capability available to us, so it was an easy decision to support our efforts leveraging this.

We defined a meta model, consistent with that shown previously. in the tool. The tool uses the meta model to dynamically generate a wide range of user interfaces and facilities including reports and visualisations. It also provides a portal which exposes selected domains (e.g. Methods) to a community. EVA also supports the storage, indexing, search and versioning of documents. This makes it possible to store examples and actual documents in the repository, linked to the method definition. These documents can be any type of file/content meaningful to the user workstation (e.g. Word, Excel, Powerpoint, PDF, Project Plan, Model from another tool...). Finally, EVA also supports the concept of Templates, which are default documents providing structure, for which content will be provided later. Templates can be identified for Artefact Types. When a user creates a new Artefact, he/she will be given a copy of the template, linked to the new Artefact which can then be populated with the project unique content. EVA also supports attributes for items which are links to intranet, internet or network resources. In this way, various other resources, such as other tool outputs, standards reference sites and the old wiki can be referenced.

As mentioned, EVA provides suitable capture, maintenance, analysis and reporting facilities “out of the box” driven by the meta model and its ability to dynamically generate a variety of interfaces and outputs. However, these interfaces require a “modeller mindset” which was deemed challenging to achieve in practitioners without investment in training. Accordingly, we designed a “single page methods portal” web interface to the method content. Actually, the full interface includes this view as well as a number of reports and generated documents, including a “method overview” which, for a selected project type, documents the lifecycle stages, expected artefacts, associated goals and quality checks that should be applied. These facilities are available to method users from a simple user menu.

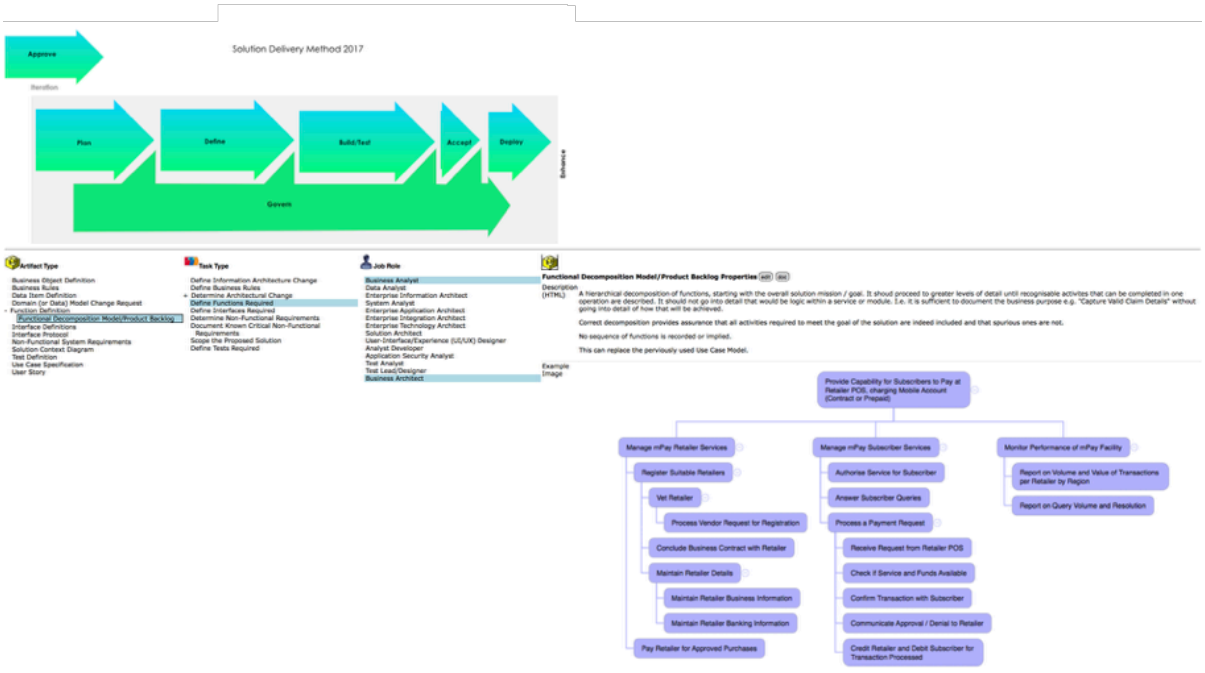

Figure 3 - The One Page Method Browser Web Portal Interface

In Figure 3, the user has selected a Project Type, which determined the Lifecycle Picture to display (arrow diagram). The Stage boxes on this diagram are hotspots which select a lifecycle stage. For the selected stage, the tool displays the relevant artefact types, task types and job roles. The user then selected the Function Decomposition artefact type, which caused the associated Task Type and responsible Job Roles to be highlighted, as well as the details of the Artefact Type (including the graphical example) to be displayed on the right.

Not visible in the figure, below the properties of the Artefact Type, its relationships will be listed as hyperlinks. Users can follow these to drill down to other information, e.g. Techniques, Reference Models, Standards, Examples or previously completed instances of the Artefact Type. This navigation can be continued to an arbitrary depth. The icons above the right hand pane create a “bread crumb” trail recording the navigation path, allowing the user to quickly jump to any item which has been traversed. This is similar to a Miller style interface as used in the MacOS Finder [17].

The user can also select items in the Task Type or Job Role lists. The selected item will then appear in the right hand pane and associated items in the other facets (Artefact Types, Task Types, Job Roles) will be highlighted.

Above the right hand details, you will notice a “Doc” button. This invokes the Document algorithm previously discussed which will generate a hyperlinked report for the selected item and all directly related items. In this way we can immediately get a view of the method relevant to a Project Type, Lifecycle Stage, Artefact Type (at any level of summary), Task Type (ditto) and Job Role (ditto). This allows creating a relevant subset which is tailored specifically to purpose. This maintains consistency across Project Types while reducing the “method burden” that specific projects or practitioners need to deal with.

Finally, for authorised method engineers, you will notice there is an “Edit” button on the top right too. This allows immediate editing of method definitions and relationships, which will instantly reflect in all the other views, allowing rapid method evolution and (where desired) instant deployment. EVA versioning facilities can be used to preserve history and create versions that are not yet live/default.

7. Results and Discussion

7.1 Advantages of a Goal-Based approach include:

- Consensus between management and technical staff is easier to reach, since goals are shared, while opinions on how to achieve them are much more diverse

- The method definition is more stable since the agreed goals change more slowly than techniques, approaches and styles e.g. We know that we need a competent data model but that could be achieved through entity modelling, object modelling or semantic modelling
- Method evolution is supported as new approaches and techniques emerge, without having to redefine the structure and reach consensus again
- The reason why we do things is much more apparent to developers and technical staff, so artefacts required are not perceived as administrative burden or "make work" imposed by management
- Goals serve as a taxonomy for identifying potentially redundant activity or deliverables, thereby reducing complexity and effort
- Achievement levels are easily reviewed and visualised. Underperforming areas are easily identified and targeted for improvement. Periodic surveys can track continual improvement

7.2 Platform

Having a competent platform to capture, model, analyse the methods and refine or forward engineer the methods provided major benefits, including:

- Ability to capture the full scope of existing method definitions in a structured way
- Ability to assess the scope, coverage, consistency and overlaps within and between methods
- Ability to add goals and assess contribution of various artefact types and associated task types and resources to goal achievement. This also allowed identification of artefacts that addressed similar goals and were therefor alternates or redundant
- Ability to identify common elements across different styles of projects (e.g. Agile vs Waterfall; or Custom Development vs Package Implementation). This allowed reducing the size of the overall methods and achieving much higher consistency, in turn leading to easier quality assurance and building of skills
- Modeling the methods and being able to navigate from perspectives such as Concern, Lifecycle, State and Domain greatly facilitated understanding of maturity, gaps and redundancies and related discussions with stakeholders
- The "one page portal" provides a "single source of truth" for practitioners in an easily accessible form and allows extracting project or role specific method fragments dynamically. Following the principles of emergent behaviour from group efforts[14], where individuals need only follow their local rules, this can greatly reduce the cognitive load that individual practitioners need to deal with to only that relevant for their role within a given project
- Method evolution is greatly facilitated by the repository, since the underlying definitions can be changed automatically flowing into portal, reports, visualisations and other formats

7.3 Adoption

Adoption in the client organisation has been limited. This was due to a number of factors, including two key players retirement towards the end of the project (the main sponsor and the senior manager responsible for the methods centre of excellence). The latter was temporarily replaced with a contract resource but this party did not have seniority, continuity or gravitas to drive the initiative fully. The sponsor was recently succeeded by a competent and sympathetic party, but this individual is playing a larger role and is new to the organisation, so has a lot to get to grips with. The initiative has also competed with the better funded efforts of a strategic partner responsible for contract and offshore resources for a major programme, and a preferred method which they promote.

Those parties and projects who engaged with the work, the method and the portal have been positive. We remain hopeful that we will get more traction, but the difficulties again illustrate that methods adoption is a social process and will need management motivation, backing and support for training and change management to be successful.

8. Future Work

The deployment to date has not yet exploited the capabilities of the platform and the meta model realised in it to manage the artefact and project instances as well as the method definition. This was due to existing investment

in a Sharepoint portal and associated content and training in the organisation and a lack of political will to change this situation. It was also an early adoption stage for the tool platform and trust must still be established with respect to the scalability and security of a SaaS platform in general and the EVA platform in particular. We hope to expand this capability and its deployment in future.

The author is also engaged in extensive research in the development of effective graphical modelling notations for enterprise and systems modelling. These activities are supported to a reasonable extent in the same tooling. A new generation of tooling now being contemplated and prototyped should allow us to more effectively integrate method engineering with graphical language design.

References

1. Humphrey, W. S. (March 1988). "Characterizing the software process: a maturity framework". IEEE Software. 5 (2): 73–79. doi:10.1109/52.2014. ISSN 0740-7459.
2. Paulk, Mark C.; Weber, Charles V; Curtis, Bill; Chrissis, Mary Beth (February 1993). "Capability Maturity Model for Software (Version 1.1)" (PDF). Technical Report. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. CMU/SEI-93-TR-024 ESC-TR-93-177.
3. Olle T W, Hagelstein J, Macdonald I G, Rolland C, Sol H G, Van Assche F J M, Verrijn-Stuart AA, (1988) Information System Methodologies - A Framework for Unerstanding. Addison-Wesley, 1988.
4. Avison D E and Wood-Harper A T (1990) Multiview: An Exploration in Information Systems Development. Blackwell Scientific Publishers, Oxford 1990.
5. Avison D E and Wood-Harper A T (1991) Information Systems Development Research: An Exploration of Ideas in Practice The Computer Journal Vol 34 No 2 1991 pp 98-112.
6. Sjaak Brinkkemper, Method engineering: engineering of information systems development methods and tools. Journal of Information & Software Technology, Vol 38, n°4, pp 275-280 (1996)
7. C. Rolland. A Primer for Method Engineering. Proceedings of the INFORSID Conference (INFormatique des ORganisations et Systemes d'Information et de Decision), Toulouse, France, June 10–13, 1997.
8. Kent Beck; James Grenning; Robert C. Martin; Mike Beedle; Jim Highsmith; Steve Mellor; Arie van Bennekum; Andrew Hunt; Ken Schwaber; Alistair Cockburn; Ron Jeffries; Jeff Sutherland; Ward Cunningham; Jon Kern; Dave Thomas; Martin Fowler; Brian Marick (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.
9. Fowler, Martin. "Using an Agile Software Process with Offshore Development". Martinfowler.com. Retrieved 6 June 2010.
10. Sutherland, Jeff; Brown, Alex. "Scrum At Scale: Part 1". Retrieved 14 September 2015.
11. McLeod, G. (1993) “A Generic Model for Method Representation, Integration and Management”. Proceedings, First European Conference on Information Systems, Henley, UK. 1993
12. McLeod, G (1992) “Creative Methods Management” (1992). South African Computer Journal. 1992
13. McLeod, G (1995) “MetaTool: A software tool in support of methods management”. Proceedings 1st International Meta-CASE Conference, Sunderland, UK, 1995
14. McLeod, G (1999) “Exploiting Emergent Behaviour and Rule Association to Make Methods Simpler”. Evaluation of Modeling Methods for Systems Analysis and Design (EMMSAD)’99, Heidelberg, Germany, 1999
15. McLeod, G (2001) “PAMELA: A Proto-pattern for Rapidly Delivered, Runtime Extensible Systems”. Evaluation of Modeling Methods for Systems Analysis and Design (EMMSAD)’01. Interlaken, Switzerland, 2001.
16. Inspired.org (2018) EVA Netmodeler Brochure. http://www.inspired.org/s/EVA-Brochure-2018Q2.pdf. Accessed Mar 2019.
17. Tidwell, Jenifer. Designing Interfaces. O'Reilly. Retrieved 5 May 2015.