DE GRUYTER OPEN

Applied Computer Systems doi: 10.1515/acss-2016-0001

2016/19

Softwareland Chronicles: A Software Development Meta-Process Proposal

Sandro Bolanos[1], Rubén González Crespo[2], Jordán Pascual Espada[3], Vicente García-Díaz[4], Janis Osis[5]

[1] *Distrital University F.J.C., Colombia,* [2] *Universidad Internacional de La Rioja, Spain,*
[3, 4] *University of Oviedo, Spain,* [5] *Riga Technical University, Latvia*

***Abstract* – This paper presents the software development meta-process (SD-MP) as a proposal to set up software projects. Within this proposal we offer conceptual elements that help solve the war of methodologies and processes in favor of an integrating viewpoint, where the main flaws associated with conventional and agile approaches are removed. Our newly developed software platform to support the meta-process is also presented together with three case studies involving projects currently in progress, where the framework proposed in SD-MP has been applied.**

***Keywords* – Software development meta-process, software process-and-methodology modeling language.**

I. Introduction

Methodology-and-software development processes have been proposed for a long time. The list of such processes is rather long; however, still more proposals appear with the aim of offering the best strategies when facing a software project [1]. In this paper, we first state the problem as: *"softwareland chronicles"* and *"the emperors' new clothes"*. Subsequently, we put the solution into context by setting up three theses that lead to the software development meta-process proposal, its conceptual framework, and its modeling language. Finally, we present the results of three projects currently in progress where the metaprocess (SD-MP) is being implemented. Some future research is suggested together with main conclusions from the present research.

II. Problem Statement

A. Softwareland Chronicles

Once upon a time, in the realm of intellect, where the binary language prevailed, there was a primary species called programs. This species lived according to simple algorithms, undertaking basic tasks like: displaying messages like "Hello World", doing sums, jumping, among other simple activities. After a K-order time (Kilo-time) interval, came to the realm a new order called Software Engineering [2]. *This order came into the realm to install and develop new ways of thinking and organising ideas.*

The following narration is not intended to be a detailed description of the story of softwareland, but it does mention the most relevant events of such a story.

1. The Lineage of Processes

As the software order emerged, also did the Development Process lineage, and so the first trend was recognised, namely Code and Fix [3]. This trend promised programs would be better; it was a simple order [4], and apparently an advantageous order, but there was a lot of failure and so this emerging society seemed to have a bleak future. Afterwards the great Waterfall came [5], which would establish a formal body of activities that, in time, would become disciplines. The great hope was the development of quality programs (i.e. software development).

Most of the teaching heritage from Waterfall would focus on documenting the story how software was being created, since, apart from other benefits, doing so favored software evolution [5]. The chronicle writer Standish Group [6] would report a great deal of victories associated with this trend, but also many defeats connected to the way software was conceived. It was not long before a phenomenon called "Software Crisis" appeared [7]; this phenomenon would cause the successive defeats of the softwareland order. Thus, other trends appeared, such as the well-known Spiral Process [8], whose objectives would focus on risk management as an inherent phenomenon in software projects. Another trend encouraged by the V Process [9] appeared; in this case the focus was to confront software development with testing in all stages, which would allow for a better conception of software. Of course other models can be listed, e.g. prototypes, incremental, evolutionary, etc.

The appearance of the great CMM [10] accompanied by its entourage, namely TSP and PSP, was undoubtedly a milestone. The prestigious model (CMM) would promise better quality in terms of maturity; continuous improvement was one of its slogans. CMM created a stratified system that would limit or permit the development of software.

The last remarkable process milestone might be represented by the courageous RUP [11], which came to rescue processes empowered by its UML language widely recognized ever since [12]. As dissatisfaction was growing, it seemed that the lineage of processes would continue forever, although problems did not cease to appear [13].

2. The Lineage of Agile Methodologies

In the time of processes, there was a dissident trend that did not accept the fact of having so much failure when conducting software projects; especially when such projects had trusted these processes to achieve quality. Thus, a synthetic manifesto

was put forward, where the followers of this dissident trend suggested why the solutions to software problems pertained to agile methodologies. The most widely recognised trend within this new order was eXtreme Programming, also known as XP [14]. Some of the best-known proposals were: planning games, small liberations, metaphor, simple design, and some other not mentioned in order to keep the story short. Other leading trends emerged, such as Scrum [15], with its popular development springs, which were founded on a backlog that allowed having visible software results in periods of time no longer than 30 days.

3. The Lineage of Open-Source

This trend focuses on visible software development for a community that is interested in working according to the bazaar metaphor [16]. This proposal is as important as the others and it gathers a great deal of population in softwareland.

4. The War between Processes and Methodologies

War was declared [17]. Battle lines were then drawn; hostility broke out among the different armies of the development community in softwareland [18]. XP, a visible leader of the methodologies lineage, is said to be harmful for the development of reliable software [19], and so its leadership has become questionable [20]; some opponents claim that "waterfall has not died" [21]. There is a court case against XP [22]; the manifesto is said to be cynical [23] and so a contra-manifesto has been issued.

Short afterwards there was a counter-attack; agile methodologies attacked processes by claiming that processes were fictional [24] and their only objective was to maintain their status quo by being normative in utopian development situations [25]; cheerful RUP was regarded as a dinosaur [26], and the lifecycle of classical development was attacked with claims like "lifecycle is harmful" [27].

B. The Emperor's New Clothes

"In those times, the clothes being made had the quality of becoming invisible for those who were not up to the tasks or for the foolish", Hans Christian Andersen [28].

The main strategy to approach a software project lies in adopting a software process or methodology. Out of the varied wardrobe of methodologies and processes, we are forced to choose some of these strategies to end up accepting that the chosen strategy is the one that best suits our needs, otherwise "we take the risk of being foolish or not being up to the task".

III. The Solution

Processes and methodologies are immersed in a discussion about which of the approaches is right; we may say that while processes have focused on solving the "know how" of the problem, methodologies have focused on the "know what", "know who", and "know when"; we will refer to this situation as H&W problems.

Each approach has put its opponents' viewpoint in the background. While processes highlight the activities and the way they articulate with one another [29], methodologies look at practices, values and principles [30]. Both methodology and processes, hereafter referred to as M⊕P, have attempted to offer clothes that overlay software with quality.

A. Theses

We consider the following theses:

- M⊕P is not part of the solution; it is part of the problem instead.
- M⊕P should be modelled using a language, "it should speak the same language" and must be observed from a superior category.
- M⊕P is a new species.

B. Flatland

M⊕P lives in Flatland [31]. Methodologies and processes are orthogonal theories. Conventional processes are typically associated to large-scale project development, whereas agile methodologies are associated to small projects [32]. However, there are some approaches [33] that resemble different mixtures.

Similar to Flatland (by Abbot [31]), here we find circles like CMMI, ISO, SPICE and other shapes with a normative character whose fundamental doctrine resides in the following motto: "listen to your configuration". Also remarkable within Flatland, we find polygons like Rup, WaterFall, XP, and Scrum among others; even some Open-Source irregular shapes.

C. Metaization

The principle of Metaization [34] is used in different areas; for example, the famous Hilbert's program [35]. This system was proposed to be called meta-mathematics.

In order to introduce this concept into our discussion, we need to recall Flatland, where it was established that any proposal, either from methodologies or from processes, constitutes a limited solution. This is because the solution to the H&W problem that this kind of proposals deals with is unable to cope with the questionings about M⊕P, in other words, the clothes produced do not represent the solution on the whole, as stated in Thesis 1. The more aware we become (within our Software Engineering discipline) that studying M⊕P is advantageous to software development, the higher the quality of software will be; that is, studying M⊕P to conduct a particular software project is as necessary as using requirements engineering, architecture, design, implementation or testing. Before considering these aspects in detail, let us recall Flatland once again. The proposal is then to add up one more dimension; this dimension permits describing M⊕P since it creates a higher level, which is achieved by applying metaization.

From this extra-dimension, hereafter referred to as meta!, it is possible to see flat shapes that, just like Flatland, will represent either methodologies or processes. It is clear that from the meta! dimension it is possible to understand, and of course to question M⊕P, even to propose something. What we have is a wide variety of methodology proposals and software processes that have been very useful, but they cannot continue to appear out of spontaneous generation [36] *"it is*

unknown as how they came into existence, except for the people who create them". We cannot rely on this with no questioning about how suitable these principles are for a particular project [37].

D. Meta-Process

A meta-process has been observed from the process evolution perspective. In this approach, there is distinction between the real-world process, which gathers the necessary activities to make a software product by including people, and the process model, which reflects the real-world process [38]. Evolution addresses the establishment of the steps that affect the process in the real world as well as the process model. Moreover, evolution keeps consistency in the possible changes. Models, such as CMMI [39] and SPICE [40], are regarded as meta-processes; although these models focus on real-world process evolution rather than on process model evolution, whose subject of study lies in the process cycle, and/or on the methodologies.

We can also resort to the concept of prescriptive and descriptive model [41]. As long as we establish the necessary steps for the real-world process to evolve, we may say that we are configuring a prescriptive model, in other words, proposals like CMMI are prescriptive models of the meta-process, hence, its normative character. On the other hand, if we establish a model of the steps that were taken in the real world, we are referring to a descriptive model [42].

The meta-process must be an umbrella concept that defines the reasons (why) for the "know how", "know where", "know who", "know when", in other words, it defines cause-effect [41]. This type of holistic question hereafter will be referred to as W(H&W), and it arises as a fundamental trait to build meta-process instances [43].

E. Software Development Meta-process

From the meta-process viewpoint, conceptual tool must be provided for the genesis of M⊕P, that is, W(H&W) must be answered; this will constitute the support for Thesis 1. To establish the software development meta-process, hereafter SD-MP, we propose a framework founded on three questions as follows:

a Why is it possible to select a methodology or a software process?
b Why is it necessary to build a methodology or a software process?
c Why is it necessary to create new ways of development within and beyond the concepts of methodology or software process?

The answers to these questions configure SD-MP as follows:

a Because there are M⊕P that can be adjustable to a particular project, the solution would correspond to a management process.
b Because there is no M⊕P that adjusts to the software product; the solution would correspond to a structuring process.
c Because, within M⊕P there is no concept that adjusts to the singularity of the software project, the solution corresponds to an innovation process.

1. Management

This idea is based on selecting and/or adapting the appropriate M⊕P for a particular software project [44]. In order to select or adapt M⊕P, it is necessary to lean on strategy definition, organisation, production, and documentation processes. Strategy definition consists in determining whether a methodology or a process is to be adopted [45], or else if a combination of the two approaches is adopted. As a consequence of the management process, the M⊕P-manager role is established. The knowledge of this role focuses on the acknowledgement and execution of an adjustable M⊕P (see Table I).

2. Structuring

This idea is based on the suitable construction of M⊕P for a particular project. In order to build M⊕P, the following is necessary: defining the architecture that should be adopted according to the methodology or process. Additionally, M⊕P mechanism configuration and/or guideline configuration must be carried out, either from the perspective of the activity (process) [46] or from the perspective of the value (methodology) [47]. As a consequence of the structuring process, the M⊕P architect's role is established, whose knowledge addresses the assembly and construction of an M⊕P that suits the project (see Table I).

3. Innovation

This idea is based on the creation of new ways of development within and beyond M⊕P. The creation of M⊕P requires establishing communication mechanisms [48] within M⊕P and promoting knowledge management [49] in its two dimensions, namely tacit and explicit knowledge [50]. As a consequence of innovation, the M⊕P innovator's role is established; whose knowledge addresses the creation of new ways of M⊕P (see Table I).

TABLE I
PROCESSES AND ROLES

Management "Selection"	Structuring "Construction"	Innovation "Creation"
Process: - strategy - organization - production - documentation	Process: - architecture - mechanisms - artefacts map - contribution	Process: - communication - knowledge - problem - improvement
Role: M⊕P manager	Role: M⊕P architect	Role: M⊕P innover

F. The Meta-Process as a Layered Architecture

The meta-process is configured in a layered architecture [51] and its position is on the top layer. The software development meta-process constitutes the ontological instance [52] of the meta-process. Methodologies and processes are

located at intermediate layers, and methodologies such as XP, Scrum, etc. constitute the ontological instance of a methodology. Processes like Waterfall, RUP, etc. constitute the ontological instance of a process. Methodologies and processes are linguistic instances [52] of the meta-process; likewise, XP, Scrum, Rup, Waterfall, etc. represent linguistic instances of SD-MP. Customising M⊕P, creating new M⊕P concepts, and going beyond M⊕P represent ontological instances of M⊕P and linguistic instances of SD-MP. In these instances, we find most of the software development research from the perspective of M⊕P and beyond. This is not about reinventing the wheel [53] once and again, these ideas are about suggesting effective solutions over and beyond M⊕P. These ideas should constitute a silver bullet in software engineering.

G. Modelling Language for M⊕P from the Meta-Process

Process Modelling Languages (PMLs) [38] have been proposed by the process-study community, giving rise to a considerable number of them. These languages are based on activities, roles and artifacts; likewise, process-centered environments (PSEEs) [38] have also been developed. This phenomenon has not spread with the same intensity on methodologies. The problems associated with development still place a strong emphasis on solving the "know how".

We propose a modelling language for M⊕P from the perspective of software development meta-process, hereafter referred to as M⊕P Modelling Language (M⊕P-ML). This language will provide support to Thesis 2. The language is structured over an object-layer and a meta-layer; within the proposal, the language considers the PML approach, but additionally includes fundamental elements to model methodologies.

The M⊕P modelling language is intended to facilitate the use of, construction of, and creation on M⊕P. The vocabulary involved configures the most representative elements of M⊕P as well as the possibility of creating new concepts based on language categories and also on their meta-vocabulary, derived from the root that help maintain the structure and the philosophy of the language. This language is equipped with a type of graphical notation similar to UML [12] and also to BPMN. The specific notation we propose might be seen in the Coloso [54] environment. It is out of the scope of the present paper to go into details of the language and its notation.

H. M⊕P-ML as a Layered Architecture

Regarding PMLs, M⊕P-ML is placed on an upper layer since, from its target layer, M⊕P-ML permits specifying a PML. M⊕P-ML proposes its own object language (M⊕P-OML) and extends it to a PML, since M⊕P-ML suggests modelling not only processes but also methodologies, that is, modelling both the "know how" as well as the "know who", "where", "when", etc.

PMLs such as SPEM are particular instances of the language for M⊕P on its meta-layer (M⊕P-MM). The fundamental contribution of M⊕P-ML lies in estimating concepts, not only with regard to the processes but also to the methodologies, which permits a more complete type of modelling. Both process and methodology are considered to be types of strategy.

I. M⊕P Is a New Species

Coexistence between processes and methodologies is not an easy matter. The emphasis associated with one or another strategy makes them eclipse one another. However, it is important to know when to switch strategies, that is, when considering the procedure to solve the problem, it is essential that the "know how" stands out; conversely, whenever it is required to know "who", "where" or "when", methodology should stand out. With the software development meta-process and the proposed language, the idea is to exploit the advantages of processes as well as of methodologies, and so facilitate their modeling, their use and their evolution. We consider that if it involves human aspects and automatic aspects, M⊕P might be considered new species of processware, supporting Thesis 3.

IV. Coloso Platform

The SD-MP Coloso platform is the type of software developed to support the SD-MP meta-process. This software can be downloaded from www.colosoft.com.co. Here, the meta-process framework is offered together with its viewpoints, which are implemented to support M⊕P management, M⊕P structuring, and M⊕P innovation.

V. Result

The following section presents the results of three development projects currently in progress. These projects implement the meta-process. The projects have been proposed under the following conditions:

a Project A proposes an information system about Colombian biodiversity. Project B proposes the creation of an information system for the Health Department of the Capital District in Bogota, Colombia, addressing the problem of animal control. Project C proposes the development of an entrepreneurial portal.

b All projects are subject to an 8-month schedule and a 6-member development team.

c The development platforms can be freely chosen.

It is worth mentioning that the meta-process engineer working on the three projects is the same person; the idea is to simultaneously evaluate SD-MP execution and measure the different levels of management, construction and creation for M⊕P. After three months, projects are underway and we have gathered the following evidence:

A. Management Process

TABLE II
Strategies

	Project A	Project B	Project C
SD-MP	RUP	OpenUP	Scrum
Strategy	process	Proces/methodology	methodology

1. Strategy Definition

In the case of project A, the RUP process was chosen. In project B the choice was OpenUP, and project C used Scrum (see Table II).

Findings associated with this activity: leaders were clear about the need to use a strategy, but they were unclear about the process and methodology trends. Project A chose RUP since it was demanded by the organisation; Project B chose OpenUp due to the team's knowledge on projects running over Eclipse platform; while Project C chose Scrum, due to the team's intention to experiment with agile methodologies.

2. Organisation

The SD-MP meta-process proposes a division of roles into two categories, namely solution-domain roles and problem-domain roles. In the case of the meta-process, the bedrock is human resources; a clear definition of the participants and their functions constitutes a fundamental trait that must be established at the beginning of a project.

Defined roles were influenced by the character of the process or methodology, as shown in Table III.

TABLE III
ROLES

SD-MP	Project A RUP	Project B OpenUP	Project C Scrum
solution domain roles	- analyst - architect - implementer - tester - project manager	- architect - project - manager - analyst - tester - developer	- scrum master - development - team manager
problema domain roles	- stakeholder	- stakeholder	- stakeholder - product owner

The meta-process engineer insisted on the importance of defining problem-domain roles and solution-domain roles, arguing that, regardless of the M⊕P chosen, there must be responsibilities for the two fundamental actors, that is, who are to have the problem and who are to offer the solution.

3. Production Planning

The SD-MP meta-process considers production as the role-product association. The main purpose is to identify and control M⊕P through products generated by each role. This viewpoint can be interpreted from a higher level and also at a detailed level. In this particular case, the higher level perspective is estimated, that is, the macro-products that configure the links leading to the final product. Each M⊕P can also be interpreted from the detailed production viewpoint, where the most specific products are linked to the roles that created them.

There was consensus about moving forward with the functionalities of the entrepreneurial portal on a weekly basis, milestones were established and it was also agreed to move forward with milestones as goals were attained. This activity involved the participation of the meta-process engineer, the scrum master and the product owner (Table IV).

TABLE IV
PRODUCTS

SD-MP	Project A RUP	Project B OpenUP	Project C Scrum
subproducts	- architecture	- architecture	- spring backlog
final product	- information system	- information system	- enterprise portal

Findings associated with this activity: the roles involved were clear about the final product. Once requirements were established, while looking for the links leading to the final product, the meta-process engineer highlighted the importance of defining higher-level sub-products to help visualise an anticipated configuration of the final product.

4. Document Planning

In SD-MP, it is possible to support the lifecycle of a software project through documentation. There are two types of documents: the fundamental document, which consists of all source codes and data that run directly on machines; and support documents, which serve to extend the semantics of fundamental documents. The double nature and importance of documentation warns about the relevance of observing the two branches of the same project that should be kept consistent throughout the project's lifecycle (see Table IV).

TABLE IV
DOCUMENTS

SD-MP	Project A RUP	Project B OpenUP	Project C Scrum
fundamental documents	- codes	- codes	- codes
supporting documents	- use cases - uml diarams - E-R models - manuals - others	- use cases - uml diarams - E-R models - manuals	- user stories - uml diarams - product backlog

Findings associated with this activity: managers were clear about the importance of documents. The meta-process engineer stressed the concept of documentation for M⊕P as a mechanism to support and contribute to the evolution of M⊕P. In the case of RUP and OpenUP, it was clear and straightforward that UML should be used due to the background links between the two proposals, namely "language-process". In the case of Scrum, the integration of UML was not as straightforward, but it was proposed as an essential element of design.

B. Structuring Process

1. Architecture Realization

The SD-MP meta-process identified that the M⊕P architecture involved two concepts, namely the construction and/or adaptation of the conceptual blocks that conform M⊕P. This is performed from two perspectives, namely from the model that establishes M⊕P and from its corresponding

execution. The first architectural source lies in the diagram or scheme used by each M⊕P to show the articulation between phases, activities, tasks, instructions and practices.

For the purposes of our studies this architecture includes only the development disciplines; although the support disciplines are carried out, they are not included in our research schedule. There is particular emphasis on the relevance of understanding the philosophy of the process, its iterations, and the impact that each phase has on the different disciplines after each iteration. More specific details that pertain to the process begin to appear. In this case, it was not necessary to build the model since the model itself was proposed by RUP. However, it was essential to adapt the model specifically to the support disciplines. The recommendations of the meta-process emphasise the importance of being coherent regarding the model and its execution, recording the adaptations made throughout the process on the model's architecture.

The architecture proposed in the OpenUP defines the following: increments on a daily basis, weekly iterations and the life cycle of the project. The assembly between process and methodology is highlighted due to the influence of RUP and the agile methodologies (Scrum and XP) on OpenUP. In this case, it was not necessary to build the model since the model itself had already been proposed by OpenUP. The recommendations of SD-MP were also emphasised regarding the importance of being coherent between the model and its execution and also the importance of recording the adaptations made throughout the process on the model's architecture.

The architecture proposed for the Scrum project defines the way Sprints must be carried out. In this particular case, it was not necessary to build the model since it had already been proposed by Scrum. The recommendations of SD-MP were also emphasised regarding the importance of preserving coherence between the model and its execution. Unlike RUP and OpenUP, Sprint's architecture is far simpler to follow and does not require any major adaptations.

Findings associated with this activity: Defining the architecture of the different projects represented significant progress and also helped to clarify the spirit of each M⊕P. In the case of RUP, the architecture strongly suggested the need to have a more specialised team capable of dealing with the process model in its true dimension. In the case of OpenUP, the situation was more manageable. The feeling within the OpenUP team reflected that team members were able to cope with their own M⊕P. Scrum team members reflected a light, understandable, and easy-to-follow structure.

2. Configuration of Mechanisms and Guidelines

In the SD-MP meta-process, configuring the mechanisms and guidelines allows detailing the building blocks that constitute M⊕P. For processes, the following can be distinguished: phases, activities, tasks and instructions. For methodologies, the following can be distinguished: practices, values, principles, risks, restrictions, limitations, and methodology's own methods among others. The value of identifying the different shades of M⊕P from its configuration elements lies in the precise definition of the project-driving philosophy.

In the case of RUP, phases, activities, and tasks are distinguished. From the perspective of SD-MP, these characteristics constitute a process model with a high degree of complexity, since it is necessary to understand the way in which these mechanisms articulate with one another. Some of the major difficulties of RUP are the comprehension of the disciplines involved and the simultaneous execution of the phases; moreover, it is difficult to involve iterations. It is clear that the use of more mechanisms implies more elements. We stress the importance of having a clear purpose for each mechanism so that the process reinforces the development and does not become an obstacle when trying to find fluency and go forward with the project. In the case of OpenUP, the following are distinguished: phases, activities and tasks. These aspects are affected by risk management and value management; here we assumed daily increments. Both mechanisms and guidelines were perceived. Additionally, a fusion between process and methodology was made clear. The complexity of the process is alleviated by using methods such as the personal approach to daily work. From the perspective of SD-MP, emphasis is placed on how important it is to acknowledge the value of maintaining the same mechanisms and guidelines within the same model. In the case of Scrum, tasks are distinguished as a fundamental mechanism. Additionally, there was no doubt about the use of a set of guidelines that were represented by practice evidence such as: early releases, daily meetings, tracking of a sprint whose input was the product backlog, among other methods configuring the nature of the methodology. From the perspective of SD-MP, the following aspects can be highlighted: the value of applying these practices, the need to control development, and the fact that, despite facing the possibility of constantly updating user's requirements, this represents no obstacle to the progress of the project (see Table VI).

TABLE VI
MECHANISMS AND GUIDELINES

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
mechanisms	- phase, activity task, iteration discipline	- phase, activity task	- task
guidelines		- risks manage - increments	- early release - meeting - sprint

Findings associated with this activity: defining the mechanisms and guidelines for each of the M⊕P permits estimating the degree of complexity of processes and methodologies as well as the degree of commitment that should be taken when facing each concept, leading to an appropriate understanding of the process or methodology.

3. Artifact-Map Making

In the SD-MP meta-process, establishing the artifact map means creating a linking thread for M⊕P through the

artifacts, representing visible milestones. The conceptual continuity achieved by using artifact traceability eases the execution and evolution of M⊕P.

These artifacts were easily identifiable (see Table VII).

TABLE VII

ARTIFACTS

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
artefacts	- uml diagrams	- uml diagrams - fortnightly release	- task - weekly release

Findings associated with this activity: defining the artifacts as the route map allowed identifying the traits of each M⊕P. In the case of RUP and OpenUP, the prescriptive character of processes was noticeable, yet this was more evident in RUP. In the Scrum project, the adaptive character was more evident; since by the third month, the path that the proposed releases should follow had changed at least twice. There was a higher degree of customer satisfaction in the Scrum project due to the amount of progress materialized in the product; however, RUP and OpenUP appeared to have been more robust due to their evident levels of planning.

4. Contribution

The SD-MP meta-process suggests representing the contribution as the input−output relation between strategies, mechanism, and guidelines; which permits valuing the importance of each functional element of M⊕P.

The contribution of the process, as well as of the activities and tasks, was easily identifiable in the RUP project. Such a contribution is fundamentally represented in the documentation. This can be seen in the product versions obtained so far (see Table VIII.)

TABLE VIII

INPUT-(STRATEGY/MECHANISM/GUIDELINES)-OUTPUT

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
input	- use cases	- use cases	- product backlog
strategies	- process	- process - methodology	- methodology
mechanisms	- activities - tasks	- activities - tasks	- task
guidelines		- early release	- early release
output	- documentation	- early version	- early version

Findings associated with this activity: each M⊕P in each project justifies its conceptual framework with its contribution; however, the RUP project is the one whose evidence is more noticeable in terms of project execution, that is, the input-output relation for this project is enriched with more artifacts. However, the project that instills more confidence in the customers is Scrum due to its direct requirement-product relation. In the OpenUP project, there is a balanced relation between documentation and software.

C. Innovation Process

1. Communication

Communication is the fundamental pillar of innovation. SD-MP states the need to establish various communication methods that ease knowledge socialisation and exchange, making it flow throughout M⊕P.

The following methods were proposed for communication in the RUP project: interviews with users at the beginning of the project in order to establish the requirements; meetings with the leader and the project team on a fortnightly basis. In the case of the OpenUP project, communication methods were as follows: interviews with users every week, and meetings with the project manager and the development team. The communication methods for the Scrum project were the following: interviews with the stakeholders at least every two days, a weekly meeting with the product owner, daily meetings with the developers, and weekly meetings involving the Scrum master, the manager and the development team (see Table IX).

TABLE IX

COMMUNICATION

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
comunication	- interviews	- regular interviews	- intensive interviews
	- meetings	- regular meetings	- intensive meetings

Findings associated with this activity: making communication methods explicit favours interaction within projects. In the Scrum project, the strong tendency towards using different interaction and communication mechanisms is evident, which strengthens the bonds between team members. In fact, a higher degree of collaboration was easily observable in the Scrum project, which could be explained by the dynamic nature that pertained to this agile methodology. In the cases of RUP and OpenUP, the most interaction-engaging and bond-strengthening activities were requirement elicitation (in the second week of the first month) and architectural design (in the second month). Subsequently, each of the teams focused on their own activities; OpenUP, however, switched a bit more between activities like interviews and meetings.

2. Problem Identification

In the SD-MP meta-process, the determination of the problems arising when modeling and executing M⊕P is a key aspect to the improvement of the M⊕P per se. Problem identification draws its attention to determining possible dissatisfaction when using M⊕P and also attempts to overcome these inconveniences by adjusting to the execution expected by the M⊕P, or else adjusting the framework of M⊕P.

It was easily observed that the RUP project framework could not be carried out as rigorously as proposed due to the limitations imposed mainly by the participant roles. Difficulty in adopting and following the RUP conceptual framework was also observed, including the fact that the process was imposed

by the organisation. In the OpenUP project, the greatest difficulty lies in understanding the coexistence between process and methodology as well as the implications of the different strategies. In the Scrum project, one of the greatest difficulties arises when integrating UML with a methodology that does not incorporate UML explicitly. Putting together diagrams such as use cases and the user-stories method was first seen as cumbersome [56], instead of thinking about this situation as complementary (see Table X).

Findings associated with this activity: when starting each project, each of the teams became familiar with their corresponding process or methodology.

TABLE X

PROBLEM IDENTIFICATION

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
comunication	- role and responsibility adjustments	- release underestimation	- lack of control in requirements

Although leaders had some background on the chosen M⊕P, the sensitising process caused some complications. When carrying out the three projects, it was necessary to emphasise the use and tracking of the framework for each M⊕P. In the RUP project, it was necessary to make some adjustments that overloaded the functions of some of the roles. In the case of OpenUP, some releases were underestimated and so had to be reprogrammed, which took more time than expected. In the case of Scrum, it was necessary to issue a warning about user control in order to avoid underestimating or overestimating particular requests.

3. Improvement

Once the problems associated with an M⊕P have been determined, together with their corresponding recommendations, the SD-MP meta-process goes beyond by proposing improvements. Such improvements consist in revising M⊕P in order to identify new ways to strengthen it. This activity requires a previous assessment based on the evidence provided in the short, mid, and long run.

The main evaluation criteria proposed for the three projects are the supervision of critical factors for the execution of the process or methodology together with the mechanisms that facilitate the strategy architecture update. Improvement may take place in three phases according to the execution of M⊕P, namely pre-, in- or post-M⊕P. The pre-M⊕P improvement represents the type of M⊕P reinforcement that is based on evidence and knowledge. This improvement can be directly extracted from the framework proposed by the M⊕P. The in-M⊕P improvement is the type of reinforcement of M⊕P that is mainly based on the evidence obtained throughout the M⊕P execution process. Finally, the post-M⊕P improvement corresponds to the M⊕P reinforcement that is based on the results from the full execution of M⊕P lifecycle.

In the RUP project, the recommendation is to improve communication channels. In the OpenUP project, it has been suggested that the proposed architecture should not be neglected in an attempt to have early releases. In the Scrum project it has been recommended that more UML artifacts be included to strengthen documentation (see Table XI).

TABLE XI

IMPROVEMENT

SD-MP	Project A RUP	Project B OpenUP	Project c Scrum
improve	- reinforcement of communication channels	- follow the indicated architecture	- strengthening documentation

Findings associated with this activity: the improvement perspective has allowed for proper evaluation and reinforcement of each M⊕P, aiming for evolution. In-M⊕P improvement is currently taking place and, at the end of each project, post-M⊕P improvement is proposed.

4. Knowledge Management

In the SD-MP meta-process, knowledge management is one of the most important trends to be adopted in terms of software engineering; considering that software is knowledge [55]. These processes determine the possible changes that take place between tacit and explicit knowledge.

Each team has been sensitised to the knowledge sharing process within each project and such a process has been reinforced through the aforementioned communication methods. Emphasis has been placed on the relevance of personal work together with permanent interaction to strengthen cooperative bonds and experience exchange.

Findings associated with this activity: From the meetings held by the teams involved in the three projects it has been observed that there is an intention to share experiences supported by the knowledge of the job carried out. The concept of knowledge management and its corresponding importance is still rudimentary within the teams, but there is a growing interest in using the strategies proposed.

D. Observations on the SD-MP Meta-Process

By gathering all concepts involved in the SD-MP metaprocess, namely management, structuring and innovation, and also taking the three projects as reference samples, we conclude that, for the current experiment, the RUP project has a weight of 45 %, the OpenUP project has a weight of 33 %, and the Scrum project has a weight of 22 %. In addition, in the three projects, the management process was denser weighting 60 %. The structuring process weighted 30 % and innovation occurred with a weight of 10 %.

This statistical trend occurs due to the M⊕Ps used. Each of them configures a well-defined framework, which is fundamentally tracked within each of the projects. The degree of construction within the structure corresponds to some adaptations; such adaptations were mainly necessary in the RUP project.

Although the projects have only been assessed up to a third of their complete time span, there are significant findings regarding the framework proposed by the SD-MP meta-process. Additionally, there has been a positive impact on the execution of M⊕P due to the recommendations proposed by SD-MP as each project is running. The significant trait that is

proposed by the SD-MP meta-process is aimed at the configuration of framework concepts that allow for progressive evaluation of the M⊕Ps without interrupting their execution.

VI. Future Research

Our goal is to carry on revising M⊕P proposals, applied to processes currently in progress from the perspective of the SDMP meta-process where a greater impact on structure and innovation is required, thus achieving a more significant sample to obtain more conclusive results that permit providing feedback on the meta-process proposal presented.

For the present proposal, we have developed a tool called Coloso, which integrates languages such as Archimate and UML on the basis of its formulization [57]. These integrated languages are handled from M⊕P-ML, which allows modeling meta-process ideas. The idea is to integrate so that it is possible to combine the proposals and further strengthen the software development methodology and process.

VII. Conclusion

M⊕P is part of a problem that needs to be solved and must not be considered simply as an ad-hoc solution. This is what gives birth to M⊕P roles like manager, architect and innovator, whose functions are associated with management, structuring or innovation, respectively. These concepts configure the meta-process proposed in this study.

The models approach represents a powerful paradigm to software development. Proposals such as MDD permit seeing the impact of modelling. When modelling is implemented together with a language, it becomes more powerful. The software development meta-process proposal and the M⊕P modeling language address these two approaches.

The novelty value in M⊕P, from the perspective of the meta-process, is the strategic fusion of the two paths, namely methodology/process, where the advantages of both are exploited; the idea is to conveniently carry out either the methodology or the process as needed.

The SD-MP meta-process proposed suggests a light conceptual framework from which M⊕P is conceived as the object of study. This facilitates not only the revision of a particular M⊕P, but also suggests a framework for the construction of M⊕P that allows it to be adjusted to a particular project.

References

[1] J. Osis and E. Asnina. "Is Modeling a Treatment for the Weakness of Software Engineering?" in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, New York, USA, pp. 1–14, 2011. http://dx.doi.org/10.4018/978-1-61692-874-2.ch001

[2] P. Naur and B. Randell, *Software engineering: Report of a conference sponsored by the NATO science committee, Scientific Affairs Division, NATO*, Garmisch, Germany, November, 1968.

[3] S. McConnell, *Rapid Development*, Microsoft Press, 1996.

[4] J. Maeda, *Las leyes de la simplicidad*, Gedisa, 2008.

[5] R. W. Winston, *Managing the development of large software systems*: *IEEE WESCON*, July 22–26, 2002, Los Ángeles, USA, 1970.

[6] T. Clancy, "The Extreme Chaos," Standish Group International Inc., 2012.

[7] R. L. Glass, "The Standish Report: does it really describe a software crisis?" *Communications of the ACM*, vol. 49, no. 8 pp. 15–15, 2006. http://dx.doi.org/10.1145/1145287.1145301

[8] B. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988. http://dx.doi.org/10.1109/2.59

[9] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," *National Council On System Engineering*, vol. 1, no. 1, pp. 57–65, 1981. http://dx.doi.org/10.1002/j.2334-5837.1991.tb01484.x

[10] M. C. Paulk, C. V. Weber, B. Curtis and M. B. Chrissis, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley, 1994.

[11] P. B. Kruchten, *The Rational Unified Process (An Introduction)*, Addison Wesley, 1999.

[12] U. Donins, J. Osis, A. Slihte, E. Asnina and B. Gulbis. *Towards the Refinement of Topological Class Diagram as a Platform Independent Model*: *ENASE Model-Driven Architecture and Modeling-Driven Software Development*, Portugal, pp. 79–88, 2011.

[13] J. Osis, E. Asnina and A. Grave, "Computation Independent Representation of the Problem Domain in MDA," *e-Informatica Software Engineering Journal*, vol. 2, no. 1, pp. 29–46, 2008.

[14] K. Schwaber, *Scrum Development Process: OOPSLA95 Business Object Design and Implementation Workshop*, New York, USA. Springer, vol. 6, no. 4, pp. 170–175, 1995.

[15] E. S. Raymond, *The Cathedral and the Bazaar*, O'Reilly, 1999

[16] R. L. Glass, "Agile versus traditional: Make love not war," *Cutter IT Journal*, vol. 14, no. 2, pp. 12–18, 2001.

[17] M. McCormick, "Technical opinion: Programming extremism," *Communications of the ACM*, vol. 44, no. 6, pp. 109–119, 2001. http://dx.doi.org/10.1145/376134.376181

[18] AVOCA-GmbH, "Extreme Programming", 2012. [Online]. Available: www.avocallc.com/downloads/

[19] H. Saiedian, "Panel: Extreme programming: helpful or harmful?" *Proceedings. International Conference on Software Engineering*, Washington, USA. IEEE Computer Society, p. 718, 2003. http://dx.doi.org/10.1109/icse.2003.1201258

[20] S. Chatterjee, "The waterfall that won't go away," *SIGSOFT Software Engineering Notes*, vol. 35, no. 1, pp. 9–10, 2010. http://dx.doi.org/10.1145/1668862.1668875

[21] M. Stephens, *The Case against Extreme Programming*, 2012. [Online]. Available: http://www. softwarereality.com/lifecycle/xp/case_against_xp.jsp

[22] S. R. Rakitin, "Manifesto elicits cynicism," *Computer*, vol. 34, pp. 4–7, 2001.

[23] J. Nandhakumar and D. E. Avison, "The fiction of methodological development: a field study of information systems development," *Information Technology & People*, vol. 12, pp. 176–191, 1999. http://dx.doi.org/10.1108/09593849910267224

[24] D. Truex, R. Baskerville and J. Travis, "Amethodical systems development: the deferred meaning of systems development methods, Accounting," *Management and Information Technologies*, vol. 10, pp. 53–79, 2000. http://dx.doi.org/10.1016/S0959-8022(99)00009-0

[25] W. Hesse, "Dinosaur meets archaeopteryx? or: Is there an alternative for rational unified process?" *Software and Systems Modeling*, vol. 2, pp. 240–247, 2003. http://dx.doi.org/10.1007/s10270-003-0033-y

[26] D. D. McCracken and M. A. Jackson, "Life cycle concept considered harmful," *SIGSOFT Software Engineering Notes*, vol. 7, no. 2, pp. 29–32, 1982. http://dx.doi.org/10.1145/1005937.1005943

[27] C. Andersen, *El traje Nuevo del emperador*, La galera, 2009.

[28] N. B. Ruparelia, "Software development lifecycle models," *SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8–13, 2010. http://dx.doi.org/10.1145/1764810.1764814

[29] R. Hoda, P. Kruchten, J. Noble and S. Marshall, "Agility in context," *ACM SIGPLAN Notices*, vol. 45, no. 10, pp. 74–88, 2010. http://dx.doi.org/10.1145/1932682.1869467

[30] E. A. Abbott, *Flatland: A romance of many dimensions*, John Wilson and Son, Cambridge, 1885.

[31] L. Wang, "Agility counts in developing small-size software," *IEEE Potentials*, vol. 26, no. 6, pp. 16–23, 2007. http://dx.doi.org/10.1109/MPOT.2007.906114

[32] K. Sureshchandra and J. Shrinivasavadhani, "Moving from waterfall to agile," in *Agile Conf.*, AGILE '08., Toronto, USA, IEEE, 2008, pp. 97–101. http://dx.doi.org/10.1109/agile.2008.49

[33] T. Khne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, pp. 369–385, 2006. http://dx.doi.org/10.1007/s10270-006-0017-9

[34] R. Zach, "Hilbert's Program Then and Now", 2015. [Online]. Available: http://arxiv. org/abs/math/0508572
[35] J. Strick, "Darwinism and the origin of life: The role of H. C. Bastian in the British spontaneous generation debates," *Journal of the History of Biolog*, vol. 32, no. 1, pp. 1868–1873, 1999. http://dx.doi.org/10.1023/A:1004460408116
[36] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Information and Software Technology*, vol. 54, no. 5, pp. 433–447, 2012. http://dx.doi.org/10.1016/j.infsof.2011.12.003
[37] J. C. Derniame, B. A. Kaba and D. Wastell, *Software process: principles, methodology, and technology*, Springer-Verlag, 1999.
[38] E. Asnina and J. Osis, "Topological Functioning Model as a CIM-Business Model," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, New York, USA, pp. 40–64, 2011. http://dx.doi.org/10.4018/978-1-61692-874-2.ch003
[39] J. Sanders, "Spice: New directions in software process assessment," in *Proc. of Quality and Productivity in Software Development Conf.*, University of Iceland, 1970.
[40] M. Suula, T. Makinen and T. Varkoi, "An approach to characterize a software process," in *Int. Conf. on Management of Engineering & Technology*, PICMET '09. IEEE, pp. 1003–1109, 2009. http://dx.doi.org/10.1109/picmet.2009.5262007
[41] J. E. Gibson, "Cause and effect analysis: "power tool" for total quality," *Electrical Electronics Insulation Conference and Electrical Manufacturing & Coil Winding Conference*, EEIC/ICWA, Chicago, USA, pp. 751–753, 1993. http://dx.doi.org/10.1109/EEIC.1993.631320
[42] E. Asnina and J. Osis, "Computation Independent Models: Bridging Problem and Solution Domains," in *ENASE Model-Driven Architecture and Modeling Theory-Driven Development*, Portugal, pp. 23–32, 2010.
[43] F. Woo, R. Mikusauskas, D. Bartlett and R. Law, "A framework for the effective adoption of software development methodologies," in *ACMSE Proc. of the 44th Annual Southeast Regional Conf.*, New York, USA, ACM, pp. 198–203, 2006. http://dx.doi.org/10.1145/1185448.1185493
[44] E. Germain and P. N. Robillard, "Engineering-based processes and agile methodologies for software development: a comparative case study," *Journal of Systems and Software*, vol. 75, no. 1–2, pp. 17–27, 2005. http://dx.doi.org/10.1016/j.jss.2004.02.022
[45] F. Garca, M. Piattini, F. Ruiz, G. Canfora and C. A. Visaggio, "FMESP: Framework for the modeling and evaluation of software processes," *Journal of Systems Architecture*, vol. 52, no. 11, pp. 627–639, 2006. http://dx.doi.org/10.1016/j.sysarc.2006.06.007
[46] C. Gonzalez-Perez and B. Henderson-Sellers, "Modelling software development methodologies: A conceptual foundation," *Journal of Systems and Software*, vol. 80, no. 11, pp. 1778–1796, 2007. http://dx.doi.org/10.1016/j.jss.2007.02.048
[47] F. Brooks, *The Mythical Man – Month*, Addison Wesley, 1995.
[48] S. Dakhli and M. Ben Chouikha, "The knowledge-gap reduction in software engineering," in *Third Int. Conf. on Research Challenges in Information Science*, Fez, USA, IEEE, pp. 287–294, 2009. http://dx.doi.org/10.1109/RCIS.2009.5089292
[49] M. Polanyi, *The Tacit Dimension*, Routledge & Kegan Paul, 1967.
[50] M. Shaw and D. Garlan, *Software Architecture*, Prentice Hall, 1996.
[51] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundation," *IEEE Software*, vol. 20, no. 5, pp. 36–41, 2003. http://dx.doi.org/10.1109/MS.2003.1231149
[52] S. J. Bolaños-Castro, R. Gonzalez-Crespo and V. H. Medina-Garcia, "Antipatterns: A Compendium of Bad Practices in Software Development Processes," in *Int. J. of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 4, pp. 41–46, 2011. http://dx.doi.org/10.9781/ijimai.2011.147
[53] S.J. Bolaños-Castro, R. Gonzalez-Crespo, and V. H. Medina-Garcia, "Patterns of Software Development Process," *Int. J. of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 4, pp. 33–40, 2011. http://dx.doi.org/10.9781/ijimai.2011.146
[54] S. J. Bolaños-Castro, R. Gonzalez-Crespo and V. H. Medina-Garcia, "Patterns of Software Development Process," *IEEE Latin America Transactions*, vol. 12, no. 4, pp. 818–824, 2014.
[55] A. Slihte, J. Osis and U. Donins, "Knowledge Integration for Domain Modeling," in *ENASE Model-Driven Architecture and Modeling-Driven Software Development*, Portugal, pp. 46–56, 2011.
[56] J. Osis and A. Slihte, "Transforming Textual Use Cases to a Computation Independent Model," in *ENASE Model-Driven Architecture and Modeling Theory-Driven Development*, Portugal, pp. 33–42, 2010.
[57] J. Osis and U. Donins, "Formalization of the UML Class Diagrams," *Evaluation of Novel Approaches to Software Engineering*, Springer-Verlag, Berlin Heidelberg, New York, pp. 180–192, 2010. http://dx.doi.org/10.1007/978-3-642-14819-4_13

Sandro Bolanos, PhD, Pontifical University of Salamanca (Spain). Extraordinary PhD award at the same University. He graduated in Systems Engineering and Master in Teleinformatics from the Distrital University Francisco José de Caldas (Colombia). He is a member of the international research group GICOGE, Software Engineering Researcher and Lecturer. He is the Head of undergraduate and graduate curriculum projects. Currently heads the post degrees in Software Engineering and Informatics Projects at the Distrital University Francisco José de Caldas.
Contact address: Engineering Faculty, Distrital University Francisco José de Caldas Carrea 8#40–62 (Bogotá, Colombia);
E-mail: sbolanos@udistrital.edu.co

Rubén González Crespo, PhD, Dean of the School of Engineering at the Universidad Internacional de La Rioja – UNIR. Professor of Project Management and Software Engineering. He is an Honorary Professor and guest of various institutions such as the University of Oviedo and University Francisco José de Caldas. Previously, he worked as a Manager and Director of Postgraduate Department of the School of Engineering and Architecture at the Pontifical University of Salamanca for over 10 years. He has participated in several I + D + I projects such as SEACW, GMOSS, eInkPlusPlusy among others. He advises a number of public and private, national and international institutions. His research and scientific production focuses on accessibility, web engineering, mobile technologies and project management. He has published more than 100 works in indexed research journals, books, book chapters and conferences.
Contact address: International University of La Rioja Madrid office. C/ Almansa s/n. 101 Madrid (Spain);
E-mail: ruben.gonzalez@unir.net

Jordán Pascual Espada is a Research Scientist at Computer Science Department of the University of Oviedo. PhD from the University of Oviedo in Computer Engineering, B. sc. in Computer Science Engineering and a M. sc. in Web. He has published several articles in international journals and conferences, he has worked in several national research projects. His research interests include the Internet of Things, exploration of new applications and associated human computer interaction issues in ubiquitous computing and emerging technologies, particularly mobile and Web applications.
Contact address: Computer Science Department, University of Oviedo Science Building. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, Spain);
E-mail: pascualjordan@uniovi.es

Vicente García-Díaz is an Associate Professor at the Computer Science Department of the University of Oviedo. He has a PhD from the University of Oviedo in Computer Engineering. His research interests include model-driven engineering, domain-specific languages, technology for learning and entertainment, project risk management, software development processes and practices. He is a Certified Associate in Project Management through the PMI.
Contact address: Computer Science Department, University of Oviedo Science Building. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, Spain);
E-mail: garciavicente@uniovi.es

Janis Osis is a Professor at the Faculty of Computer Science and Information Technology at Riga Technical University, Latvia. He holds Dr. habil. sc. ing. degree and is an honorary member of the Latvian Academy of Sciences. The list of publications contains more than 250 titles, including 16 books. During many years his main research interest was topological modelling of complex systems. Recent fields of interests are object-oriented system development, formal methods of software engineering, software development within the framework of MDA by means of topological functioning model support.
Contact address: Department of Applied Computer Science, Riga Technical University, Setas Str. 1/3, Riga, LV-1048, Latvia;
E-mail: Janis.Osis@rtu.lv