# Specifications of the "Informatisation" Processes for Productive-Instructive Workflows

**Ioan Rosca[1], Val Rosca[2]**

(1) LICEF institute- Teleuniversity of Montreal
E-mail: ioan.rosca@licef.teluq.uqam.ca
(2) Amazon development center - Iasi
E-mail: rosca@amazon.com

**Abstract**

*Interlacing to do and to know justifies flexible support of instructive-productive workflows. To assist beneficiary systems B can consist in supply, reorganization or instruction. Support resources Rb are conceived- for emergent activities. Complex workflows can be orchestrated through Mbo models of B procedures, activated as "functions". To produce Rb resources, a P development procedure can base the "requirements engineering" on Mb models- seen as "use cases". The P workflow can be centred on the progressive concretisation of Mb as functions - that first become Mbt testbeds and then Mbs support instruments (or Mbi instruction tools). Such sequences (and any software production workflows) can be modelled as Mp "functions", acting as development coordination instruments. Applying a systemic vision, we seek the correlation between the "informatising" system P (that produces the RbMb support infrastructure) and the "informatised" system B, to facilitate the global management of the "informatisation" phenomenon: BRbMbP. The crisis of sporadic engineering can thus be surpassed, shifting from the "reengineering" and "agile" paradigms to a continuous engineering, where the target R(t) evolves within the B(t)Mb(t)Rb(t)Mp(t)P(t) system, where production, organization and instruction-form a global physiology. But such organisation formulas have a cost… that also must be managed.*

***Keywords**:* software engineering, requirements engineering, use case activation, instruction-production workflows ("functions"), knowledge (carriers) management, learning by co-doing, systemic and evolving approach, managing the "informatisation".

## 1. Introduction

In the past years, we have been studying the support of productive-instructive processes, in a sequence of LICEF projects (ADISA, GEFO, LORNET etc) based on the development of exploratory prototypes. As we progressed in the definition and implementation of behavioural specifications for procedural models, active and indexed semantically, we realized that the instrument produced during this long research-development process could have also sustained the management ... of its own evolution. This recursive situation first led us to the idea of applying the "functions" mechanism to activate the "use cases" employed in SE, then to model and orchestrate application

development processes through functions and finally to correlate the "informatised" process and the "informatising" process, by interlacing the use of functions and metafunctions. Before explaining the systemic method for managing workflows with instructive potential that we propose here for the software industry (chapter 3), we will succinctly relate (chapter 2) the vision on which the "function" procedural aggregation formula is based (Rosca and Rosca, 2008; Rosca, 2006a; Rosca and Rosca, 2004).

## 2. Facilitating processes- productive or instructive, emergent or orchestrated

### *2.1. To do and to learn*

Human action has exterior and internal causes and effects. Doing something requires the application of knowledge, and enriches competencies. Learning by practicing means to do in order to know better, and that- for doing better. An indissoluble spiral is created between learning and operating - on which the evolution of everybody's performance ascends. The intimate interlacing between production and learning is found in communities as well, both at the level of their parts (human participants bearing evolving meaning, documentary resources loaded with messages, instructively organized actions) and at that of the unitary metabolism of the respective systems, that do/learn, produce/evolve. The production and use of resources is indissolubly interweaved with the production and use of information/knowledge, forming generative cascades: objects and knowledge used for the generation of objects and knowledge used for ...(Rosca, 2006a).

Interlacing knowledge evolution and activity physiology poses a challenging theoretical and practical problem: how should we correlate methods and instruments for action facilitation with those for learning facilitation, surpassing the traditional separation, through the elaboration of mixed and flexible systems for formative action facilitation? The coordination logic of cooperative activities (like CSCW), even specialised for pedagogical cooperation (in CSCL), generally divides the operations (what the teacher does, what the student does) and is less dedicated to the instructive sharing of the same operation. Yet, co-action execution in expert-novice pair is the most natural way to interweave two production/knowledge spirals, so that the expertise of the one that does because he knows support the one that discovers because he does (Rosca I, 1999). On the other hand, many instruments pretending to be "CSCL", by lacking a "semantic layer" (not managing explicitly the knowledge involved in the operational stream) - do not allow learning to be monitored and assisted with the help of computer networks. They are dedicated to the operational management of a "pedagogic" workflow, not to the pedagogic management of an operational workflow (Rosca and Rosca, 2004). Advancement in a cooperative-instructive workflow can be governed by three interlaced logics: the productive logic of correct operation sequencing, the administrative logic of intervention coordination (according to rights, responsibilities, availability, contracts) and the informational (instructive) logic of knowledge application and enrichment (Rosca and Rosca, 2008).

### *2.2. System, process, model, lifecycle, reproduction, adaptation, evolution*

To usefully influence the work and evolution of a system (material, biological, anatomic, social, economic, conceptual), one must keep track of its global coherence and treat it as a unitary whole, with its morphology (structures) and physiology (processes) determined by its components and the relationships between them (Zadeh, 1969; Rosca, 1999). For mixed systems, in which objects, humans and concepts intervene, the involved aspects' complexity make tracing, controlling, and optimizing difficult (Le Moigne, 1987; Andreewsky, 1991). We make use of modelling structures and processes, especially when their reproduction is sought. Phenomenal reality inspires the model which in its turn inspires the production of replicas (secondary, derivate phenomena). We can thus create structures or generate process – similar to the original. Like other resources, structure or process models have a "lifecycle" (edition/conception, adaptation to various contexts, usage by those who produce structures or processes on their grounds, annotation for resuming the cycle) – which constitutes a meta-process, that at its turn can be modelled, assisted and reproduced.

Analysis and design difficulties rise when the morphology and physiology of the system, modelled in order to be understood, assisted or reproduced, changes in time, through the modification of material and cognitive components, or that of the relationships between them. The system that evolves, adapts, learns- has a plastic structure, a "longitudinal" existence (Lehmann *et al*, 2002). Furthermore, it must be considered in the context of the evolving metasystems it is part of, taking into account exterior influences and interactions with other systems. One such growth is support intervention: adding production and cooperation instruments (equipping), reorganising operations, human assistance, documentary resource supply, increasing participant expertise through instruction, etc. By enriching an assisted system *B* with a support system *Sb*, we create a new system, *BSb* – the physiology of which, containing improvements, can differ from the initial one, more or less.

### *2.3. Emergent and orchestrated modes: functions*

Long term evolutions (Mens *et al*, 2005; Lehmann *et al*, 2002) and operational chains (dedicated to instruction or with productive goal and implicit instructive dimension) can take place emergently. Participants establish what to do next at each step (in order to meet requirements, respect conditions and criteria, solve problems) and choose their support tools (using repositories of available material and human resource). In order to facilitate the retrieval of informative/instructive resources, the records of catalogued persons and documents can be indexed: competencies related to some knowledge, pedagogical potential, communicational particularities, pragmatic considerations.

In other cases, participants to a productive/instructive process conform themselves, more or less accurately, to pre-established scenarios, that suggest (impose, evaluate, facilitate) the accomplishment and sequencing of operations. To equip this second, "orchestrated", mode, combining coordinative, administrative and instructive logics, we have developed, in the GEFO project (Rosca and Rosca, 2004), the prototype of the

"function" manager (a biological metaphor, suggesting the physiology of knowledge management, within intelligent collective organisms).
As can be seen in *figure 1*, the first step of a "function's" lifecycle is to model the observed or imagined process. A generic model is defined, composed of abstract elements: "operations" (represented with ovals, that will become executions), "actors" (represented with hexagons – that will be concretised with participants) and "instruments" (represented with rectangles – that will be concretised with resources). The choice of real elements (using specialized repositories) can be organized in a distinct phase (adaptation to a beneficiary's context) – or can be distributed in time, from edition to the execution (enactement, see Ventroys and Peter, 2003) which concretises the operation. The prepared facilities are exploited, by manipulating, in parallel, the *R* resources and the functional model- while resolving ergonomic issues (through presentation of both windows, or "hiding" resources behind the steering model, or hiding the supervising model behind the resource's window) and technical interoperation issues (between the interface, business and data layers of the *R* and *M* applications). Based on execution results (traces, productions and annotations), reports can be created, evaluations made and corrections directed.

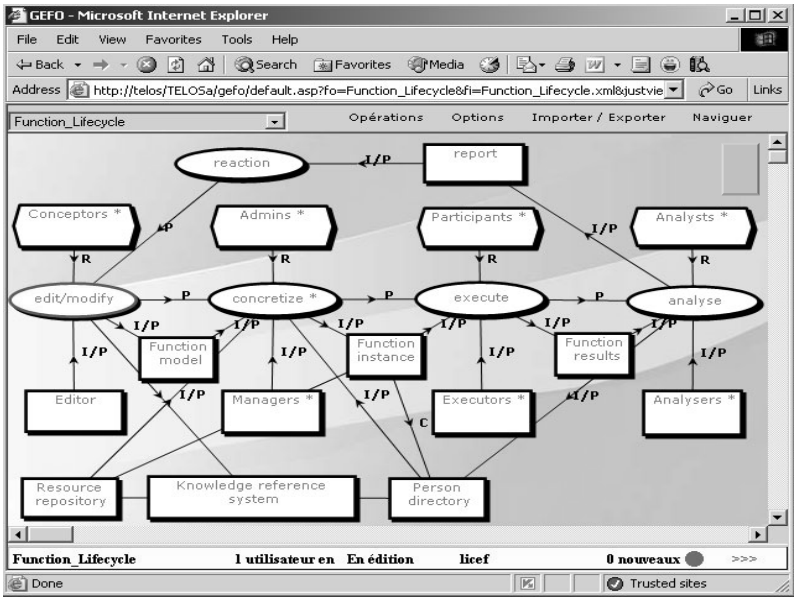


*Figure 1.* Functions and metafunctions

The function processing chain, represented in figure 1, being at its turn a process, can be handled functionally, thus obtaining "metafunctions", useful in the global management of procedures *Pb* used for supporting the procedure *B*. Complex (instructive) functions or "operations" (interfaces for realising a unique activity) can be

managed as any resource, having characterisation records- including competency descriptions (what someone must know to approach them and what he learns by executing them). Offering fine-grained cognitive interconnection services requires a proper preparation (internal indexing) of functions (Rosca, 2006; Rosca, 2008). The intervention of a semantic service (connection, verification) agent for any sub-operation of a function (workflow of a modelled procedure) requires some preparing steps: 1. Indexation, related to the involved knowledge *K,* of the action to be realized *o(k)*: required level- *coi(k)* and level obtained by practicing- *cof(k)* 2. Presumed competence for the executing (beneficiary) actor- *b(k)*. 3. Pertinence of the presumed support documents *d(k)*: from what level *cdi(k)* to what level *cf(k)* they can guide the user. 4. Pedagogical competency of the *a(k)* presumed assistants, indexed by the (*cai(k), caf(k)*) pair. The competence equilibriums monitored are related to the relationships between these abstract elements and the components that progressively concretise them. For instance, when concretising an abstract beneficiary *b* with a person *P,* having a *cp(k)* competency, we can verify if the competency condition required for realizing the operation is satisfied, depending on the other components that have already been chosen (assistants, documents) and on the presumed competencies of the still unconnected components. Thus can be sustained services such as: signalling competency disequilibria, matching a participant that could optimise assistance etc.

## 3. Application: the systemic engineering of "informatisation" workflows

### *3.1. Progressive use-case activation in software engineering*

Modelling processes that take or should take place within a technically assisted social system (Herrmann, 2004), requiring representation and coordination techniques for man-machine orchestration, was intensely approached in software engineering (Hermann at al, 2004; Noelle *et al*, 2002). Between the founders of UML, Jacobson was the one who promoted standardising the modelling of sought processes ("use cases") as central design instrument (Hollub,2001; http://www.uml.org). In the RUP methodology (Larman et al, 2002; http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process), an application is developed by continuous reference to these physiological descriptions (see Fowlers' observations at http://www.martinfowler.com/articles/newMethodology.html#N401).

Apart the dialogue between architects and developers, the relationship with a client imposes the description of the aimed processes, in textual or graphic languages, more natural-culturally, exploited in "scenario-based design" (Bustard *et al*, 2000; Caroll, 2000). In "requirements engineering" specification extraction techniques have been elaborated, including use case management methods (Anton *et al*, 2000).

In this context, model activation formulas, with the help of the "function manager", can intervene. The proposed method is described (illustrated) in *figure 2*:

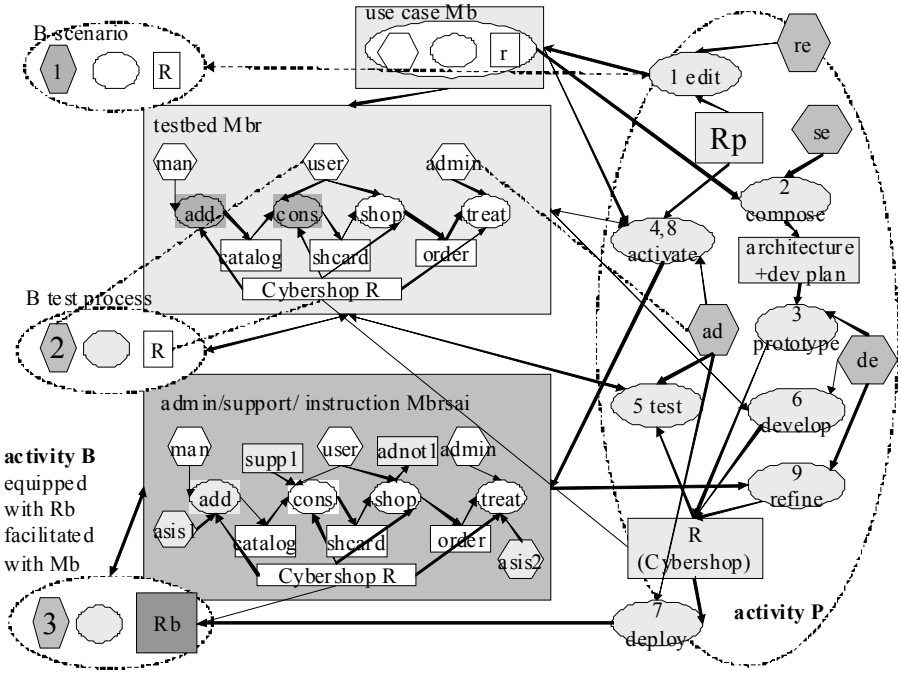


*Figure 2.* Lifecycle of an active use case

Let us consider a system *B* – that will be "informatised" by introducing an *Rb* resource (program, application, infrastructure) – transforming the system into *BRb*. The *Mb* modelling of the current, un-equipped processes (or *Mbr* of the aimed processes, equipped with *Rb*) is usually done in passive forms. Three distinct phases are sequenced: modelling (the *B-Mb* relationship), development (the *Mb-Rb* relationship) and use (the *B-Rb* relationship). By enriching the procedural model *Mb* with facilitation layers for participation to B processes (including access intermediation to the built resource *Rb*), we can go from a simple "use case" to an orchestration function –- *Mbro*, a facilitation interface, placed between *B* and *Rb*. Such use-case processing permits a coherent management – in RUP vision – of the entire lifecycle of a software product, from the *Mb* edition phase to the *Rb* production phase (based on *Mb*) and afterwards to the use of an activated model *Mbr* as an interface for prototype testing – *Mbrt*, or as a tool for user support- *Mbrs*, operation administration – *Mbra* or user instruction – *Mbri*.

Observing (or imagining) a *B* application scenario (for instance – the use of a virtual store by those responsible of updating the product catalogue, then by the end-user that adds products to the shopping cart and places an order and finally by the administrator that handles commands) – we can edit (phase 1) a first form of an *Mb* "use

case", a passive model of the *B* operational chain, that can be used as inspiration source by the conceiver of the desired application's architecture (phase 2) or by the prototype developers (phase 3). Afterwards (phase 4), by adding functional facilities (sequencing, coordination, annotation, binding to existing resources and to the tested prototype), we attain a facilitating or orchestrating use case – *Mbr* or *Mbro* – upon which can be organized the activity (phase 5) of testing the procedure assisted by the evolving prototype *Rb*. The development cycle (phase 6) – (or the modelling) can be resumed, for the progressive improvement of *Rb* – fuelled by the data captured by the *Mbrt* test device.

After finalizing *Rb*'s development and installation in the beneficiary's context (phase 7), we can operate (phase 8) a new functional activation (preparation) of *Mbr*, leading it to an *Mbra* management instrument (model centred on the enrichment of administrative logic- access and surveillance) or to an *Mbrs* support instrument (centred on the assistance facilities). The *Mbr* use case becomes a facilitation tool for the *Rb* application's use within the beneficiary system *B*. Preparing instructive co-action mechanisms and semantic services, the final activation can transform *Mbrs* from a support instrument to an instruction instrument *Mbri* (of tutorial type, with human assistants and support documents or with automated advises). An *Mbrasi* model – rich in all facilitation layers – will provide coordination, administrative, support and instructive services – and thus will be usable as a plastic form of support for the *B* system, optimising the added infrastructure's use. Tracing and annotation capabilities intermediated by *Mbr* can also be used for the inspiration of refinements and corrections to the *RbMb* application (phase 9).

### *3.2. Active modelling of development workflows*

Building resource-applications *Rb*, passive models *Mb* or active interfaces *Mbro* is accomplished in a software production process *P*, that exploits specific resources. *Figure 2*, for example, illustrates a programming method *P* based on the "functional" management of use case evolution. However, we can also resort (figure 3) to a *Mp* functional modelling of the proposed method's flow, usable for the orchestration (management, coordination) of *P* programming activities or for orienting the conception of an *Rp* resource, necessary to programming. Workflow management is now both target (in system *B*) and instrument (in system *P*).

Software engineering is dedicated to optimising processes that take place in the *P* system. Facilitating these processes can be accomplished through: equipping, reorganization, assistance, instruction. We can again compose *Mp* models and construct *Rp* applications that can be accessed through activated versions of models – *Mpr*, having enriched coordination, administration or instruction logic layers. Making abstraction of the beneficiary systems B-n (we'll come back on this in 3.3), let us observe that activating the programming workflows *Mp* implies new functional cascades: reality-model-construction-application. The considerations presented in 3.1 can thus generate, through particularisation (specialisation), specifications for programming support instruments, based on the activation of models representing design procedures. We reach the management of a "programming case's" lifecycle:

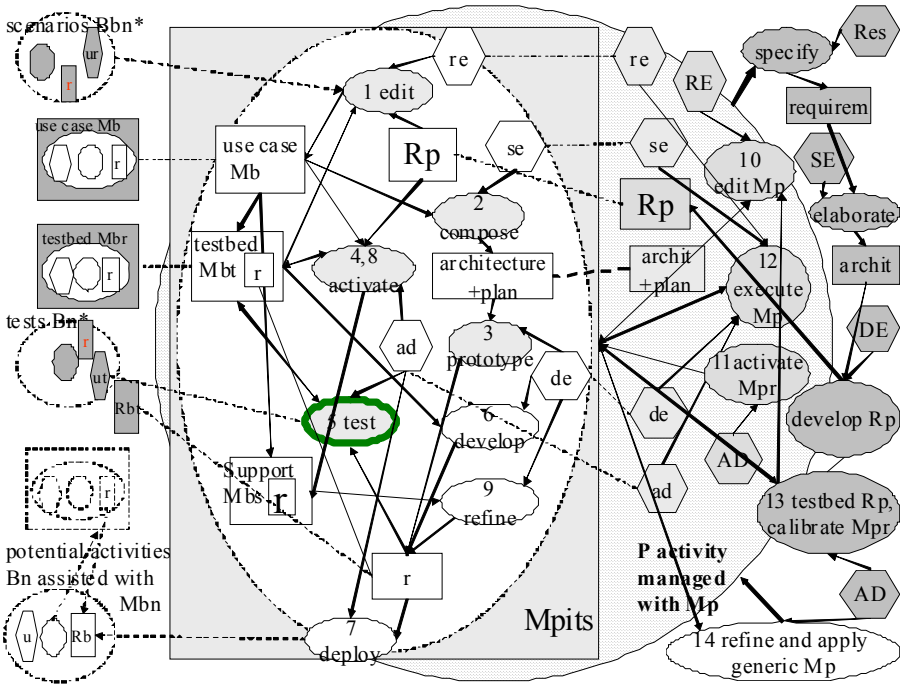


*Figure 3.* The lifecycle of an active programming model

The *Mp* software production process's model is obtained through manual edition (phase 10), but its extraction by interaction interception – within a pilot (demonstrative) programming process – is not excluded. The sequencing and coordination logic of the (team) programming process – expressed by the *Mp* functional workflow – can correspond to methods of work organization in SE, be subject to standardization (norms) for production quality evaluation or facilitate the project continuation by other persons or teams etc. By an *Mpr* correlation between *Mp* and *Rp* and the activation of the model (*Mpro*), we can coordinate the programming activity and facilitate the use of the *Rp* instruments, eventually rebuilt upon the *Mpr* model (phase 11). The execution of these development scenarios (phase 12), for building various *Bn* applications – exploits the facilities organized during the preparation phase of the *Mp* development model.

Enriching the *Mp* model can also be accomplished, in order to test the *Rp* instruments or calibrate the *P* method by creating (phase 13) *Mprt* models, allowing the *P* development method to be tested. These calibration tests, supervised by an *AD* meta-administrator, can orientate the modification of the programming workflow or of an *Rp* programming instrument. The development of *Rp*, conceived by the *DE* meta-developers, can be resumed, upon the meta-architecture elaborated by *SE* meta-engineers, based on the behavioural specifications defined by the conceptual researchers. After refining the development method and the design support instruments, we can implement (phase 14)

new programming management facilities, by processing (activating) *Mp* models so that they become *Mpr* tools, specialized as "test", "support" or "tutorial".

Before applying generic organization functions to the development of a particular application *B* (see 3.3), we recall that the flexibility of the *Mpr* model permits assistance metamorphosis between: action inspiration and guiding, project advancement observation, performance analysis and evaluation, information and learning, communication and cooperation coordination, adapted human and documentary support, locating, accessing and manipulating programming instruments, operation automation, intervention management – according to rights, contracts and policies.

All these facilities, but especially the evolving explicitation of cooperative programming procedures, the operation sharing (so that they can be effectuated in expert-novice instructive tandem) and the competence explicitation (to keep track of programming knowledge evolution) – create an appropriate context for an "as you go" integration method of interns, in programming teams, allowing a smooth transition from the initiation through instructive participation to the productive participation, with continuing knowledge development. By refining access logic, we can support the formation and work of ad-hoc constituted programming teams- according to availabilities and the existing conventions, in a distributed programmer community (Rosca and Rosca, 2002; Lee, 2003). The optimisation services for the concretisation of human and document resources bound to the development workflow require the indexation of person, resource and operation repositories- using an adequate knowledge reference system. Using the participants' competencies (programming expertise), the support documents' pedagogical pertinence or the opportunity of available training operations-competency equations can be solved, resource selections and role allocations made and other services of expertise management offered.

There are frequent cases (Larman, 2002) when programming activities are too complex, variable, un-reproducible – to be meticulously planned, efficiently. In such cases, operation sequencing is done emergently, participants deciding what they do at each stage, choosing the required resources and respecting a given set of rules. Using *Mp* workflows can intervene even in such cases, after realizing productive cascades (to illustrate them a posteriori), using various observation methods (action interception, annotations, etc). Organizing knowledge reference systems specific to the programming activity and indexing resources and expertises upon them – can therefore be useful, regardless of the production mode (emergent or orchestrated).

### *3.3. The systemic engineering of "informatisation" workflows*

Isolating the programming activity from the physiology of the systems for which it builds support tools- can be useful, only when we seek optimisation of some generic processes and instruments. However, the *P-n* activities of developing *Rb-n* tools (including passive *Mb-n* and active *Mbr-n* models) for the support of processes within the *B-n* beneficiary systems generally depend strongly on the *B-n* system's metabolism, as suggested by the *Pb* indexing. Understanding the functioning of the system of systems: *BMbrRbPb* has major consequences for the success of "informatisation" projects.

Isolating the informatising process *P* from the informatised process *B* (reducing the relationship between them to the – "deliverable" – product *Rb* that *P* offers to *B*) has led to superficial treatment of the beneficiary's requirements (Rosca, 2006b).

If the beneficiary *B*, confronted with new situations (changes in objectives, priorities, equipment, criteria, etc) requires a modification of the *Rb* infrastructure, a reconstruction process is run (in which *Rb* is enriched or modified.). The scale of "reengineering" needs (http://www.sei.cmu.edu/reengineering/) has caught the software industry off guard (Larman, 2002), diminishing reproducibility. Hasty approaches lead to endless requests for corrections. Careful ones delay application to the point where initial specifications no longer correspond. Teams called on to resume a project encounter major difficulties in understanding and continuing what has been done in the antecedent construction, asking for a solid documentation, costly for the initial designer. Supporting very labile informational systems must surpass the paradigm of cascading requirements/ development/ application) cycles. Socio-economical systems change continuously, become. An infrastructure that pretends to sustain their physiology should evolve continuously, ameliorate progressively, become – together with the beneficiary system. "Continuous software engineering" requires a cybernetic view of the relationship between the producing system *P* and the beneficiary system *B* – forming an evolving meta-system. For the support application *MbRb* to evolve together with the *B* system that solicits and uses it - you must manage the evolution of *Mb* and *Rb* following their coupled long-term lifecycles. All these can lead to a global management formula, based on functions.

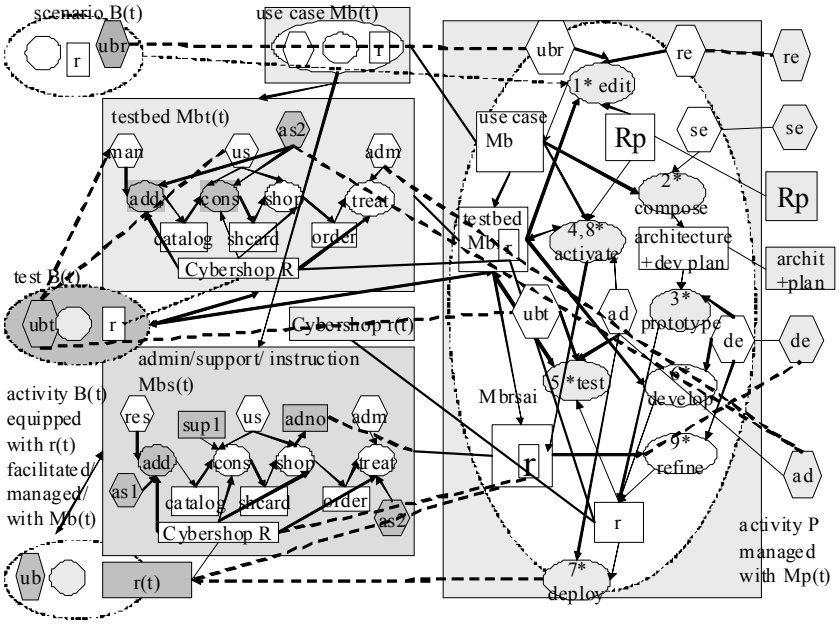


*Figure 4.* Global management of the informatisation process's evolution

*Figure 4* illustrates the procedural flow of the global evolving system *BMbrRbPMprRp(t)* – presuming that the represented processes are resumed, as many times as necessary (see the * symbol). The beneficiary system *B* has an evolving morphology and physiology *B(t)*, even the continuous improvement of the *Rb(t)* resources it employs – being a transformation. We can establish multiple connections between the components of the *B* and *P* systems (managing the interference between *B* and *P* processes- being the goal of model 4). It could be the communication between two persons from the *B* and *P* spaces, their coordination in common processes, resource sharing, mutual support – with the occasion of collaboration in the *Mb(t)* scenario's definition, solution testing with the help of the *Mbrt(t)* model, *Rb(t)* product installation or the analysis of data collected by the support interface *Mbrs(t)*.

Processes in *B* are observed in a convenient rhythm, and used by specification engineers to edit and modify the *Mb(t)* models – following observations made in the method's upstream (but downstream in the temporal evolution spiral sense). Using these models, developers, organized within the *P(t)* evolving physiology programming process – program or reprogram the *Rb(t)* instruments. Also within the *P* system takes place, in several rounds, the *Mbr(t)* test models' activation (reactivation), by enrichment with improvement layers of the coordination, negotiation and annotation logic. The active *Mbr(t)* models, installed at the beneficiary, along with *Rb(t),* and used for the inspiration/orchestration of activities in B, facilitation of work with *Rb* instruments, participant connection or instruction – can be progressively improved. Interactions between the *Bx* components of the *B* phenomena and the *Py* components of the *P* phenomena can be realized by sharing the *Rb* product, through the *Mb* or *Mp* models or outside them: *BxPy*, *BxRbPy*, *BxMbPy*, *BxMpPy*, *BxMbRbPy*, *BxRbMpPy*, *BxMbMpPy*, etc.

The *P* process cascade that supports the *B(t)* system's evolution by equipping it with an evolving *Rb(t)* support can be modelled in passive forms *Mp* or orchestrated with active *Mpro* models – that may facilitate the integration of new persons in the development team or the transformation of the *Rp* programming instruments. As the software industry is in continuous "progress", the activity organization formula within the *P* system is also evolving (see chapter 4.2). Another cause for *P*'s organization change can come from the requirements emerging from *B* – that can't be optimally satisfied anymore by the old development formulas. Processes represented in the *P* layer must therefore evolve, adapt, sustain a labile *P(t)* physiology, axed on the evolution of *Mp(t)* models and on the use of programming instruments in continuous renewal: *Rp(t)*.

Figure 4 thus represents an evolving engineering, based on the continuous update of *Mb(t)* models and *Rb(t)* resources – within evolving design procedures *Mp(t)*. The global bi-layer stream represents the correlated history of the informatised – informatising system *BMbRbPMpRp(t)*. By organizing it as a function (enriching the coordination, negotiation, assistance and instruction layers), we can obtain an orchestration tool for the informatised-informatising ensemble, a management instrument for *P* design workflows that supports some *B* use workflows. Modelling such a management mode could at its turn be represented or piloted by higher-rank models. This "organic", recursive and "genetic" approach leads us, in the SISIF project, to a continuous engineering of informatisation: the modification of *MbRb(t)* in the labile *Pb(t)* workflow so that *B(t)*'s optimal support can be attained, along its evolution.

## 4. In guise of conclusion: the problem of costs

Informatisation is not limited to the development of applications. It also consists in the amelioration of the informational physiology, the informatising process continuously supporting the evolving system in which the informatised processes take place. A process can be assisted by facilitating the access of participants to resource repositories or by planning passive or active workflows. For the computer network to operate "synaptically" (Lee, 2003) matching partners (and then eventually supporting their cooperation with workflows having pedagogical potential) – a "knowledge" explicitation is necessary.

However, the organization of these facilities involves costs and is not profitable if a certain reproducibility does not occur. Benefits must surpass the preparatory expenses (Olufunmilayo, 2006). The problem of economic management of knowledge evolution (Rosca, 2008a) is complementary to that of the economy of operational workflow management (Rosca, 2008b) Informatisation (automation) projects should not be accepted without doubts, from technological enthusiasm or blind obedience towards the obligation of updating- pushed by an imaginary competitiveness (Rosca, 2006b). Huge funds have already been expended for "necessary" transformations with unknown local and global value, neglecting the real profitability issue, or lacking an appropriate analytical apparatus.

The stratified question we conclude our analysis with is:

> *on what analytical grounds can we solve economical problems such as:*
> "Is it worth (according to chosen criteria) to organise resource repositories, model and activate workflows and explicit knowledge at a certain granularity level – in order to support a class of productive-instructive processes/applications, that will take place in a system, evolving in a certain period?".

We have been confronted to this issue on the course of our participation to the LICEF chain of projects (that have raised many questions regarding the efficiency of instruction technologisation efforts)- decomposing it in a few questions: "*How profitable is the meticulous management of metadata records for components that can participate to the processes of a socio-technical system?*"; "*To what degree is the modelling of processes that take place in the system, or the activation of these models (to be used as orchestration instruments) – profitable?*"; "*Is the explicit management of knowledge and competencies advantageous in comparison to the case where their evolution intrinsically results from the activity of knowledge bearers?*".

We have sought a synthetic expression for these questions, in the form of "profitability equations".

***1. Equipping emergent operations***. In comparison to unprepared emergent (programming) operations, preparing *Rp* material and *Hp* human resources (metadata record catalogue, etc) introduces additional management costs *Cmr* and *Cmh*. The profitability condition in this case is that the additional gain *G* be greater than the organization expenses:

(1) $$Cmr + Cmh < G\ ?$$

***2. Semantically supported emergence***. When resorting to the explicitation of persons' competencies *Hk* and pertinence of documentary resources *Rk* relating to *K* knowledge (reference prepared by the edition of knowledge structures and competency norms used as reference system), additional expenses are introduced for the semantic support of emergent operations: managing knowledge *Cmk*, human competencies *Chk* and documentary pertinence *Crk*. Facilitating resource retrieval – brings a *Gk* benefit. The profitability condition is that the additional benefit (in contrast to using semantically unindexed repositories) surpass the additional expenses:

(2) $$Cmk+Crk+Chk<Gk-G?$$

***3. Operation orchestration.*** Another possible improvement, is the use of activated operation models ("functions") *O*, that facilitate the inspiration of sequencing, access management and resource manipulation, collaboration coordination, etc. In this case, the profitability condition is that the additional gain (to the simple emergent case) *Go-G* surpass the *Cmo* expenses of organising operational models (edition, activation, etc):

(3) $$Cmo<Go-G?$$

***4. Semantically supported orchestrations.*** The last situation is that where both repositories and operational models *O* are semantically indexed on K reference systems. The realized benefits (possibility to offer optimization services, matching services, etc) *Gok* must justify the *Cok* expenses for procedural model indexation:

(4) $$Cok<Gok-Go?$$

In the absence of a "magic formula" for solving these open problems, we can formulate some orientations:

1. Estimating the global profitability of a global engineering approach requires a rigorous meta-analysis (using intuition, trends and slogans being legitimate… only when the cost of meta-analysis is – too – prohibitive). 2. The analysis must begin with the clear enunciation of the priorities (values) chosen by those involved in defining the specifications. 3. The methods can be judged and combined, according to the particular context: Are we dealing with a large scale application – posing retrieval problems – or with a small scale one – allowing easy participant orientation? Is the need for discretion/confidentiality a priority, or that for tracing/evaluating the activities and knowledge evolutions? Do we wish to conserve the system and only ease some operations or to ameliorate the physiology? Is the involved knowledge stable, or in continuous change? Do collective, or individual interests, take precedence? Spiritual, or material goals? Minimum price, or maximum quality? Do we want to exploit human intelligence or automatisms?

When the reproducibility of the assisted operations is substantial (see situations like Google, EBay, Amazon, etc), efforts for preparing retrieval and matching, based on semantic inferences – are justified. For unrepeatable phenomena, with rapidly changing topology and physiology – the investment in semantic explicitation (to permit computer

intervention) will not redeem, exploit the participants' intelligence being more profitable. In intermediary situations, the problem may become undecidable, due to barriers created by complexity and even by epistemological difficulties.

The problem of managing… management costs (also see comments at http://www.strassmann.com/pubs/measuring-kc/) remains provocative. We will try to deal with it as "recourse to the method": modelling the estimation process.

## REFERENCES


ANDREEWSKY, E. (1991), *Systemique & Cognition*, Dunod, Paris, 1991.

ANTON, A., DEMPSTER, J., SIEGE, D. (2000), *Managing Use Cases During Goal-Driven Requirements Engineering*, http://www4.ncsu.edu/~aianton/pubs/icse2000.pdf

BUSTARD, D. W., HE, Z., WILKIE F. G., (2000), Linking soft systems and use case modelling through scenarios, Interacting with computers 13 97-110, 2000, Elsevier Science.

CAROLL, J. M. (2000), Five reasons for scenario based design, Interacting with computers, 13 43-60, 2000, Elsevier Science.

HERRMANN, T., HOFFMANN, M., KUNAU, G., LOSER, KAI-UWE (2004), A modelling method for the development of groupware applications as socio-technical systems Behaviour & Information Technology Vol. 23, No. 2, 119-135, 2004, Taylor & Francis Ed.

LEE, Y, CHONG, Q. (2003), Multi-agent systems support for Communities-Based Learning, Interacting with computers 15, 33-55, 2003, Elsevier Science.

LARMAN, C., KRUCHTEN, P., BITTNER, K. (2002), How to Fail with RUP Process: Seven Steps to Pain and Suffering, http://www.agilealliance.org/articles/larmancraigkruchtenph/fi

LEHMANN, M. M., G. KAHEN & J. F. RAMIL (2002), Behavioural modelling of long-lived evolution process- some issues and an example. Journal of software maintenance and evolution: research and practice, 335-351, 2002, John Wiley & Sons Ed.

MENS, T., WERMELINGER, M., DUCASSE, S., DEMEYER, S., HIRSCHFELD, R., JAZAYERI, M. (2005), Challenges in software evolution, Principles of Software Evolution, Eighth International Workshop on Volume, Issue, 5-6 Sept. 2005, pp. 13-22.

LE MOIGNE, J. L. (1990), La modelisation des systemes complexes, Dunod, Paris.

NOELLE, T., KABEL, D., LUCZACK, H. (2002), Requirements for software support in concurrent engineering teams, Behaviour and information technology, Vol. 21, No. 5, pp. 345-350.

OLUFUNMILAYO, B. A. (2006), "Measuring and Representing the Knowledge Economy: Economic Reality under the Intangibles Paradigm" Buffalo Law Review 54 , 1-102.

ROSCA, I. (2006a), Knowledge and object phylogenetic production cascades – the TELOS case, CE'2006 conference, Antibes, Proceedings in Leading the web in concurrent engineering, Parisa Godus & Others Ed., 2006, pp. 296-304, IOS pres, 2006.

ROSCA, I. (2006b), The risks of virtual learning ICVL'2006 Proceedings, Bucharest Univ Press, Marin Vlada & Others Ed.., pp. 155-163.

ROSCA, I. (1999), Towards a systemic vision of the explanation process; the story of a research on integrating pedagogy, engineering and modeling- PhD thesis, Montreal.

ROSCA, I. and ROSCA, V. (2008), Meaningful access: policy, management and orchestration. IJAMC (in print).

ROSCA, I. and ROSCA, V. (2004), Pedagogical workflow management with functions, LOR'04 congress, Montreal.

ROSCA, V. (2008a), Explicit knowledge management and management of knowledge carriers. Graduation Paper -Economy-Management, Iaşi, 2008.

ROSCA, V. (2008b), Connections between expertises, activities and documents in Knowledge Management, Graduation Paper -Management, Iaşi, 2008.

VANTROYS, T., PETER, Y. (2003), Cow, a flexible platform for the enactment of learning scenarios, CRIWG proceedings, Autrans, France, 2003, Springer Verlag.

ZADEH, L. A. (1969), System theory, McGraw-Hill, N.Y., Toronto.