Software or Service? That's the Question!

Luigi Buglione[1,2(✉)], Alain Abran[2], Christiane Gresse von Wangenheim[3], Fergal Mc Caffery[4], and Jean Carlo Rossa Hauck[3]

[1] Engineering Ingegneria Informatica Spa, Via R. Morandi 32, 00148 Rome, Italy
luigi.buglione@eng.it
[2] Ecole de Technologie Supérieure (ETS), Montréal, Canada
alain.abran@etsmtl.ca
[3] Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
{gresse,jeanhauck}@gmail.com
[4] Regulated Software Research Group & Lero, Dundalk Institute of Technology, Dundalk, Ireland
fergal.mccaffery@dkit.ie

Abstract. In Information and Communication Technology (ICT) a 'deliverable' may be either software (perceived as an 'output') or a service (perceived as an 'outcome'). On the one hand, the differences between software and service have led to the design of parallel models and lifecycles with more commonalities than differences, thereby not supporting the adoption of different frameworks. For instance, a software project could be managed applying best practices for services (e.g. ITIL), while some processes (e.g. Verification & Validation) are better defined in models of the Software Management domain. Thus, this paper aims at reconciling these differences and provides suggestions for a better joint usage of models/frameworks. To unify existing models we use the LEGO approach, which aims at keeping the element of interest from any potential model/framework for being inserted in the process architecture of the target Business Process Model (BPM) of an organization, strengthening the organizational way of working. An example of a LEGO application is presented to show the benefit from the joint view of the 'software + service' sides as a whole across the project lifecycle, increasing the opportunity to have many more sources for this type of improvement task.

Keywords: Software management · Service management · ISO 20000 · CMMI-DEV · CMMI-SVC · ITIL

1 Introduction

To classify items human beings create mental boundaries for distinguishing items, including adopting different terms: this is a classical approach for benchmarking purposes. For example, in the Automobile market SUVs or Crossovers are recognized as distinct car segments by adopting a number of criteria, for instance their length and main characteristics. Again, in the Telecom market smartphones, tablets or 'phablets' are now recognized as distinct kinds of products mainly according to their size etc. However, when classification rules become too strict we may risk losing the 'big picture'.

A. Kobyliński et al. (Eds.): IWSM-Mensura 2015, LNBIP 230, pp. 30–45, 2015.
DOI: 10.1007/978-3-319-24285-9_3

This can also be observed through the division of three main groups of processes (development, operation, maintenance) of a system throughout its lifetime. Here the first and the third group are often associated with the Software domain, while the second is associated with the Service domain. In this context CMMI-DEV [1] or ISO 15504-2 [2] now in the ISO 33000 series) are examples of process improvement models for the Software domain, while, ITIL [3], MOF[4], CMMI-SVC [5] or eTOM [6] are examples for the Service domain. Yet, a mix of 'components' from models of those two 'separated domains' is rarely observed. However, when analyzing the existing models/frameworks both at the process level and product level, the differences are not as sharp. For instance on the process level 16 out of the 22 processes within CMMI-DEV and CMMI-SVC are about the same, with very slight differences, mostly in the glossary adapted to the specific tasks to be performed [31].

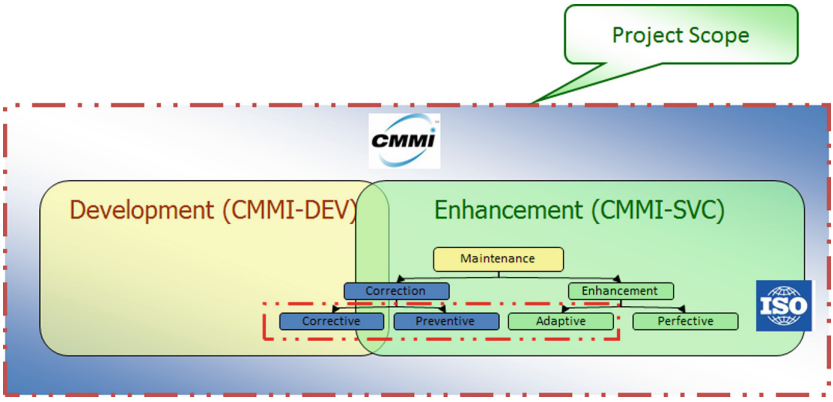

Fig. 1. Mixing CMMI constellations (DEV + SVC) into a unique 'project scope'

Looking at the product level, the quality model for a software product in the ISO 25010 standard [7], if the term 'software product' was substituted by 'service', the list of categories and sub-categories could still be a good fit for a pure service, with such a service not being necessarily an ICT-related service (e.g. a service should be designed to be maintainable, reusable, usable, reliable, etc.). A question that arises is: are software and services different or not? And, if not, how to preserve the best aspects from any existing model/framework/classification currently available within both communities?

Furthermore, how to lower the Total Cost of Ownership (TCO) for the management of a project? And could Knowledge Management (KM) be part of such a solution? In this respect, this paper attempts to answer these questions by introducing an improvement approach based on merging elements from different frameworks and models, having in mind one final goal: to reinforce the organization's Business Process Model (BPM) respecting, instead of upsetting, its architecture (Fig. 2).

This paper is structured as follows: Sect. 2 describes the main differences and commonalities as they are perceived by both communities of interest (software and service). Section 3 discusses how to merge best practices from other models as improvements to be integrated into an existing organizational BPM using the **LEGO** (Living EnGineering process) approach [8]. Section 4 presents an example using a typical ICT management case. Section 5 presents some conclusions and suggestions for improvements.

2 Software vs Service? Friends or Foes?

This section describes briefly the main differences and commonalities between products and services. According to ITIL and ISO 20000-1 [9]) '*a service is a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks*'. ISO 20000-1 adds a note that '*service is generally intangible*'.

2.1 Differences

A typical difference between product and service is the level of tangibility of a deliverable. A product (such as a table) is more tangible than a service that is typically intangible (e.g., the value perceived from whatever experience and therefore perishable, with the need to reproduce the same level of quality each time (QoS – Quality of Service)). Whereas a product – once produced – may be used many times and typically exhibiting the same level of quality (QoP – Quality of Product[1]).

In the early 90's, ISO published two similar but distinct standards for managing quality at the organizational level: ISO 9001 for the products and ISO 9002 for the services, whatever the application domain. Later, the so-called 'Vision 2000' project reconciled the two standards into a single one: ISO 9001:2000 indicated that while the formal term adopted in the document was 'product' it was intended as a 'product/service', with the ultimate aim being to achieve customer satisfaction. In particular, the ICS (International Classification for Standards) code distinguished the working sector for an organization to be audited[2]: EA33 is the code for those organizations managing software, while EA35 is the code for organizations managing services.

In relation to services for developing software ISO has published the ISO 20000 series, based upon ITIL (IT Infrastructure Library), the UK standard of best practices for the IT Service Management (ITSM) community. While some concepts are differentiated (e.g. service catalogue, risk register, capacity management), other elements (such as the process improvement approach based on the Deming's PDCA cycle) are the same in both standards. ITIL details the '*7-Step Improvement process*' stressing the

[1] http://asq.org/services/why-quality/overview.html.

[2] http://www.iso.org/iso/home/store/catalogue_ics.htm.

role of a proper Knowledge Management (KM) process for a more effective continuous improvement.

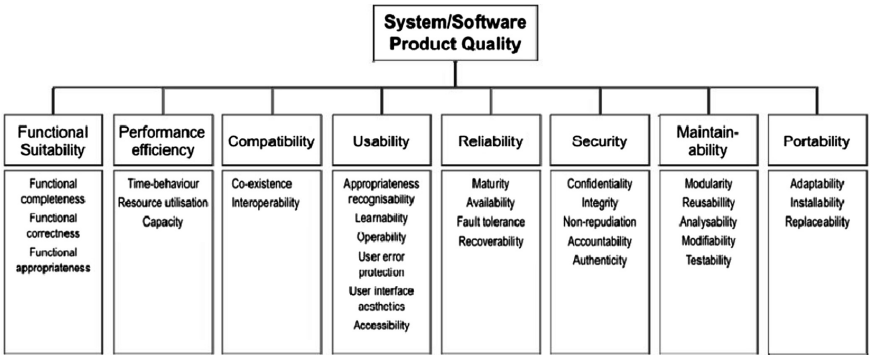

Fig. 2. ISO 25010:2011 quality model [7] – also for service?

2.2 Commonalities

There are different types of service. For instance, maintenance is a service; operation is a service, in addition to both developing and managing software also being a service. In order to understand how the 'boundaries' of both standards overlap, evidence from one set of standards can be mapped to evidence from the other one (e.g. taking ISO 20000-1 [9] requirement and applying them to a software development project or, alternatively, taking ISO 9001 requirements [10] and applying them to a pure service). Their differences seem to be mostly in the terminology adopted rather than the actual contents.

A 'service catalogue' is a library of services: applying the same concept to any type of asset library, it works well using libraries with software code for reuse. Furthermore, the ITIL definition in the Definitive Media Library (DML) where software and related documentation and licenses are stored makes use of the same configuration management process. Some specifications ('Service Asset and Configuration Management') do not change the inner content of the process but extend the common concept from software code to any kind of organizational asset (including HR-asset) needed as 'components' of a service, that is - conceptually speaking - wider than the software itself.

2.3 The Challenge – Possible Benefits

The commonalities appear to out-number the differences. Could this therefore indicate that an organization could be made more successful through using more knowledge from models/frameworks typically developed and adopted by each of the two communities? Here it follows a list of candidate drivers for stimulating a positive and

effective change and possible benefits from the joint usage and vision of software + services as a whole within the 'project' umbrella:

- **Unique, Continuous Lifecycle and Process Flow:** what the final customer is asking for is a not a product itself, but the value that such a product (software or system) can provide to his users by its usage. Thus, the whole project can be split into a series of sub-projects (or iterations-sprints with several deliverables, outputs (e.g. software or a user manual) plus outcomes (e.g. training, positive perception after the usage), as in Fig. 1. Organizations often tend to have a group for the development part and another one for the maintenance part, creating possible logical breaks in terms of service continuity and value provided, while a customer would expect such continuity defined as 'availability' as stated in the agreed service levels to be as high as possible.
- **Glossary:** a Change Request (CR) is exactly the same as a Request for Change (RfC) or a Modification Request (MR): these are different terms for the same concept. This happens also for other terms: for instance, the internal 'capacity' of a service team can be associated to the 'productivity' levels needed to do a project estimate, etc. However, frequently words can create barriers that may be difficult to overcome and they may reduce the initial effectiveness when people with different professional paths work together. A common, shared glossary – including a list of most used acronyms – could help in speeding the communication among people within a team.
- **Knowledge Management (KM):** the input for improving anything is to know and, better, to know how to do better things. Most of the software process models assume that such knowledge is already part of a team, while often it is not. Thus, an organization stimulating creativity and knowledge sharing (e.g. the SECI model [14]) could have a greater probability to be effective on the market than its competitors.
- **Product/Service:** the more tangible product is the means for providing a service to the final users. In that sense, the management of a project should be more service-oriented because the focus in the mid-long term should be in measuring not only the ROI (Return on Investment) but also the VOI (Value on Investment). This includes also what is generated by intangibles.

2.4 A Short Example – Software + Service Together

A further example comes from project 'lifecycles'. The Agile approach was born in the mid '90 s for managing Telco projects with unstable requirements and short lead times for delivering the 'products'. When looking at an Agile Project Management (APM) method such as Scrum [11], it is possible to adapt it to a pure service using, for example, a revised version of user stories, US^2 (2^{nd} generation of User Stories (US)) [12]. This is illustrated in Fig. 3: a US^2 card adds: (a) sizing FUR and NFR by functional and non-functional sizing units (fsu/nfsu) or – as in typical US cards – directly assigning the effort in person/days (or person/hours); (b) specifying a priority (after the INVEST grid evaluation [13], see below) according to the well-known

'MoSCoW' (Must or Should, Could or Would) criteria from BABOK [30] and Project Management guides; (c) the formal writing also of the non-functional side of a story, which is far from obvious in a typical US card.

US² Title: ***Update User Profile***				
Id: 1.2	**Ver:** 1.1	**Priority:** M☐ S☐ C☐ W☐	**fsu:** ... **nfsu:** ...	**Effort (m/d):** ...
FUR	• User can update his/her own profile, included email, address, preferences and information about credit card(s)			
NFR	• Accessibility according to Section-508 standard • Browse with MS IE8			
US² Acceptance Tests				
1.	Successfully modify address, profession, hobbies, email, preferences, credit card data (positive)			
2.	Verify to insert another user's credit card data (negative)			

Fig. 3. US²-Type2, including both FURs and NFRs

There is also the possibility to have only NFR (including in this definition for sake of simplicity also the project-related tasks, as quality assurance, measurement, project planning and monitoring & control, etc.), in what we call a Type1 US² card (Fig. 4).

US² Title: ***Install Mozilla Firefox v32.x***				
Id: 1.2	**Ver:** 1.1	**Priority:** M☐ S☐ C☐ W☐	**fsu:** ... **nfsu:** ...	**Effort (m/d):** ...
FUR	• ---			
NFR	• Install Firefox on all PCs			
US² Acceptance Tests				
1.	Verify browser compatibility with previous installed software			
2.	...			

Fig. 4. US²-Type1, including only NFRs

After the customer and provider create the single US² cards, as achieved in an agile context, their analysis and evaluation can be done through applying a grid based on criteria such as independent, negotiable, valuable, estimable, small and testable (INVEST) Grid. The process is fully defined in [13] and uses Table 1 as the basic template to use between a customer and a provider. The six attributes have been described using a four-point ordinal scale (0–3), as in the ISO 14598-x standards, where '0' means 'poor/absent', '1' means 'fair', '2' means 'good' and '3' 'excellent'. Each cell contains a description that proposes a rating for that attribute at that level.

Table 1. The INVEST Grid [13]

INVEST	Description	0	1	2	3
		Poor /Absent	Fair	*Good*	***Excellent***
I – Independent	*User Stories should be as independent as possible*	The start of construction of a US is tied to the completion of at least one other US	The completion of a US hinders the start of construction of at least one other US	The US can contain any constraint, but its release can be constrained by the completion of at least one other US	The US is fully independent, and it can be realized and released with any constraint
N – Negotiable	*User Stories should be "open", reporting any relevant details as much as possible*	The US contains enough detail to be a technical specification (Design phase), leaving no room to negotiate any element	The US is written with enough detail to be a functional specification (Analysis phase), leaving no room to negotiate any element	The US is written with informative content defining a User Requirement in a consolidated manner, yet shared between Customer and Provider	The US is written with the informative content typical of a high-level need, allowing feedback between customer and provider
V – Valuable	*User Stories should provide value to end users in terms of the solution*	The functional part (F) of the US does not contain all the functionalities requested by the customer	The functional (F) part of the US expresses mostly qualitative (Q) and technical (T) requirements about the system, and needs to be more developed in terms of functional requirements	The functional (F) part of the US expresses mostly the functional requirements requested by the Customer, but also includes qualitative (Q) and technical (T) requirements	The functional (F) part of the US correctly expresses only the functional requirements requested by the customer
E – Estimable	*Each User Story must be able to be estimated in terms of relative size and effort*	The US shows only its functional (F) part, filled in by the customer, but without sufficient detail to allow the provider to fill in the Q/T parts	The US shows only its functional (F) part, filled in by the customer, but validated with the provider	The US has been completed by the provider with respect to Q/T issues, but still needs to be validated jointly with the customer	All the useful parts of the US (F/Q/T) are shown, allowing the effort need to size and estimate it, and validated by both parts
S – Small	*Each User Story should be sufficiently granular, and not defined at too high a level*	The US is very large, and cannot be completed within a Sprint	The US is very large, and can be completed within a Sprint, but cannot accommodate the creation/delivery of other US	The size of the US is such that it can be completed within a Sprint, jointly with other US, but it is too small to create overhead about the Testing phase	The size of the US is such that it can be completed within a Sprint, jointly with other US, ensuring an appropriate balance between development and testing activities
T – Testable	*Each User Story must be formulated in an effort to stress useful details for creating tests*	The US does not include tips about Acceptance Tests	The US includes a formal indication of Acceptance Tests, but yet to be completed	The US includes an indication of Acceptance Tests which are complete, but yet to be validated	The US includes an indication of completed and validated Acceptance Tests

In a service management context the main goal is to release 'value' to a customer. This is a summary of 'utility' (*fit for purpose*) and 'warranty' (*fit for use*), where the first one covers functional user requirements (FUR[3]) and the second one for non-functional requirements (NFR[4]) (see Fig. 5).

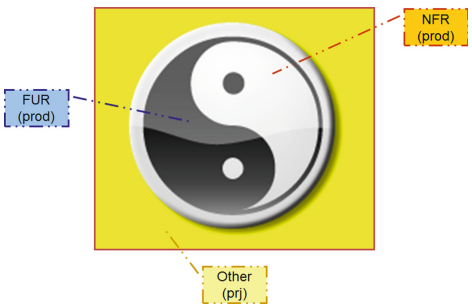

Fig. 5. Functional User Requirements (FUR) vs Non-Functional Requirements (NFR)

[3] A requirement that specifies a function that a system or system component must be able to perform (ISO/IEC/IEEE 24765:2010 [Systems and software Engineering Vocabulary]).

[4] A software requirement that describes not what the software will do but how the software will do it (ISO/IEC/IEEE 24765:2010 [Systems and software Engineering Vocabulary]).

There are two factors: (1) the ISO 25010 quality model for a software product can also be applied to a service for describing and managing the 'warranty'; (2) a service project could plan for each iteration ('sprint' in the Scrum glossary) to release firstly the 'core + enabling' services (in software this could be the 'development' part) and the following ones typically 'enhancing' services (in software this could be a series of enhancements).

3 Methodology

Our objective in this paper is to introduce a discussion and try to demonstrate that only one side of the story (software or service) may not deliver all the benefits an organization could achieve from a joint adoption. For instance, in two out of the three CMMI constellations, DEV and SVC, 16 out of 22 processes are the same [37, 38]: in this case would an organization run two separated process improvement initiatives or a single one by evaluating commonalities for a unique improvement plan? This corresponds to what BSI (British Standards Institution) called a 'publicly available specification' for an integrated management system ('PAS 99') [32]. For such an integration, our proposal is based on the **LEGO** (**L**iving **E**n**G**ineering pr**O**cess) approach [8] proposed for stimulating organizations to improve their own processes: it suggests to take pieces (as LEGO bricks) from multiple candidate information sources and integrate them to form a unique, reinforced picture for a particular process or set of processes. It allows organizations to avoid searching conformity to 'external' models, when a model itself is an abstraction for trying to catch several instances at a time[5]. Again, any model/framework can represent only a part of the observed reality, not all of its possible views, since it needs to represent a single viewpoint at a time. In that way, enlarging the scope of potential useful elements for improving the organizational BPM, there could be more chances for success. LEGO has four main elements (Fig. 6):

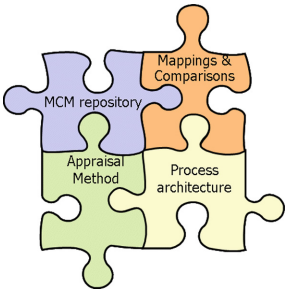

Fig. 6. The four elements of the LEGO approach

[5] Other related works are e.g. [27, 28, 33–35].

1. a 'Maturity & Capability Models' (MCM) repository (www.gqs.ufsc.br/mcm), from which relevant processes (i.e., MCMs) can be identified;
2. knowledge about the process architecture of each model, for understanding how to transform the desired elements from a certain model into the target format, especially when considering that the source models may have different architectures that need to be integrated into a single model;
3. mapping(s) & comparisons between relevant models, in order to understand the real differences or the deeper level of detail from 'model A' to import into 'model B';
4. a process appraisal method (PAM) to be applied on the target organization's BPM (Business Process Model).

The LEGO approach follows a four step process:

1. **Identify informative/business goals**: clearly identify your needs from the current BPM version and content.
2. **Query the MCM repository**: browse the MCM repository, setting up the proper filters in order to obtain the desired elements (processes; practices; etc.) to be inserted into the target BPM.
3. **Include the selected element(s) into the target BPM**: include the new element(s) in the proper position in the target BPM (e.g. process group, maturity level, etc.).
4. **Adapt & Adopt the selected element(s)**: according to the process architecture of both process models (the target and the source one), the selected elements may need to be adapted, tailoring such elements as needed.

Such an approach has been applied to several contexts and processes (e.g. requirement management [15], risk management [16], etc.), and it could be applied also to an improved BPM where some elements could have been missing.

4 Unification of Software + Service

This section presents an example on how to use jointly software and service models (what ISO calls a Process Reference Models (PRM)), picking up those information and best practices that one model eventually could not have yet foreseen, and to strengthen them. As already mentioned, an organization has to manage a project that could be often composed by pure software development and maintenance and pure services (e.g. incident management). Thus one of its goals will be how to lower the Total Cost of Ownership (TCO). Are really software and services so different or not? And if not, how keep the best from any model/framework/classification currently available from both communities? Can Knowledge Management (KM) be part of such solution? A real application should be done on your own BPM processes. Here, for sake of simplicity, ISO 15504-2 is selected as the target BPM to be reinforced and a series of Software and Service PRM as the sources to be investigated for picking up some interesting additional 'bricks' to be added, in case. Now a new application can be done considering the SPICE Knowledge Management process (RIN.3). KM was chosen because it is important to stress and discuss how to create and generate value for an organization. Having a proper KM process in place would help as a support process for many

initiatives, and it is not yet part of CMMI-DEV/SVC but of another SEI-based model (People CMM): thus it can be proposed as an example, together with CMMI, as a target model for the positioning of 'missing pieces'. Table 2 presents a list of KM-related models/frameworks explored for finding Elements of Interest (EoI) to be inserted for reinforcing RIN.3.

Table 2. Some KM-related models/frameworks

Model/framework	**Repr. Type**	**ML (#)**	**Architect-Type**	**Comments/notes**
APQC KMMM [17]	Staged	5 [1–5]	Level-based	•
Siemens KMMM [18]	Staged	5 [1–5]	Level-based	• 8 Key Areas
ONTOKNOM [19]	Staged	5 [1–5]	Level-based	• Ontology included
(G-KMMM [20]	Staged	5 [1–5]	Matrix -based	• Assessment with questionnaire by ML
InfoSys KMMM [25]	Staged	5 [1–5]	Level-based	•
KPMG Knowledge Journey [21, 22]	Staged	5 [1–5]	Level-based	• 4 KPAs
K3 M [22]	Staged	8 [1–8]	Level-based	•
KMCA [23]	Staged	6 [0–5]	Level-based	• Added a 'zero' ML
ITIL v3 Refresh 2011 [3]	—	—	—	• Svc Mgmt Framework, 5 SLC phases → KM in the Svc Transition (ST) phase; 7-Step Improvement Process in CSI (Continual Svc Improvement) phase
Microsoft MOF v4 [4]	—	—	—	• Svc Mgmt Framework, 4 SLC phases → KM in the 'Manage' phase
COBIT [24]	—	—	—	• IT Governance Framework → 4 main phases (PO, AI, DS, ME)

The following preconditions, process and main results from the application of the LEGO process to the KM domain are proposed for a better process that may be applied in an organization:

1. **Identify informative/business goals:** improve the capability of the organization to collect, share, reuse and improve its knowledge by its employees and partners.
2. **Query the MCM repository:** Table 3 proposes the list of potential elements of interest (EoI) to consider for improving ISO 15504 KM process.

Table 3. KM MCM: Elements of Interest (EoI)

Model/ Framework	Elements of Interest (EoI)
APQC KMMM	• —
Siemens KMMM	• 8 Key Areas (Planning, Ext Knowledge, People, Informal Rules, Operation, Int. Knowledge, Technology, Formal Rules)
ONTOKNOM	• KM Maturity Model Ontology based on three components (Admin, Author, User)
KPMG KJourney	• 4 KPAs (People, Process, Content, Technology)
G-KMMM	• 3 KPAs (People/Org, Process, Technology)
K3 M	• More refined levels for a gradual implementation
	• Top-down retention measurement at ML3 and a formal Org Knowledge Base (ML4)
KMCA	• Separating 'behavior' and 'infrastructure' into the analysis
ITIL v3 KM	• Overall, global concept of SKMS (Service Knowledge Management System)
	• The four waves for KM: DIKW (Data, Information, Knowledge, Wisdom)
	• Goal-oriented KM, well linked with the Measurement perspective and the CSI (Continual Service Improvement) process
Microsoft MOF v4	• 'Plan' phase, POL (Policy) area, Process 2 (Create Policy), activity #5 (Create KM policies)
	• 'Operate' phase, CUS (Customer Service) area, Process 3 (Resolve the Request), activities asking to search, locate, verify knowledge base articles
	• 'Manage' phase, GRC (Governance, Risk, Compliance) area, Process 2 (Assess, Monitor & Risk), Activity #9 (Learn from prior effects and update the Knowledge Base) → Stressed the 'learning' activity as a 'risky' element whether not properly managed
COBIT v4.1	• PO2.1 (Enterprise Architecture Model)
	• PO2.4 (Integrity Management)
	• AI4.2 (Knowledge Transfer to Business Management)
	• AI4.3 (Knowledge Transfer to End Users)
	• AI4.4 (Knowledge Transfer to Operation and Support Staff)

3. **Include the selected element(s) into the target BPM:** looking at the analysis of potential EoI (Elements of Interest) in Table 3. Table 4 shows how our suggestions were introduced in the current RIN.3 process, describing a new possible improved process that may be mapped against your own QMS internal process(es) covering that subject.
4. **Adapt & Adopt the selected element(s):** after adapting the original RIN.3 process considering the proposed suggestions for improvement (see Table 4), the improved RIN.3 process should be mapped now against the related QMS internal process

Table 4. KM process - suggestions for improvements.

ISO/IEC 15504 KM Process	Suggested Improvements
RIN.3	KM BPs
BP 01 – Establish a KM system	• Distinguish the 'behavior' from the 'infrastructure' [KMCA]
	• Define/Refine which Information Systems are part of the overall SKMS in Architectural terms [ITIL] [COBIT PO2.1]
	• Define – according to the 'four waves of KM' – the layers and related IS for gathering and distributing data, information, knowledge and wisdom [ITIL]
BP02 - Create the Network of Knowledge contributors	• Create and update a list of (primary, secondary) stakeholders to consider as the main input for formulating requirements and for checking their validity [COBIT PO2.4]
BP 03 – Develop a KM strategy	• The strategy should have clear KM axes of interest well defined from the beginning, to be periodically updated [ITIL SS, Siemens KMMM; KPMG KJourney; G-KMMM]
	• The specification of which KM areas could be the most relevant to the organization for a proper generation of value is welcome [Siemens KMMM, G-KMMM, KPMG]
	• Consider KM process and its implication also from a Risk perspective [MOF]
	• A KM Ontology could help during the creation/periodical update of the organizational overall strategy [ONTOKNOM]
	• The KM Strategy must be goal-oriented, receiving feedbacks from previous improvements put in action [ITIL CSI; MOF Plan]
BP04 - Capture Knowledge	• Revise periodically the potential sources of data/information gathering, also considering new technologies (e.g. Social media and the possibility to interface organization's website and intranet) [MOF Operate CUS; SECI model [14])
BP 05 – Disseminate Knowledge Assets (KAs)	• Keep in mind several stakeholders, not only customers but mostly Users and their perceptions in the creation of value [COBIT AI4.x][ITIL CSI]
BP06 - Improve KAs	• KAs must be managed as one of the several organization's Configuration Items (CI) to be updated on a regular basis [ITIL ST]
	• KAs must be updated as part of a regular CSI (Continual Service Improvement) program [ITIL CSI]

covering that subject. Since many organizations adopt an ISO management system (e.g. ISO 9001), a cross-check for validating potential improvements from the design phase could be achieved through re-applying the related mapping document to their own internal process (e.g. using the N/P/L/F – Not/Partially/Largely/Fully achieved ordinal scale from CMMI or ISO 15504). Moving from ISO 15504, it could be used also the Mutafeljia & Stromberg's mapping [26] and/or the one by Peldzius and Ragaisis taking CMMI-DEV and ISO 15504 [29] as a basis. In this paper, our focus was limited to only the design phase. However, a case study with the application of hybrid-RIN processes will be included in a future paper.

The EoI presented in Table 3, as well as the included elements, respect the BPs of the RIN.3 process provided in Table 4 are not to be considered exhaustive: to the contrary, these two tables are to be considered as a starting point for the application of the LEGO approach in practice.

What can be easily observed reading the 'EoI' column is that any model can propose a series of elements and good practices, but just a single 'model' cannot include in one possible viewpoint every possible EoI, simply because they all were originated from different assumptions and rationales.

In particular this short example from ISO 15504 RIN.3 process stressed the need:

- to reinforce the list of work products/deliverables defined at the end of the ISO PRM with few more elements, not currently defined;
- to provide suggestions about the communication area, because – as in the SECI[6] model [14]– you can also have a great idea but being limited to few applications, while the larger the diffusion, the higher the probability to create/generate new joint ideas from the initial one, being refined little by little after an initial application;
- a list of stakeholders to be periodically contacted (e.g. panels) for providing opinions/ideas on new products-services or revision for current services provided to the market.

5 Conclusions and Next Steps

Software and Service are two sides of the same coin within an ICT project. Too often they are viewed as separated issues to be managed and improved by specific models and frameworks. After reviewing the main differences and commonalities, it could be valuable to an organization to start looking at them as friends and not as foes. A list of common items valid both from the software and service sides has been discussed (e.g. the way to manage requirements by User Stories), depicting the main challenges to properly manage them together as a whole.

The LEGO (Living EnGineering prOcess) approach has been presented as an effective way to take into account several information sources from the MCM (Maturity & Capability Models) belonging to the desired area/domain to be improved. LEGO has been applied in different ways over the past years to specific process areas (PA) to be

[6] SECI (Socialization – Externalization – Combination – Internalization).

improved as a 'vertical' improvement, while in this paper it was applied in a 'horizontal' way, trying to give continuity to a continuous flow (from the development of a software system till its maintenance) within the unique ICT project frame. The RIN.3 Knowledge Management process from the ISO 15504 PRM (Process Reference Model) has been considered as a small application example, considering models from both domains (software; service & governance) for picking up potential Elements of Interest (EoI) to be suggested for strengthening RIN.3.

An organization needs more and more to 'pick up' pieces from several frameworks and models in order to reinforce its own unique Business Process Model (BPM), while too often organizations search for compliance to 'external' models (e.g. one or more CMMI constellations) thinking such models could be the target instead of being simply suggestions for an internal improvement. But a model is and remains simply a model. Each model can have its way to look at a phenomenon but cannot capture all the potential interpretations and 'nuances' of a certain process/domain. Therefore the need to know more sources of information and try to summarize them in the best possible way but respecting the organization's BPM process architecture, that is the real target to improve. The papers about LEGO applications (e.g. [8, 15, 16, 39]) can be a starting point to learn and try to replicate the approach on 'your' own BPM and processes. The most challenging item can be how to filter the EoI (Elements of Interest) useful for being incorporated into your own BMP (target). This is why the issue dealt with in this paper was Knowledge Management and the way organizations typically deal with that process (or not). Too often such process seems to be too implicit in many medium-large organizations and could be confused with solely training.

LEGO represents a different way to improve processes from multi-source models than done in EnterpriseSPICE [27] or FAA iCMM [28] or other approaches (e.g. [33–35]) since LEGO stresses a dynamic perspective about how to find room for improvement in your own BPM, rather than considering a meta-model.

Next steps will be about the analysis of other points of contacts between the software and the service side of ICT projects, such as the measurability issue, where the knowledge coming from the software community could bring some useful tips for reinforcing the Service Level Management (SLM) process as well as how paradigms, such as DevOps [36], can help improving better and faster software and services by a more focused collaboration and communication about stakeholders. Again, a mapping of crossed terms and the way there are differently mentioned in the respective communities (e.g. a 'Change Request' is on software side the same concept and working item that a 'Request for Change' in the service side) will be created in order to facilitate such logical merging.

References

1. CMMI Product Team, CMMI for Development, Version 1.3, CMMI-DEV v1.3, Continuous Representation, CMU/SEI-2010-TR-033, Technical report, Software Engineering Institute, November 2010

2. ISO/IEC, IS 15504-2: 2003, Information technology – Process assessment – Part 2: Performing an assessment, October 2003
3. ITIL v3 Refresh 2011 suite, AXELOS (2011). http://goo.gl/Ets5Xb
4. Microsoft, Microsoft Operation Framework (MOF) v4.0 (2012). http://goo.gl/BvGg3i
5. CMMI Product Team, CMMI for Service, Version 1.3, CMMI-SVC v1.3, CMU/SEI-2010-TR-034, Technical report, Software Engineering Institute, November 2010
6. TM Forum, Business Process Framework (eTOM), v14.5 (2015). http://goo.gl/vTXjkh
7. ISO/IEC IS 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, Geneve (2011)
8. Buglione, L., Gresse von Wangenheim, C., Hauck, J.C.R., McCaffery, F.: The LEGO maturity & capability model approach. In: Proceedings of the 5th World Congress on Software Quality, Shanghai (China), October 2011
9. ISO/IEC IS 20000-1:2011, Information technology – Service management – Part 1: Service management system requirements, Geneve (2011)
10. ISO, IS 9001:2008, Quality management systems – Requirements, Geneve (2008)
11. Schwaber, K.: Agile Project Management with Scrum, Microsoft Press (2004). ISBN 978-0735619937
12. Buglione L.: Agile-4-FSM. Improving estimates by a 4-pieces puzzle, Webinar, IFPUG Agile Interest Group, 17 May 2012. http://goo.gl/wtXWt
13. Buglione, L., Abran, A.: Improving the user story agile technique using the INVEST Criteria, IWSM-MENSURA 2013. In: 23th International Workshop on Software Measurement and 8th International Conference on Software Process and Product Measurement. IEEE/CS Proceedings, Ankara (Turkey), 23–26 October 2013, pp. 49–53 (2013)
14. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies create the Dynamics of Innovation, OUP USA (1995). ISBN 978-0195092691
15. Buglione, L., Hauck, J.C.R., Gresse von Wangenheim, C., Mc Caffery, F.: Hybriding CMMI and requirement engineering maturity & capability models: Applying the LEGO approach for improving estimates. In: ICSOFT 2012, Proceedings of the 7th International Conference on Software Paradigm Trends, Rome (Italy), 24–27 July 2012
16. Buglione, L., Lami, G., von Wangenheim, C.G., Caffery, F.M., Hauck, J.C.R.: Leveraging reuse-related maturity issues for achieving higher maturity and capability levels. In: Favaro, J., Morisio, M. (eds.) ICSR 2013. LNCS, vol. 7925, pp. 343–355. Springer, Heidelberg (2013)
17. Hubert, C., Lemons, D.: A Knowledge Management Maturity Model – APQC's Stages of Implementation (2009)
18. Langen, M.: Holistic development of KM with the KM maturity model (KMMM). In: APQC Conference, 7–8 Dec 2000
19. Hefke, M., Kleiner, F.: An ontology-based software infrastructure for retaining theoretical Knowledge Management Maturity Models. In: 1st Workshop on Formal Ontologies Meet Industry, FOMI 2005, Verona, Italy (2005)
20. Pee1, L.G., Teah, H.Y., Kankanhalli, A.: Development of a general knowledge management maturity model. In: Proceedings of the 10th Pacific Asia Conference on Information Systems (PACIS), 6–9 July, pp. 401–416. Kuala Lumpur, Malaysia
21. KPMG, Knowledge Management Assessment Exercise (1999). http://goo.gl/3c9mOR
22. WisdomSource, Knowledge Management Maturity (K3 M). WisdomSource News 2(1), 31, May 2004. http://goo.gl/WbaW7b
23. Freeze, R., Kulkami, U.: Knowledge management capability assessment: validating a knowledge assets measurement instrument. In: HICSS 2005 Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences, vol. 08, p. 251.1 (2005)

24. ISACA, COBIT v4.1 (2007). http://goo.gl/lOaHZv
25. Kochikar, V.P.: The knowledge management maturity model: a staged framework for leveraging knowledge. In: KM World 2000, Santa Clara, CA (2000). http://goo.gl/zl7tBV
26. Mutafeljia, B., Stromberg, H.: Process Improvement with CMMI v1.2 and ISO Standards, Auerback Publications (2008). http://goo.gl/BFUqq
27. ISO JTC1/SC7/WG10 Study Group, EnterpriseSPICE - An Integrated Model for Enterprise-wide Assessment and Improvement Technical report Issue 1 - September (2010). http://enterprisespice.com/
28. Ibrahim, L., Bradford, B., Cole, D., LaBruyere, L., Leinneweber, H., Piszczech, D., Reed, N., Rymond, M., Smith, D., Virga, M., Wells, C.: The Federal Aviation Administration Integrated Capability Maturity Model-, (FAA-iCMM), Version 2.0. An Integrated Capability Maturity Model for Enterprise-wide Improvement, FAA, September 2001
29. Peldzius, S., Ragaisis, S.: Investigation correspondence between CMMI-DEV and ISO/IEC 15504. Int. J. Educ. Inf. Technol. 5(4), 361–368 (2011). http://goo.gl/Dqupq9
30. IIBA, A Guide to the Business Analysis Body of Knowledge (BABOK) v3, International Institute of Business Analysis (2015)
31. Pipkin, J., Lunsford, G.H.: Synergism of the CMMI development and services constellations in an hybrid organization. In: CMMI Conference North America, May 2014. http://goo.gl/s7s9Is
32. BSI, PAS 99:2012 – Specification of common management system requirements as a framework for integration – Publicly Available Specification (2012). https://goo.gl/zWh5OZ (working draft)
33. SEI, PrIME project, Process Improvement in Multiple Envinronment. http://goo.gl/AK79wr
34. Jeners, S., Lichter, H., Dragomir, A.: Towards an integration of multiple process improvement reference models based on automated concept extraction. In: Winkler, D., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2012. CCIS, vol. 301, pp. 205–216. Springer, Heidelberg (2012)
35. Andelfinger, U., Heijstek, A., Kirwan, P.: A Unified Process Improvement Approach for Multi-Model Improvement Environments, News @ SEI, 1 Apr 2006. http://goo.gl/ztuKl2
36. Andelfinger, U., Heijstek, A., Kirwan, P.: DevOps, Wikipedia. https://goo.gl/20PrLX
37. Stall, A., Forrester, E.: Using CMMI-DEV and CMMI-SVC Together – Where 'Build Stuff' Happens in CMMI-SVC, 2012 SEPG NA, Presentation, March 2012. http://goo.gl/z0PIuT
38. Gonzales, R.M.: CMMI®-DEV versus CMMI®-SVC analysis. In: 11th Annual CMMI Technology Conference and User Group, Denver (USA), 15 Nov 2011. http://goo.gl/dhldcC
39. Buglione, L., Gresse von Wangenheim, C., Mc Caffery, F., Hauck, J.C.R.: The LEGO strategy: guidelines for a profitable deployment. Comput. Standard Interfaces **36**(1), 10–20 (2013). Elsevier