Methodological Support for the Design of Enterprise Information Systems with SDBS: Towards Distributed, Service-Oriented and Context-Aware Solutions

Boris Shishkov

IICREST, 53 Iv. Susanin Str., Sofia, Bulgaria
b.b.shishkov@iicrest.eu

Abstract. Taking in consideration the inability of traditional software development methods to meet to a full extent some new demands, such as ones related to service-orientation and context-awareness, we suggest in this paper directions to enrich software development methods, for (partially) resolving this drawback. We introduce and briefly discuss the SDBC approach – 'SDBC – Software Derived from Business Components', mentioning as well actual related work that is concerned with the alignment between business process modeling and software specification. We then discuss two of the most important challenges of current software development, namely adaptability (envisioning distribution and context-awareness as desired properties) and service-orientation. Based on this, we derive some actual solution directions that concern software development. We consider this as a contribution that relates to the further software development trends.

Keywords. SDBC, Components, Service-Orientation, Context-Awareness.

1 Introduction

Information and Communication Technology (ICT) has substantially influenced almost all spheres of human activity (including enterprise development), during the previous century, bringing on the stage information, as main asset, which in turn led to the problems of collecting, storing, processing, and communicating (enterprise-related) information [1]. Hence, Enterprise Information Systems (EIS) consist not only of software applications and software infrastructures to run them but also of people, hardware units, and so on. The EIS notion gets broader and more complex with the development of enterprises and the advances in enterprise technology. It is widely agreed nevertheless that ensuring the development and operation of enterprise software applications with predictable and improved cost, schedule, and quality, is of crucial importance for current EIS [2]. This inspires us to focus in particular on the development of ICT applications that are to support enterprises. Nevertheless, this was not a big challenge well until the late 1980s, in our view, not only because enterprise processes were not so complex (for example distributed and adaptable) as they currently are but also because the assumed support that ICT applications would deliver was mostly limited to the delivery of some calculation upon request. This changed rapidly in the following decade when more and more software development

Shishkov B.
Methodological Support for the Design of Enterprise Information Systems with SDBS: Towards Distributed, Service-Oriented and Context-Aware Solutions.
DOI: 10.5220/0004465300360050
In *Proceedings of the 4th International Workshop on Enterprise Systems and Technology (I-WEST 2010)*, pages 36-50
ISBN: 978-989-8425-44-7

tasks were dominated by the goal of (partially) automating human actions. This pushed, we believe, software engineering in a new 'dimension' where software systems were supposed to play complex role in the context of an enterprise. Hence, the alignment between business process modeling and software specification was becoming crucial.

This mentioned alignment used to be a weak point in most of the software development methods during the 1990s and even beyond [3]. What's more, the software community used to consider the challenge of closing this gap as having a breakthrough importance [4]. Reporting actual (partial) research results in this direction, we relate in this paper the mentioned challenge to further and currently actual ones, such as adaptability and service-orientation, in order to establish a perspective on how ICT applications should be built in order to meet current demands, especially as far as EIS are concerned.

In particular, we introduce and briefly discuss one software development approach that reflects the traditional software development best practices, namely the SDBC approach – 'SDBC – Software Derived from Business Components' [5], mentioning as well actual related work that is concerned with the alignment between business process modeling and software specification. We then discuss two of the most important challenges of current software development, namely adaptability (envisioning distribution and context-awareness as desired properties) and service-orientation, concluding about the importance of enriching the well-established software development tools with adaptability and service –related features. We have a two-fold problem here, nevertheless: (i) The well-established software development methods are mostly envisioning the design of an ICT application which even though built in a component-based fashion, is self-contained – services performed by unknown components through the 'Cloud', being adapted dynamically to a need, are not considered. (ii) The current service-oriented and context-driven approaches are often too abstract with regard to the actual service realization that is to be anchored in particular ICT applications and some corresponding components.

Thus, the contribution of this paper relates to a proposed reinforcement of some well-established software development best practices through actual enrichments inspired by adaptability and service –related desired features.

The outline of the remaining of this paper is as follows: Section 2 introduces the SDBC approach and discusses other representative traditional software development approaches, for the sake of getting insight on some of the well-established software development best practices. Section 3 outlines two of the currently actual EIS-related challenges, namely distribution and adaptability, in order to inspire the further discussion on enriching traditional software development methods. Then Section 4 presents some proposed solution directions with regard to traditional software development methods and their desired enrichment. Section 5 exemplifies partially some of the presented views and proposed solution directions. Section 6 briefly outlines related work. Finally, Section 7 presents our conclusions.

2 SDBC and Traditional Software Development Methods

In this section we outline briefly the SDBC approach as an approach that is based on the traditional software development principles and we discuss subsequently some other traditional approaches.

SDBC. In summarizing the approach *Software Derived from Business Components – SDBC* [3], we use the following abbreviations as applied in Figure 1: bc – *Business Component* (a business sub-system that comprises exactly one business process); bk – *Business CoMponent* (a model of a Business Component, which model is adequately elaborated in terms of statics and dynamics); glbk – *general Business CoMponent* (which is re-usable by extension); gcbk – *generic Business CoMponent* (which is re-usable by parameterization); ssm – *software specification model*; sc – *software Component* (an implemented piece of software representing a part of an application); sk – *Software CoMponent* (a conceptual specification model of a Software Component).

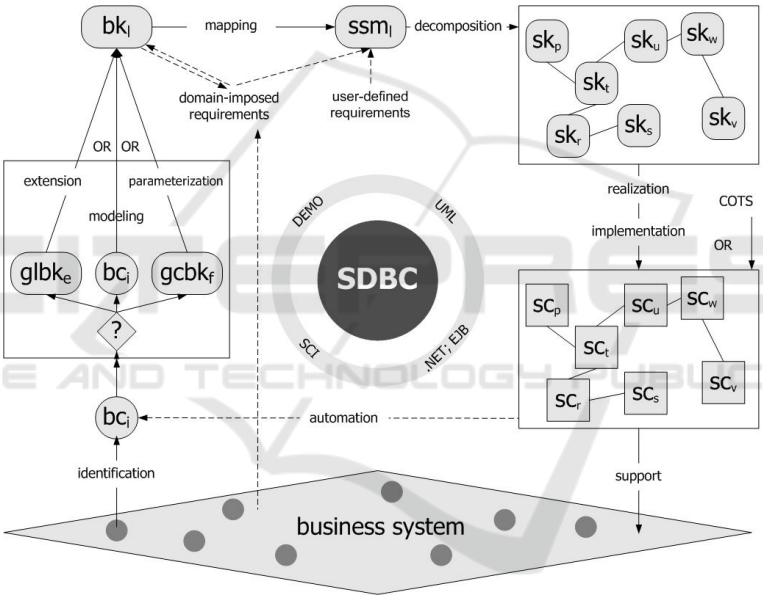

Fig. 1. Outline of the SDBC approach [5].

The Figure shows that SDBC is about a component-based business-process-modeling-driven specification and realization of software. The starting point is the consideration of a business system. Business Components are identified from it. This can be done through the *SCI* technique – *Structuring Customers' Information* [5]. The Business Components should then be reflected in corresponding Business CoMponents, in supplying an adequate modeling foundation for the further software specification activities. Another way of arriving at a Business CoMponent is by applying re-use: either extending a general Business CoMponent or parameterizing a generic Business CoMponent. DEMO and other Language-Action-Perspective -driven

modeling tools [6] are relevant as far as Business CoMponents' specification is concerned. Each Business CoMponent should be then elaborated with the domain-imposed requirements, for the purpose of adding elicitation on the particular context in which its corresponding Business Component exists within the business system. Then, a mapping towards a software specification model should take place, possibly driven by the DEMO-UML transformation mechanism [5]. The domain-imposed requirements as well as the user-defined requirements are to be considered here, since the derived software model should reflect not only the original business features but also the particular user demands towards the system-to-be. The (UML-based) software specification model would need then a precise elaboration so that it provides sufficient elicitation in terms of structure, dynamics, data, and coordination [3]. The model needs also to be decomposed into a number of Software CoMponents reflecting functionality pieces. Then these Software CoMponents are to undergo realization and implementation, being reflected (in this way) in Software Components. This final set of components might consist of such components which are implemented (using software component technologies, such as .NET or EJB, for instance) based on corresponding Software CoMponents and such components which are purchased. Finally, the (resulting) component-based application would support the target business system, by automating anything that concerns the initially identified Business Component(s) identified from the mentioned system.

Other Traditional Methods. The challenge of capturing the essential aspects about business processes for the purpose of further software specification, has been addressed not only by the SDBC approach but also by methods such as Catalysis [7] and Tropos [8] as well as by the Model Driven Architecture – MDA [9].

The Catalysis method provides a coherent set of techniques for business analysis and system development as well as well-defined consistency rules across models. However, these techniques concern the software design perspective and have no theoretical roots relevant to the modeling of business processes. Hence, the business process modeling as conducted in Catalysis would inevitably be superficial and therefore the method cannot guarantee an adequate capturing of all related real-life aspects, including semantic and pragmatic ones. In addition to this, Catalysis does not have mechanisms for a mapping between business process models and software specification models. Therefore, a definite strength (in this regard) of SDBC is that, relying on the LAP-OS 'combination', the approach supports adequately the business process modeling task and the software design activities in SDBC stem from a pure business process model, guaranteeing that the application-to-be would function adequately in the business environment in which it would have to be integrated.

The strengths of the method Tropos relate to its capability of conducting a sound requirements analysis, considering the business processes which are to be supported by the application-to-be. From such a business process modeling point of view, the method addresses the software design. The mentioned requirements analysis includes elicitation not only of the 'early requirements' that concern the original business reality but also of the 'late requirements' which are about a corresponding updated (desired) business reality. The analysis is driven by a thorough consideration of the intentions of stakeholders, modeled as goals which are then reflected in the system's global architecture. Its definition is in terms of sub-systems interconnected through data, control, and other dependencies. Then a detailed design follows. Therefore, all

this features Tropos as a powerful method for designing software, which appropriately refers to the task of capturing essential real-life aspects that concern the modeling of business processes. However, the method is incomplete with regard to some of these aspects – it is not exhaustive in handling semantics and is insufficiently concerned with essential pragmatic issues, such as communicative actions, negotiations, coordination, and so on [5]. Further, the method lacks (just like Catalysis) clear and complete guidelines (and elaboration) on how to reflect the business process modeling output in the specification of the application-to-be. Such a specification would therefore inadequately reflect the original business model.

MDA prescribes three viewpoints from which models of the application-to-be (our target software system) should be defined: Computational Independent Models (CIMs) should focus on the environment and requirements of the system, abstracting from the system's construction; Platform-Independent Models (PIMs) should focus on the functionality of the system without revealing details on the specific technological platform on which the system is built, and Platform-Specific Models (PSMs) should define how a PIM is built using some specific platform. Therefore, the Computational Independent Modeling as well as the CIM-PIM transformation relate to the problem addressed in the current paper, namely the achievement of an adequate business-software alignment which is concerned with all relevant real-life aspects. However, bridging business process models and application design by using Computational Independent Modeling and realizing a CIM-PIM mapping, are issues not enough explored, as it is well known. The MDA Community still misses sound guidelines and procedures on how to discover Computational Independent Models and how to reflect them in Platform Independent Models.

What we conclude is that traditional software development approaches fundamentally address the link between the construction of ICT components (done by developers) and the delivered functionality (mainly consumed by users). Hence, traditional software development is not considering the possibility to directly 'consume' services from the 'Cloud', adjusting this in an ad-hoc fashion, upon the appearance of a necessity.

3 Towards More Distributed and Adaptable Solutions

In this section we outline, as mentioned in the Introduction, two of the currently actual EIS-related challenges, namely distribution and adaptability, in order to inspire the further discussion on enriching traditional software development methods.

Distributed Web Services. We start discussing the actual (EIS-related) challenges (which challenges are in our view insufficiently reflected in traditional software development approaches), by firstly addressing the service concept not only because service-orientation usually demands a heavy distribution (that is not fully in line with most software development approaches) but also because the current significant changes in software technology are centered around this concept [10].

From an abstract point of view, a service represents a piece of well-defined functionality that is available at some network endpoint and is accessible via various transport protocols and specialization formats. The functionalities provided by

services cover a vast spectrum reaching from low level features like offering storage capabilities, over simple application functions like changing a customer address, to complex business processes like hiring a new employee.

To usefully utilize the service perspective in developing ICT applications would mean according to Shishkov & Van Sinderen [11] the ability to create new applications from existing services, independently of who provides these services, where they are provided, and how they are implemented. It should be nevertheless mentioned that by 'creating' an application, it is not meant constructing the application from the scratch; what is 'created' is the end result that represents a functionality consumed by users. This is the significant change in our vision concerning software technology – developers would no longer possess full control over all software components that realize services. What is more, the development task may split:

- some developers would just focus on the development of small software modules delivering generic adjustable services to whoever might be interested in using them;
- other developers would not develop software components any longer, focusing instead on the composition of complex functionalities (for this, they would be using available generic services).

According to some recent views of Frank Leymann [12], a new kind of middleware is currently evolving for the partial support directed to the dealing with services in such a way, with the idea that based on the specification of the functionality needed, the middleware determines (automatically or through a developer's intervention) a (composite) service that would deliver the required functionality. This is not only affecting the application functionality creation but it would concern performance, taking into account that services would require in most cases processing power of back-end server systems. Thus both application creation and run time processing would substantially rely on support from the 'Cloud' and this should certainly point in particular to web services that represent services which are created and executed through the Web. Hence, in order to be of actual use, such services would demand enabling technology standards and the web service technology stack as according to Papazoglou [10] outlines some actual web service technologies and standards – Figure 2.

As it is seen from the Figure, web services' relying on a transportation protocol is crucial. Although not tied to any specific transport protocol, web services build on ubiquitous Internet connectivity and infrastructure to ensure nearly universal reach and support. Hence, their mostly relying on HTTP (the connection protocol used by web servers and browsers) and XML (a widely accepted format for all exchanging data and its corresponding semantics) looks logical.

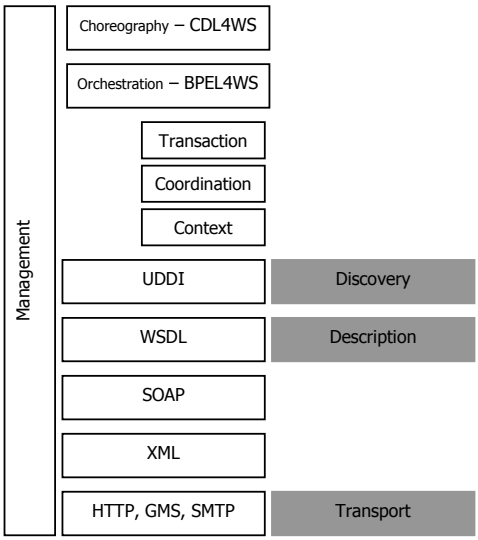

Fig. 2. Web services technology stack [10].

Having this as a 'foundation', we are to mention further the core web service standards, namely SOAP, WSDL, and UDDI:

- SOAP (Simple Object Access Protocol) is a simple XML-based messaging protocol on which web services rely in exchanging among themselves information. SOAP implements a request/response model for communication between interacting web services.
- WSDL (Web Service Description Language) is a language that specifies the interface of a web service, providing to the requestors a description of the service in this way.
- UDDI (Universal Description, Discovery, and Integration) represents a public directory that not only provides the publication of online services but also facilitates their eventual discovery.

When we then have to compose web services, we need to introduce some orchestration, defining their control flows [13], such as sequential, parallel, conditional, and so on, with this ending up with the determination of complex processes that may span many parties. BPEL4WS (Business Process Execution Language for Web Services) can support usefully such composition activities.

As the collaboration among many parties (through their web services) is concerned, a common observable behavior (choreography) would often need to be defined. CDL4WS (Choreography Description Language for Web Services) can usefully support such collaboration descriptions.

Adaptable Systems with Context-Aware Behavior. The utilization of a generic service for a specific user-related situation logically relates to acquiring knowledge on the context of the user and also exploiting this knowledge to provide the best possible service, which is labeled as context-awareness by Shishkov & Van Sinderen [14].

We hence claim that taking the end-user context into account is important in adequately delivering a service. Examples of end-user context are the location of the user, the user's activity, the availability of the user, and so on. We do assume that the end-user is in different contexts over time, and as a consequence (s)he has changing preferences or needs with respect to services.

Usefully enriching in this perspective traditional application development that may be SDBC-driven for example, means that the application under consideration or the component under consideration need some 'sensitivity' with regard to the changes in the end-user's context.

A schematic set-up for a context-aware application is depicted in Figure 3. Here, the application is informed by sensors of the context (or of context changes), where the sensing is done as unobtrusively (and invisibly) for the end-user as possible. Sensors sample the user's environment and produce (primitive) context information, which is an approximation of the actual context, suitable for computer interpretation and processing. Higher level context information may be derived through inference and aggregation (using input from multiple sensors) before it is presented to applications which in turn can decide on the current context of the end-user and the corresponding service(s) that must be offered.

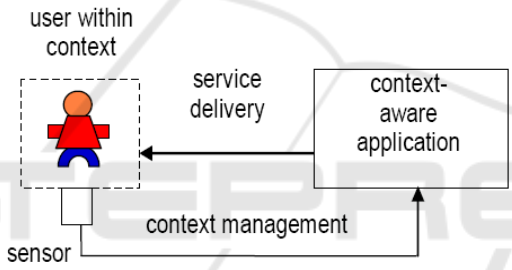

Fig. 3. Schematic representation of a context-aware application [14].

Based on previous research, the authors claim that the establishment of what desirable behavior corresponds to a (captured) change in end-user's context, is a challenge of great importance. For this reason, we would like to especially discuss this challenge in the current paper.

This challenge points to several interrelated sub-challenges, discussed briefly below:

- The Capturing: it is not trivial to establish through sensors what change in end-user's situation has actually occurred not only because the raw sensor data is very often at high risk of being misinterpreted but also because establishing adequate Quality-of-Context levels that are reliable is rarely realistic in our view.
- The Delivery: although a deterministic approach for defining the delivered behaviors, when it is precisely known still in advance what is the service that the system should deliver in one or another particular end-user context situation, appears to be a reliable solution, it is questionable how realistic is it to adequately foresee all possible end-user context situations at design time; as for non-deterministic solutions, we doubt how much reliable they would be taking into account the observed by us failure (in general) in much artificial-intelligence -related projects.

- The Response Capacity: even though it should be in most of the cases possible to predict most of the situations the end-user may appear to be in, there would always be such situations that are completely out of expectation and if they would demand service(s) that in turn need(s) unavailable resources, it would be impossible to deliver timely the proper service(s) to the end-user.

What we conclude is that traditional software development approaches can have chances to stay adequate if they allow for a paradigm shift according to which:

- Components and not applications are developed, with no restriction on how and with what other components the component under development will interact;
- The service delivered by a component is separated conceptually from the implementation of the component, keeping at the same time correspondence and traceability that would allow for different interactions of the service delivered by the component with other services, with these interactions realizable just through the service descriptions;
- A component is to allow for delivering different versions of its service putting this in dependence from the end-user situation.

A schematic representation of this vision is exhibited in Figure 4:

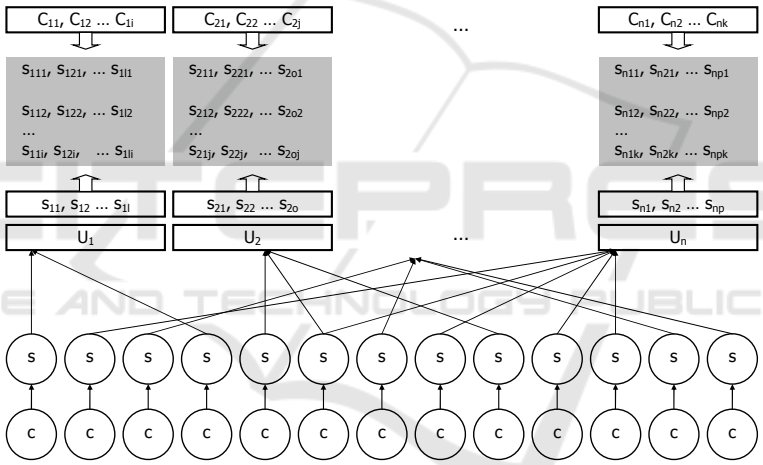

Fig.4. Actual view on the software development challenge.

As the Figure suggests, software components (C) appear to be the units to be developed, intended to deliver functionality pieces to a broad non pre-defined public. The manifestation of these functionality pieces are the services (S). It is to be noted that the corresponding underlying components remain 'hidden' from the broad public. Being utilizable in unlimited ways, a service may appear in tens and hundreds of utilization schemes (U), in each of which schemes it is combined in different ways with other different services. Hence, for a particular utilization scheme, there should be services derived from the service level (S) which are appropriately parameterized, composed and combined. The aggregate of this represents the desired functionality with regard to a utilization scheme. What should also be taken into account however is the context corresponding to each of the utilization schemes – this would point to a

number of different possible context states (C_i); for each of these context states, the service output should appear in different versions. Therefore, each matrix (colored in grey in the Figure) corresponds to the service support that relates to a particular utilization scheme – there are different versions of combinations of services for each possible context state.

The visions that were presented in this section inspire us to propose some solution directions with regard to traditional software development methods and their desired enrichment – this will be done in Section 4.

4 Solution Directions

As already mentioned, we will propose some solution directions with regard to traditional software development methods and their desired enrichment. We will do this by firstly further analyzing several actual demands towards the development of software and secondly – proposing some partial solution directions on this basis.

Analysis. Inspired by the demands identified and also discussed in the previous section, which can be summarized as: (i) floating components; (ii) utilizable services; (iii) sensitivity to end-user's context, we add as well the following issues that are to be taken into consideration:

- the *run-time control* will be slipping away from the hands of the user since different services are delivered by components which are implemented/run on remote servers where the user has limited or no control;
- the inevitable *infrastructure dependency* regarding service utilization will push much more attention to the service environments development, which together with the development of generic components, will be comprising the main part of the software development efforts.

These changes put in our view less technical and more business oriented people as the actual producers of the solutions that are finally delivered to the user; it becomes more necessary to be able to properly combine functionalities in the context of a business process and different requirements rather than being able to implement components that deliver functionality pieces (such tasks are already becoming trivial in many cases).

Proposal for Reinforcing Traditional Software Development Practices. Considering (i) the traditional application development ways, and particularly the perspective of the SDBC approach, (ii) the current needs for distribution, service-orientation, and context-awareness, and also the conclusions (derived) on their applicability within software development, (iii) the brief analysis (see the first part of the current section) on the impact of the change in the way we look upon software development, we propose a software development vision (outlined in Figure 5) that we expect would better fit the upcoming demands:

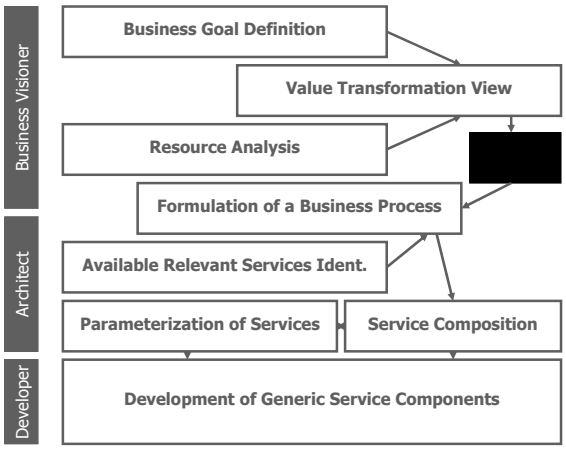

Fig. 5. Vision on some actual software development tasks and roles.

As it can be seen from the Figure, the role of a Business Visioner is becoming more important and far reaching - Business Visioners are defining a business goal and this goes often irregardless of the technology to be used, it is just an abstract goal statement. There is another activity that in our opinion is to go in parallel with the mentioned one: resource analysis (it is impossible to realize even a very small business goal without resources - resources are to be analyzed in order to justify the adequacy of the business goal). What is nevertheless of significant importance in our view is the definition of the desired value transformations: how we transform our available resources in order to achieve the business goal. This gives already a rough view on the business process to be addressed but how to move from the value transformation view to a specification of a business process, is a challenge and we still have not seen a definitive transformation vision. It is to be mentioned that technology is applied not for the sake of applying technology but as a way to improve effectiveness, efficiency, and quality. We need a business-level motivation or justification that a web service is a better 'fit' in a particular business model than a human service, for example. For this reason, the decision to use web service(s) has business-level roots. Services should be identified for knowing how they can be used with regard to the business process. For this reason, the Business Process formulation relies not only on Business Visioners' value transformation view but also on Architect's service-related view. Moreover, after there is a business process that is (partially) implementable through web services that are basically known as type, the particular service selection may start (what is the best service solution or the cheapest, and so on). Once the services are selected for being used, they would need to be parameterized and possibly composed (service composition is a topic that will not be discussed in this paper). These all are responsibilities of the Architect. Finally, the Developer must have implemented the components which realize the actual services.

Considering distributed systems and typical distributed architectures and inspired by previous research, we propose a layered design architecture illustrated in Fig. 6:

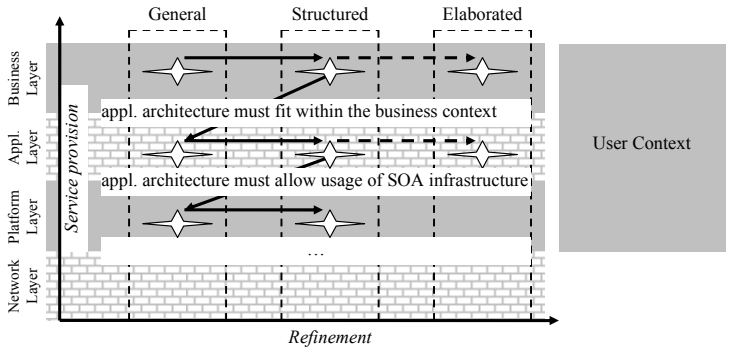

Fig. 6. Proposed layered architecture.

As it can be seen from the Figure, we distinguish 4 service provisioning layers, namely Network Layer (concerned with networking protocols), Platform Layer (concerned with (IT) infrastructures), Application Layer (concerned with the application logic), and Business Layer (concerned with the business logic that is not delegated to the application layer).

Further, we consider 3 'degrees' of refinement, namely General ('black-box') view, Structured (high-level 'white-box') view, and Elaborated (detailed 'white-box') view.

Next to that, we enforce two relevant desired properties (requirements):

- The service(s) provided by the Application Layer must fit within the business context;
- The architecture of the Application Layer must be SOA compliant.

Furthermore, we consider end-user's context as crosscutting at least with regard to the Platform, Application, and Business layers since in different context situations, different business steps and solutions are to be considered which leads in turn to different services and they in turn are projected on platforms. Nevertheless, for the sake of brevity, we will not discuss in more detail the proposed architecture.

In the following Section, we partially illustrate our presented views.

5 Illustrating Example

To illustrate partially our views, we consider the 'derivation' of an Education Mediator (EM) that would support customers in a number of ways, in an e-learning context. By 'customers', we mean the users of EM's services; those could be students and teachers (in the simplest case). Furthermore, we address (for the sake of brevity) only EM's advice provisioning service: *a customer can receive from EM advice which of the Student/Teacher entities (registered in the system) best satisfy a need* (for example, which is the best teacher with respect to a particular student demand). To receive advice from EM, the customer approaches EM's *ADVISOR* (an entity that is internal with regard to EM, which is responsible for handling the advice provisioning). It should be nevertheless noted that the Advisor may be shielded from the customer by the EM and in such a case the customer would be 'talking' to the EM

and the EM would in turn route requests to (and results from) the Advisor. Approaching the Advisor, the customer should specify a request: course type (e.g. lecturing course or experimental course), preferences (e.g. closest to a particular subject), and so on. Based on this (and acting 'through' the Match-maker, to be introduced further on in this paragraph), EM's *REQUEST HANDLER* (an entity that is internal with regard to EM also; this entity processes requests) generates a standardized request specification, appropriately synthesizing some of the information provided by the customer. This is delivered then to EM' *MATCH-MAKER* (an entity that is also internal with regard to EM; this entity is responsible for finding a match using the standardized request and considering what is currently available); the Match-maker realizes matches driven by particular criteria, chosen by the customer (and represented in the standardized request), for instance: a preference for a teacher from a particular country or institution or the earliest available teacher. In order to realize a criterion-driven match, the Match-maker applies relevant rules and procedures, nevertheless needing input from EM's *DATA SEARCHER* (an entity that is also internal with regard to EM; this entity is responsible for searching). The Data searcher searches through the information concerning the available (Student/Teacher) entities and also applies procedures to it. This hence supports the identification of candidate matches relevant to the particular customer's request. The Match-maker applies its rules and procedures to realize a final match, passing this information to the EM's Advisor.

Considering the above-presented briefing, a business entity model is built (Figure 7), with a notation that is inspired by DEMO [6].

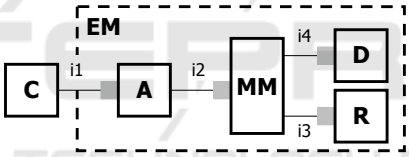

Fig. 7. Business entity model for the EM case.

The identified entities are presented in named boxes – these are Customer (C), Advisor (A), Match-maker (MM), Request handler (R), and Data searcher (D). Interactions i1 – i4 are identified as follows: between C and A (i1), between A and MM (i2), between MM and R (i3) and between MM and D (i4). As for the delimitation, C is positioned in the environment of the education mediation system EM, and A, MM, R and D together form the EM system.

We model then interactions using the notations of UML Activity Diagram [16]: i3 and i4 are to be progressing in parallel and only after they have been exhausted (the standardized requests and candidate matches have been delivered) the match-making can be done (i2) followed by the advice (i1) – this is illustrated in Figure 8 (upper part). This is the *business process level*, as labeled in the Figure, and it is assumed that human-driven roles (and responsibilities) stay behind each of the interactions and as it is about human activities, much is driven by complex organizational (and societal) norms, much is actually done using best practices, and much is done in an intuitive way. IT services nevertheless require many explicit definitions. That is why the IT services that correspond to the business-process-level interactions, are considered

together with other related issues, as it is shown in Figure 8 (lower part), depicting the *IT service level*, as labeled in the Figure.

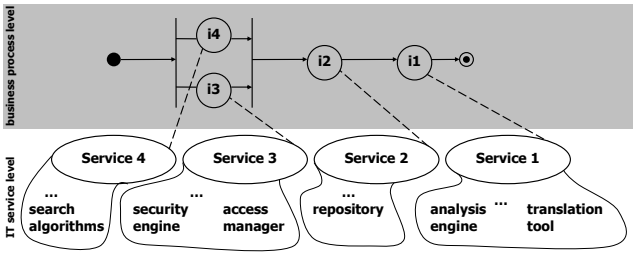

Fig. 8. Service derivation for the EM case.

As it is seen from the Figure, searching requires *search algorithms*, request processing requires an adequate supportive *security engine* and *access control facilities*, match-making needs *repositories* with candidate matches and match criteria, the delivery of an advice requires an *analysis engine* and sometimes, a *translation facility*, just to name a few.

We need to further extend this model, particularly with respect to 'IT Services level', by considering an adopted service pattern proposed in [11] according to which a coordination service (supported by an information service) orchestrates the work of the other services. Hence, the final EM service model is presented in Figure 9:

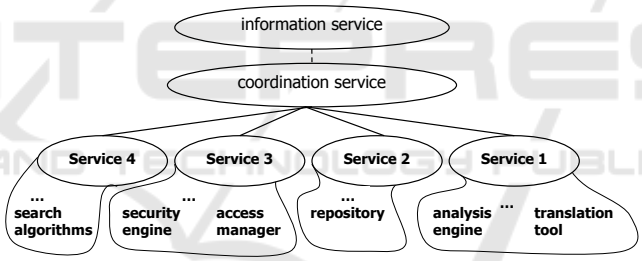

Fig. 9. Service model for the EM case.

As it is suggested by the Figure, the coordination service orchestrates the work of the other services, namely Service 1, Service 2, Service 3, and Service 4. Taking into account the case information and the considered domain, we label these 4 services in the following way:

- Service 1: **Educational Mediation Service**;
- Service 2: **Educational Broker Service**;
- Service 3: **Educational User Agent Service**;
- Service 4: **Educational Resource Discovery Service**.

This example, although simplified, partial and incomplete, illustrates the view on the paradigm shift, as discussed in Section 3, implicitly suggesting that business entities are not to be considered any more exclusively as source for identification of software

components but it could often be that the models of such entities are just valuable to actually give the right restrictions with regard to the services to be composed.

6 Related Work

In this paper, we have presented a service-orientation and context-awareness –rooted vision on software development, as a proposal for a desired reinforcement of traditional software development approaches and methodologies.

With regard to considering related work, among the work that is focusing on core service-orientation and partially on context awareness concerns are [15,16,17], driven mainly by consideration of particular key problems, such as service composition and web service technologies, overlooking nevertheless the issue of requirements-functionality alignment. On the other hand, there is reported research concerning the requirements-functionality alignment, mainly related to relevant methods, such as Catalysis [7], Tropos [8], and SDBC [5] which however lack the proper service-orientation focus.

7 Conclusions

In this paper, have discussed traditional software development approaches, putting the discussion in the perspective of some of the important current challenges related to software development, such as service-orientation and context awareness. Also, based on an analysis of the change brought to the desired vision on software development by these actual challenges, we have considered the impact of this change influencing already the ways software is developed. Finally, we propose a partial vision on how these new influences can be incorporated in software development, through some new software development tasks and roles. We have partially exemplified our vision.

Hence, the contribution of this paper represents an explicit proposal concerning the development of software and limited to proposed enrichments inspired by the consideration of the mentioned challenges.

As further research, we plan partial re-work of the SDBC approach, in the direction of service-orientation and adaptability.

References

1. Encyclopedia Britannica (Home), http://www.britannica.com
2. Software Engineering Institute (Home), http://www.sei.cmu.edu
3. Shishkov, B., Dietz, J. L. G., Liu, K.: Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In ICEIS'06, 8th Int. Conf. on Enterpr. Inf. Systems (2006)
4. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)
5. Shishkov, B.: Software Specification Based on Re-usable Business Components. Delft, The Netherlands: Sieca Repro (2005)

6. Dietz, J.: Enterprise Ontology, Theory and Methodology. Springer-Verlag, Berlin Heidelberg (2006)
7. D'Souza, D. F. and A.C. Willis: Object, Components, and Frameworks with UML, the Catalysis Approach. Addison-Wesley, USA (1998)
8. Tropos Project (Home). http://www.troposproject.com
9. Object Management Group (OMG): MDA – Guide, V1.0.1, omg/03-06-01 (2003)
10. Papazoglou, M. P.: Web Services: Principles and Technology. Pearson Pr. Hall (2008)
11. Shishkov. B. and Van Sinderen, M. J.: Service-Oriented Coordination Platform for Technology-Enhanced Learning. In: I-WEST'09, 3rd International Workshop on Enterprise Systems and Technology. INSTICC Press (2009)
12. Leymann, F.: Combining Web Services and the Grid: Towards Adaptive Enterprise Applications. CAiSE Workshops (2005)
13. Van Sinderen, M. J.: From Service-Oriented Architecture to Service-Oriented Enterprise. In: I-WEST'09, 3rd International Workshop on Enterprise Systems and Technology (2009)
14. Shishkov. B. and Van Sinderen, M. J.: On the Design of Context-Aware Applications. In: I-WEST'08, 2nd Int. Workshop on Enterprise Systems and Technology. INSTICC Press (2008)
15. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services, Concepts, Architectures and Applications. Springer-Verlag, Berlin-Heidelberg (2004)
16. Bosworth, A.: Developing Web Services. In: 17th Int. Conference on Data Engineering (2001)
17. Pasley, J.: How BPEL and SOA are Changing Web Services Development. Internet Computing, IEEE (2005)