Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

An Evaluation Framework for Business Process Modeling Languages in Healthcare

Amir Afrasiabi Rad[1], Morad Benyoucef[2] and Craig E. Kuziemsky[3]

University of Ottawa, Telfer School of Management
[1] a.afrasiabi@uottawa.ca, [2] benyoucef@telfer.uottawa.ca, [3] kuziemsky@telfer.uottawa.ca

Received 15 January 2009; received in revised form 28 May 2009; accepted 15 June 2009

Abstract

Web services composition is an emerging paradigm for enabling inter and intra organizational integration, and a landscape of languages and techniques for modeling business processes in web service based environments has emerged and is continuously being enriched. With the advent of modeling standards, different business sectors are investigating the options for modeling their workflows. In terms of business process modeling, healthcare is a rather complex sector of activity. Indeed, modeling healthcare processes presents special requirements dictated by the complicated and dynamic nature of these processes as well as by the specificity and diversity of the actors involved in these processes. Little effort has been dedicated to evaluating the capabilities and limitations of modeling languages based on healthcare requirements. This paper presents a set of healthcare modeling requirements and proposes an evaluation framework for process modeling languages based on these requirements. The suitability of two major process based service composition languages, namely BPEL and WS-CDL, is evaluated.

Key words: Business Process Modeling, Modeling Language, Web Service Composition, Healthcare, BPEL, WS-CDL

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

1 Introduction

Web services foster the integration of disparate systems and applications despite being developed at different times by different people. Efforts on standardization of web service composition and modeling have resulted in the release of two major process based service composition streams, orchestration and choreography, as well as different standards and languages, namely Process Definition Language (PDL), XML Process Definition Language (XPDL), Business Process Schema Specification (BPSS), Business Process Modeling Language (BPML), Web Services Choreography Interface (WSCI), Electronic Business using eXtensible Markup Language (ebXML), Business Process Modeling Notation (BPMN), Web Services Choreography Description Language (WS-CDL), and Business Process Execution Language for Web Services (WS-BPEL).

The increase in the number of modeling standards and the diversity of business sectors where process modeling can be applied has rendered modeling very complex. Hence, the question of identifying suitable modeling standards for specific business sectors has arisen. Many frameworks [11], [12], [20], [36], [38] have been proposed for evaluating the suitability of a process modeling language for a given sector. However, most of the frameworks only focus on one or two aspects of languages, and evaluation results usually present overlaps, limitations, inconsistencies, and ambiguities. These limitations are particularly observable when modeling complex systems such as healthcare. The complexity of healthcare business processes results from the integration of workflows, and collaboration and data transactions between different units. Meanwhile, the existence of different medical data transaction and process execution standards (e.g., Health Level Seven (HL7) and Digital Imaging and Communications in Medicine (DICOM)) further adds to the complexity. Moreover, an increase in the number of medical disciplines and the dynamic nature of healthcare delivery call for dynamic process models [2].

A process model is not just a graphical representation, but rather it should also serve as a communication base both for communicating domain details between stakeholders, and for communicating domain details to system designers [11]. Moreover, according to [18] models should be representative, easily understood, easy to use, optimized in the level of details, and support abstraction. The ability to use a process model for communication decreases if modeling specialists are the only people who understand the models. Therefore models should be representative enough to be understood by all model users such as healthcare administrative or clinical stakeholders. A further issue is integration. If each unit of a healthcare organization uses a modeling language that best fits its requirements, the model of the entire organization will be a combination of several standards making the integration and understandability of the models difficult.

The overall goal of this paper is to develop an evaluation framework for assessing the suitability of service-based business process modeling languages in healthcare. We achieve that objective in three steps. First we outline the difficulties of modeling complex healthcare processes in a service based environment. Second, we introduce the healthcare domain, the business processes within it; healthcare's modeling requirements, and challenges of healthcare business process modeling. Third, we introduce our evaluation framework, and then use it to assess the suitability of two selected modeling languages in order to model a complex healthcare process, the Scheduled Workflow (SWF) process from the Integrating Healthcare Enterprise initiative.

The paper is organized as follows. After a quick coverage of related work in Section 2, Section 3 discusses the requirements for modeling healthcare processes. In Section 4, based on the identified requirements, we introduce a framework for evaluating business process modeling languages in healthcare. In Section 5, we use the framework to conduct a comprehensive evaluation of two leading modeling languages, namely BPEL and WS-CDL. In Section 6 we investigate combining two modeling languages to address deficiencies of a single language. Concluding remarks are presented in Section 7.

2 Background

Modeling languages have fundamental differences with regard to expressiveness, flexibility, adaptability, dynamism, and complexity aspects [19]. A consequence is that different classes of modeling languages suit different business sectors. Explorations of common factors of modeling languages concluded that the understandability and complexity of models have a positive relationship [13]. However, to date, focus has been on common features of all modeling languages while unique features have been neglected.

2.1 Assessment of Modeling Languages

A quality framework for the evaluation of modeling languages has been described in [25]. However the quality framework does not suggest any definite metrics for evaluating languages, and the comparison of modeling languages is only case based and tightly related to the domain's quality area. Other research evaluated the understandability of models, providing clear metrics that make the measurement of understandability more accurate. Moreover, [21] classified the characteristics of modeling languages and defined a set of steps for their selection.

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Workflows and data flows are important factors in an evaluation framework. Van der Aalst et al. and Wohed et al. [35], [36], [40], [41] extracted Workflow, Data flow, and Control flow patterns, and provided detailed evaluations of modeling languages based on the abovementioned patterns. Green et al. [12] provided an ontological framework for evaluating process modeling languages. Languages were evaluated based on their ability to model ontological constructs. Results show that the number of constructs that each business sector implements is different, and that some modeling languages are unable to represent various ontological constructs or the representations are too complex.

2.2 Healthcare Process Modeling

Healthcare processes are complex owing to the need for decision making, communication, and information sharing that takes place asynchronously across multiple providers [32]. The evaluation frameworks mentioned earlier do not transfer well into complex sectors like healthcare that have specific requirements such as constant evolution. Thus specific initiatives have been developed to address the unique needs of healthcare. Integrating the Healthcare Enterprise (IHE) is an initiative to develop standard compatibility in healthcare (Site 3). IHE also defines and documents healthcare business processes for a standard and unified system workflow [15]. Since IHE documented healthcare processes comprehensively, many research initiatives utilized IHE workflows for identifying healthcare requirements. Anzbock and Dustdar [2] explored IHE workflows to study healthcare process modeling in a web service based environment. Their work is limited to the evaluation of BPEL in test cases and the identification of considerations for modeling healthcare web services in BPEL. They defined transport, security and reliability as healthcare process model requirements and provided solutions for them in BPEL. Since their work is limited to the applicability of BPEL, and they have not analyzed the resulting models based on an evaluation framework, we believe further research should be done on the appropriateness of modeling languages besides their applicability. Furthermore, the authors argue that some healthcare process modeling challenges, which come from the immaturity of the web service stack standards, remain unsolved.

Roseman et al. [31] and Green et al. [12] compared process modeling languages from a representational aspect. They performed an ontological analysis but neglected other factors that contribute to the evaluation of process modeling languages. Hence, although the analysis of expressiveness is important in the evaluation of workflow languages, it is not sufficient to evaluate the overall usefulness of a language based on its expressiveness. Rossi and Turrini [32] considered a wide range of requirements for their analysis of modeling languages. Their research is novel in terms of identifying features of modeling languages that contribute to their usefulness. However, by focusing on simplicity, they analyzed the languages from the users' perspective and paid little attention to systems requirements. Complex systems, like healthcare, have specific requirements that sometimes cannot be met by simple language features. Mendling et al. [22] investigated two major frameworks for evaluating modeling languages and proposed their own framework based on software engineering criteria. A similar work was also described in [25]. Although the goal was to analyze process models, the focus on software engineering methods limited the evaluation to process modeling languages.

Existing research on business process modeling does not completely address the requirements of healthcare. Although there are works that focused on security [14], [30] and dynamicity [17], other important requirements are not considered. Moreover, research on language simplicity and understandability [22], [31], [32] only focused on a certain group of actors while healthcare actors are different in terms of their IT and modeling knowledge. Therefore, there is a lack of a comprehensive analysis of healthcare process modeling considering all its modeling requirements.

2.3 Choreography and WS-CDL

Choreography defines collaboration rules and interactions between two or more business entities or processes. It describes all participants' observable collaborative behavior from the perspective of all involved parties, and it enables the use of web services in a collaborative manner by providing the means for describing the behavioral interface of a single web service, and for specifying interaction protocols for web service collaborations [4].

WS-CDL is an XML-based language for describing service interactions in a peer-to-peer manner where each involved entity remains autonomous and in which there is no coordinating web service or any coordinated one. It is used to describe the collaborative observable behavior of multiple interacting service based entities that pursue a common goal. Interactions between entities are documented and described as a set of message exchanges which form a contract between entities and set a basis for their further interactions. Since WS-CDL is not explicitly bound to WSDL, it is possible to use WS-CDL for describing collaboration protocols for both services which have no WSDL descriptions and those that have WSDL descriptions [29].

2.4 Orchestration and BPEL

Orchestration, one of two main service composition methodologies, is seen through the Business Process Execution Language (WS-BPEL or BPEL). In the context of orchestration, a central process takes control of the involved web services providing a workflow style composition of services and coordinating the execution of operations on the involved web services. Orchestration imposes a level of abstraction on all involved web services, except the process

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

coordinator, as they do not need to know that they are involved in a composition process, and they only focus on incoming and outgoing messages [21]. BPEL is a language for specifying and executing service orchestration processes against a domain of web services. It is built on Web Services Flow Language (WSFL) and Web Services for Business Process Design (XLANG), and inherits its ancestors' graphical and text based notations. Genuine BPEL is notably used for creating executable processes, while it is able to model abstract processes forgoing execution [5].

3 Required Features for Healthcare Processes Modeling

Modeling complex processes has always been a challenge, and, as systems and models get more complex, the use of complicated models becomes challenging for model users. As an important factor in modeling, models should be easily understood by their target users in addition to containing an appropriate level of detail to fulfill their development purposes. Furthermore, depending on system requirements, completeness, extensibility, and the ability to represent security and exception handling features are important criteria. These requirements are more coherent in the context of modeling complex systems. Specific features of complex systems impose the existence of many previously mentioned features in their models. Not unlike other complex systems, healthcare systems have specific modeling requirements.

Referring to previous research on modeling web service-based healthcare processes, Anzbock and Dustdar [1] identified four major challenges of modeling medical web services:

- a high degree of vertical standardization through DICOM, HL7 and IHE
- currently implemented systems based on conventional middleware
- lack of inter-organizational workflow support as a common problem
- no current web services-based approach which tries to fill this gap

The authors argued that some challenges are caused by the immaturity of the web service stack standards and the existence of competing standards in that context, and some challenges have not been solved yet. The identified shortcomings are mainly discussed in the implementation phase, but the lack of integration, workflow support, and suitable system architecture imply design and modeling level deficiencies. Plesk et al. [27], [28] and Fraser et al. [9] also investigated healthcare processes, but not from a modeling perspective, yet their work is useful in identifying the challenges of healthcare processes and environments. Combining those challenges with the modeling deficiencies explained in [18] - [20], [23], [25], [26], [30], [31] lead us to identify healthcare process modeling challenges more accurately. In this section we elaborate on these deficiencies as well as healthcare's modeling requirements and complexity, with special attention paid to features that should be included in modeling languages to satisfy those requirements.

Complexity of Processes: Healthcare delivery is comprised of complex processes since modern health care delivery frequently involves multiple types of providers with different knowledge levels, often residing in different physical locations. Two measures of process complexity are the number of units involved in the process and the number of transactions between collaborating parts [13]. For example, the radiology subsystem in the IHE framework contains 34 units, more than 65 transactions, and each transaction contains several message types. Furthermore, individual departments such as radiology frequently collaborate with other departments as part of the provision of patient care and such collaboration adds to the complexity of healthcare processes. Efforts have been made to model collaboration such as (Site 1) who defined seven specifications of a business process, four of which are directly connected to organizational collaboration. Although business process modeling languages have focused on representing collaborations as a key goal [6], research has shown that modeling collaboration can be difficult. The results presented in [13] show that generated models for collaborative systems are still far from meeting expectations.

Security and Privacy: The involvement of several actors in a process increases the probability of security and privacy related issues arising during the development of a system and during its use. This is especially important in healthcare systems as they often interact with other organizations such as insurance, financial and billing companies that need detailed information about patients. Therefore, representing security features in order to determine the type of communication channel for interaction among a large number of contributing partners, and also privacy features for representing and classifying transmitted data for each partner, are important aspects of model development for healthcare systems. Security and privacy are particularly important when communication is conducted over the Internet [28] such as within a service oriented environment which gives access to the systems from anywhere on the network. Hence the models need the ability to represent security and privacy considerations.

User Understandability: Modelers and system managers pursue a number of goals in modeling processes, especially complex ones such as healthcare processes. Although models are being developed for documenting and describing systems, they should also enhance the stakeholders' general understanding of processes. Complex systems are usually far from being mature and optimal, thus process redesign and reengineering take place to reform process inefficiencies, reduce cycle time, and remove non-value-added tasks [9]. The important question is

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

that if models do not provide a vivid picture of processes to the systems' stakeholders, how can process inefficiencies be identified? Models are also being used for training purposes and decision making procedures in addition to system documentation and development. For both training and decision making purposes, models should provide a graphical view of the system [18]. Training in healthcare systems with a huge number of staff, departments and relations is an important issue and models can play an important role as training tools. This becomes even more important when target users are IT professionals. In fact, efficient and effective model design mostly depends on the modeler's creativity, but some modeling language features such as uniqueness of the language's notational constructs and following standards for representing each construct improve the degree of understandability of the model by target users.

Optimized models for different purposes: To deal with the complexity of healthcare processes, process modeling languages should allow modelers to create optimized models with different numbers of specifications for different purposes. For models used by healthcare staff, for instance, more detailed models can be confusing; hence they should only represent the high level graphical representation of processes. However, models used by IT department staff require detailed specifications. To comply with the need for different views, flexibility is required in the level of details and representation.

Evolution of processes: The new definition of success for health services describes healthcare's goal as not merely change, improvement, and response, but changeability, improvability, and responsiveness [28], which can be seen as the "evolution" dimension of healthcare process complexity. As a result, constant evolution and the dynamic nature of healthcare systems are the two main contributing factors to the complexity of processes. New technologies and medical techniques lead to the creation of new expertise, so processes change in order to adopt new services [27]. Evolution and change in systems introduce new requirements. Currently developed system models should accommodate solutions to new requirements. Moreover, a modeling language should adapt to a wide range of changes in models without imposing the need for extensive remodeling [25]. Modularity and support for abstraction are key concepts for change adoption in process models, and they are also pre-requisites of loose design. Modular models are easy to read and less resistant to change. Modularity also increases reusability in model development. Furthermore, due to the dynamic nature of healthcare, an agent's actions do not always follow normal routines since many emergency situations occur. In emergency situations speed has the most important role; hence when agents react, they usually do not follow the routines which may lead to conflicts. Modeling languages should be able to represent the dynamic nature of healthcare processes through exception handling.

Nested processes and Integration: In complex systems different agents may simultaneously be members of more than one unit and memberships may change frequently due to unforeseen events. Nested healthcare processes create inherently non-linear relationships and remove boundaries between sub-processes [22]. This reality complicates the modeling task and increases the difficulty of changing a model as internal rule sets are not very strict. Therefore process modeling languages should be able to represent healthcare's fuzzy departmental boundaries [22]. The influence of change in complex systems is greater if inter-departmental interaction is tight. A change in an agent's task can affect other agents, so changes can be transmitted to other units in an incremental manner due to the non-linear behavior of complex systems. Integration of agents and departments across different subsystems is a common element of complex systems. Different subsystems may have dissimilar features and therefore divergent modeling requirements. In order to accommodate all requirements in models, it is sometimes essential to use different modeling languages for different departments. These models should be able to represent communication between departments and this would not be possible if modeling languages do not have matching features and the ability to be mapped to each other. We need to be able to map different modeling languages or integrate them together as needed to satisfy all requirements.

4 An Evaluation Framework for Business Process Modeling Languages in Healthcare

In Section 3 above we identified general requirements for healthcare process modeling languages. In this section we propose a framework with eight components to address such requirements. As will be detailed below, one requirement is usually met through one or several components/criteria of the framework.

Process modeling languages are innovatively designed or developed from existing modeling approaches by unifying several methods or adding features to existing ones [26]. We developed a framework (Figure1) for evaluating modeling languages for service based healthcare environments. In addition to modeling languages' common features, we consider the unique features of languages given that they are key metrics for specific uses. We also consider general modeling qualities, which are required by all systems, in addition to addressing healthcare specific requirements. We identified important features that allow us to measure the quality of modeling languages. In this section we explain our proposed evaluation framework and the metrics used for measuring each evaluation criterion.

Security and privacy: Security is important when dealing with service oriented architectures and communications over the public Internet. For all the various uses of the models (training, development, documentation, etc.) security and privacy issues must be considered. Moreover, processes themselves must sometimes be protected. In some

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

cases, the process should not be exposed to partners and communication should be through a special interface which keeps the process private.

Pattern representation: The understandability of a modeling language is a fundamental requirement for increasing the language's usability, and is tightly related to the process modeler's capabilities in addition to the modeling language's features. Factors such as the modeler's level of expertise, creativity, and familiarity with the business and tools are inevitable. A modeling language's capability to represent data flow, communication, and control flow patterns is an important criterion for the understandability of the language. Studies in [34] - [36] showed that some patterns cannot be modeled by any modeling language, and some patterns can only be modeled using a limited number of languages. Thus the understandability of modeling languages depends on the ability and methods of representing patterns. If there is a construct in the language that directly represents a pattern, the understandability of the language will increase. However, combining several constructs in order to represent a pattern decreases the understandability and makes the language ambiguous.

Pattern representation is important in the context of healthcare modeling since more unambiguously supported patterns mean a decrease in model complexity. Model complexity decreases due to the fact that each healthcare process is a combination of patterns, and as modeling languages provide elements corresponding to patterns, the model size and complexity decrease. Furthermore, familiarity with patterns and language elements increases model understandability; therefore processes can be better understood by different healthcare clinical and managerial actors.

Workflow Patterns were introduced by Van der Aalst et al. in (Site 2). They serve as a reference framework for assessing process modeling languages and workflow systems in terms of control flow expressiveness. BPEL and WS-CDL as well as a number of other languages have already been assessed [8], [35], [36], [40], [41]. These assessments serve as a reference for verifying if direct, partial or no support for a particular pattern is available.

Ontological Completeness: Language concepts are evaluated using ontological constructs. The BWW (Bunge-Wand-Weber) ontological framework is proposed in [37] for evaluating modeling grammars. The authors argue that modeling notations which are not able to represent all of the ontological constructs are incomplete. Language incompleteness reduces understandability because incomplete language features and structures force the modeler to ignore some features of models as there is no construct supporting those specific needs. Nevertheless, while there may be other ways to represent the needed constructs in order to compensate the languages' incompleteness, these ways are diverse and cause ambiguity.

Pattern support and ontological completeness are similarly effective in the context of healthcare. Both criteria are concerned with modeling languages' representational capability.

Extendibility: As discussed earlier, modeling languages may not support all the requirements of modeling healthcare processes. To support the requirements, there is the option of combining the language with other possible solutions, or extending the language to support the requirements. Extensions help to support different technologies and different business sectors, and also to overcome modeling language deficiencies; however, extending modeling languages may affect the understandability of the languages and models, which represents an important aspect of healthcare modeling.

Figure 1: Evaluation Framework

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Furthermore, model developers will have to learn additional features of the modeling language to be able to maintain the model during further development. Also, model users are only familiar with basic model notations and language concepts, so extensions to a language can create confusion for those who are not familiar with the extended features. This problem might be an issue in healthcare where actors do not have extensive modeling and IT knowledge. Consequently, extendibility is a compromise between increased capability and general understandability.

Extensions and language extendibility are almost inevitable to address some challenges in the healthcare modeling process (e.g., security representation as will be discussed in the next section), although they create additional challenges. The extendibility factor affects all the identified criteria except the diversity of actors. As discussed earlier, since the level of IT knowledge among healthcare staff is different, extensions may require further training for the management and IT support staff.

Notations: Modeling languages are categorized, based on notations, into graph based languages, rule based languages, and their combinations [19]. Language notations can be measured by the way they follow the standards (e.g., standard element size and colors for graphical notations). Textual notations can be measured based on the same concept. The keywords representing concepts and descriptive words should follow their existing standards. For languages that provide execution features, separation of computation and representation of processes differentiate two diverse aspects of modeling, so each actor can access a part of the model which addresses his needs. Meanwhile, uniformity (i.e., use of the same set of notation with unique and unified meanings) and formality (i.e., choosing commonly accepted graphical notations for language concepts) of notation improve the language's ease of use [19].

Language notations affect the language's understandability, which is important given that the language will be used by diverse actors with different knowledge levels. Since actors involved in healthcare environments have different IT and modeling knowledge levels, a coherent and standard language notation improves model use for all actors.

Modularity: The modularity of modeling languages can be measured by considering the support for abstract processes and sub-processes. Abstraction prevents from revealing the underlying layers of a process, hence improving the understandability of process models for non-developer actors. Also, the understandability of developers and designers for models will improve when the unimportant data is hidden. Reusability is an important factor and it is tightly related to modularity. The ability to use previously modeled processes increases the speed and accuracy of modeling. It also reduces the time required for understanding the models. Maintenance and model modification with reused parts is easier. Reusability has a positive connection with understandability in a way that more reused sections in models help decrease the required effort for learning the process. However, it is not necessarily true that languages that support sub-processes also support reusability.

Modular design reduces the complexity which is one of the main issues in healthcare models. It also increases the ability to hide unnecessary information for different model users and various model uses. This factor is important in healthcare because of the existence of different actors with different grasp levels of models. Finally, modular models can tolerate the evolving nature of healthcare processes better.

Level of detail: The level of detail provided by modeling languages should be optimized for different modeling purposes. This metric has a close positive relationship with other aspects of a modeling language such as notation, abstraction and ability of the language to execute processes. Detailed documentation of processes improves the level of understanding for the designer and developer actors. However, the end user, who uses the models for training, does not need detailed information, and more detailed documentation may impair his understanding. Consequently, modeling languages should be flexible in the level of detail, and provide facilities to support both detailed documentation and high level representation of processes. This criterion can be measured by support for abstract processes. Also, languages should not force modelers to create detailed models. Languages that support execution should be able to separate execution and representation in order to hide unnecessary details.

Flexibility in the level of detail is essential in modeling healthcare processes because of the diversity of actors and their expectations from models. High level and less detailed models enhance the general view of the model for healthcare's high level management actors who are not concerned with implementation details. However, actors who are involved in the details of processes or are members of an IT team need a fully detailed view of the system. For the first group, graphical models are better, but the latter group needs a detailed description of processes that can be achieved using text-based models.

Exception handling: Exceptions have been indispensable parts of business process-modeling for more than two decades [7]. Exception handling features increase modeling difficulty but also increase the adaptability of models to exceptional circumstances. Improved adaptability helps model users to predict all model behavior in the time that exceptions occur. Investigations of modeling languages show that not all modeling languages contain cancellation functionalities. In the case of those languages, the processes end naturally; the modeler does not have the ability to terminate a process at a special time. Being able to define the termination conditions and terminate processes at a certain moment improves the understanding of process timelines.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Healthcare environment and its processes are full of exceptions in various forms. There are many emergency situations occurring everyday that do not follow routine procedures and workflows. However, most emergency situations have their own defined procedures that are closely followed by caregivers and healthcare administrations. These exceptional situations and the way that they should be processed must be foreseen in model development. Use of exception handling features improves readability of decreases process model complexity.

The evaluation framework we propose here provides a starting point to evaluate different modeling languages in the healthcare domain. The strength of the framework is its extensiveness with regard to covering unique features of languages, which makes it an unbiased test-bed for comparing different language families. Implications for general modeling are also considered.

5 Assessment of Modeling Languages

In this section we make a thorough evaluation of BPEL and WS-CDL using our proposed framework. Based on what we described in sections 2 and 3, identifying a suitable web service based modeling language for the healthcare sector is difficult. Although our assessment focuses on language features, it also leads to a better understanding of each language's backbone methodology. Since choreography and orchestration are two main service based modeling methodologies, we chose two leading languages from each side in order to conduct our assessment. Each language's features will be assessed based on framework criteria to investigate their compliance with the identified requirements.

Security and Privacy: Neither of the languages contains constructs to represent task level or process level security in the models and neither provides a complete secure collaborative basis in the execution. Although WS-Security [24] can be embedded in transmitted messages between services in both languages, it does not provide any representational aspect, and cannot be identified by non-modeling experts. The expressiveness of process languages becomes important in complex business sectors such as healthcare where non-modeling experts interact with models for many purposes including training, decision making and process improvement. Moreover, several extensions to modeling languages have been developed to provide security requirements. Yet, as an important language requirement, embedded security features in the languages are more recommended due to some deficits that extensions may cause in the languages. These deficits will be explained later in this section.

As a language meant for execution of services, BPEL takes advantage of more developed extensions that contain added security features. However, these extensions which contain representational features need extra learning effort, plus modeling tools do not support all extensions and probably need some plug-ins installed. On the other hand, since WS-CDL is not aimed for the execution of services, security measures are less important. There exist some techniques to represent security features in WS-CDL with the help of other modeling languages [14] which do not introduce significant changes in the use and portability of models. Since WS-CDL is XML-based, is mainly executed by support of Java services, and does not have a graphical view, the portability of models would not be affected. However, the issue of model understandability for users who are unfamiliar with these techniques and extensions remains. Moreover, in the theory of service composition, since corporate entities are often unwilling to delegate control of their business processes to their integration partners, WS-CDL (or service choreography in general) offers more privacy by means of participation contracts and elimination of a central process.

Pattern support: WS-CDL is a language used to unambiguously describe observable service collaborations. It is used to describe internal workflows of processes that involve multiple services, yet it does not support all workflow patterns documented in (Site 2). Based on Van der Aalst's work and similar research, it is proven that none of the current process languages, including BPEL and WS-CDL, is capable of representing all known documented patterns. However, the level of support differs significantly in different languages. More supported patterns mean more modeling power (in most cases) and high levels of language understandability for modelers and model users. We use the term "in most cases" because pattern support does not necessarily mean that the language provides an unambiguous way for supporting the pattern. If the language supports the pattern by combining two or more constructs, the representation of that pattern may cause some ambiguity, even more so if the language has two completely different methods for representing a pattern. For instance, in BPEL, Interleaved Parallel Routing can be implemented by two totally different sequences of diverse elements.

The result of evaluating each pattern in conformance with the languages' capabilities is presented in (Table 1). The capability of modeling languages in representing patterns and the understandability of their method in that representation are separated by a slash (/) in the table.

Ontological Completeness: Findings in [12] on ontological mapping of modeling languages and Bunge-Wand-Weber (BWW) ontological base model show that modeling languages cannot represent all the ontological constructs documented in [37], [39]. Although all ontological constructs are not used in every system and process, a more complete language increases model development speed, as well as the readability of models. We used the method proposed in [12] to map WS-CDL to BWW. The results are presented in (Table 2).

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Extendibility: Both BPEL and WS-CDL are extendable and can accept new features to the language. For healthcare modeling both languages need a security extension but BPEL also needs an extension to support sub-processes. On the other hand, an execution extension to WS-CDL can relieve modelers from using Java services in order to implement WS-CDL choreographies.

It is important to mention that WS-CDL is incapable of showing the order and direction of transferred messages between services and processes, and all that WS-CDL describes is the ordering for the messages. Hence, as a process language, WS-CDL needs a notation that represents the flow of messages in the process.

Notations: Although both BPEL and WS-CDL follow a fairly accepted standard (i.e., XML) for their notations, WS-CDL has some drawbacks with regard to constructs and their naming. From a graphical notations' viewpoint, BPEL is the only one which has a strict format and a set of accepted graphical notations while WS-CDL is barely intended for graphical representation of processes. However, BPEL's notation is not easily understandable by non-modeler actors since its textual and graphical notations are entwined, hence a model user must understand BPEL's background coding as well as its graphical notation. Moreover, this inseparability makes changes to the graphical notation rather cumbersome for the non-expert user.

Table 1: Pattern support of BPEL and WS-CDL

Pattern	Language (capability / understandability)	
	BPEL	WS-CDL
Sequence	+/ +	+/+
Parallel Split	+/(+/-)	+/+
Synchronization	+/(+/-)	+/+
Exclusive Choice	+/+	+/+
Simple Merge	+/+	+/+
Structured Discriminator	-/-	-/-
Arbitrary Cycles	-/-	-/-
Implicit Termination	+/+	+/+
Multiple Instances without Synchronization	+/(+/-)	+/+
Multiple Instances with a Priori Design-Time Knowledge	-/-	+/+
Multiple Instances with a Priori Run-Time Knowledge	-/-	-/-
Deferred Choice	+/+	+/(+/-)
Interleaved Parallel Routing	(+/-)/-	-/-
Cancel Activity	+/+	+/+
Cancel Case	+/+	+/+
Structured Loop	+/(+/-)	+/+
Recursion	-/-	-/-
Transient Trigger	-/-	-/-
Persistent Trigger	+/+	+/+
Cancel Region	(+/-)/-	(+/-)/-
Blocking Discriminator	-/-	-/-
Cancelling Discriminator	-/-	-/-
Structured Partial Join	-/-	(+/-)/-
Blocking Partial Join	-/-	(+/-)/-
Cancelling Partial Join	-/-	(+/-)/-
Critical Section	+/+	-/-
Interleaved Routing	+/+	-/-
Thread Merge	(+/-)/-	(+/-)/-
Thread Split	(+/-)/-	(+/-)/-
Explicit Termination	-/-	+/+

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Table 2: Ontological representation of BPEL and WS-CDL

Constructs	Language (capability)	
	BPEL	WS-CDL
Property	+	+
Class	+	-
State	+	+
Conceivable state space	-	-
State Law	-	-
Lawful State Space	-	-
Event	+	+
Conceivable Event Space	-	+
Transformation	+	+
Lawful Transformation	+	+
Lawful Event Space	-	+
Coupling	+	+
System	+	+
System Composition	+	+
System Environment	-	-
System Structure	+	+
Subsystem	-	+
System Decomposition	-	+
Level Structure	-	+
External Event	+	+
Stable State	-	-
Unstable State	-	-
Internal Event	+	+

Without considering its lack of graphical notation, WS-CDL still suffers from possible confusions resulting from the naming of its constructs. Several WS-CDL constructs, like RoleType, ParticipationType, and ChannelType, give the wrong impression that there should be some possibility to create separate instances from them, but they function as entities. The existence of *Type* in their naming definition may create some confusion among language users [10]. Also, as an XML-based non-graphical language, specifications of even the smallest interactions become rather lengthy. The language also introduces a wealth of new design concepts (e.g., channel passing), constructs for direct expression of concurrent execution and nondeterministic choices, that are currently not well understood in the world of web services. The channel passing mechanism is poorly explained in the standard. It is not exactly clear which conditions the reference text puts on channel usage. Moreover, in general, the channel passing mechanism sounds overly clumsy. The intention of the language designers with regards to channel mechanisms is, in general, far from clear.

Modularity: The overload of transmitting information, the high amount of communication, and the need for reusability make sub-processes an indispensable feature in healthcare process modeling. BPEL does not genuinely provide any support for sub-processes as standard. Some transactions take place between different actors who are assigned an internal process determining the type of messages that will be communicated in a transaction, and create the information in a way that is necessary for the message type. Consequently, there is a process running inside each actor. Using BPEL, the goal is to model processes in a web services environment. For implementing the whole model as an orchestration, the central process will regulate communication between services if actors are considered as service providers. In this system, however, actors need to perform as internal processes. There are two ways to represent this system in BPEL. The first way is to provide an abstract central process with conceptual transactions between units in addition to actors' internal processes modeled in different BPEL processes with a reference to a central process. This makes the process simple and easy to understand without forgoing the model's execution feature, each sub-process can be executed separately. The second way is to use extensions. While extensions to BPEL do not affect its understandability by users familiar with extensions, understandability decreases for non-professional users. Moreover, the use of extensions decreases the portability of models.

On the other hand, WS-CDL is well suited for reusability purposes. It coordinates the services without need of a central process that makes the alternation of models more difficult, and choreographies can be created from parts contained in several different choreographies. So, existing choreographies can be combined to form new

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

choreographies that may be reused in different contexts, and the same choreography definition is usable by different participants operating in different contexts. Furthermore, since the contracts are described in a modular fashion, reading and learning the processes is much easier. Based on that, WS-CDL can easily represent sub-processes and the abovementioned scenario's internal and external one-to-many relationships. Besides, strict modularity of WS-CDL takes away all global variables, conditions or work units from the modeler, but still enables centralized storage to have global variables in order to avoid repetitive definitions.

Level of Detail: Different healthcare model users require different views of the processes. Different levels of detail specify the intended use of the models. To a large extent, BPEL is inflexible in the level of detail that can be provided to the end user. It resembles more a programming language than a modeling language and does not allow for any form of analysis. BPEL process models are associated with coding which contains a high volume of detail. BPEL's limitation is the inseparability of graphical notation from its background execution code. Healthcare managerial staff does not need to see detailed code while it might be necessary for IT staff. The inflexibility of BPEL at the detail level makes the modeling, on some levels, cumbersome and decreases the understandability of models by non-IT actors. WS-CDL is almost in the same situation with a difference in its ability to define abstract service interactions. Yet it is still too detailed for non-development purposes, and its lack of graphical representation makes it almost unusable for training purposes even though it can include human readable documentation and semantics for all the components in the choreography.

Exception Handling: Both languages provide a complete set of exception handling and compensation features. While these features are more important in BPEL, healthcare processes demand adequate support for exception handling to increase process understandability. However, from a representation point of view, BPEL is limited in dealing with exceptions since there is no cancellation element. The modeler cannot decide to end a process instantly; the process continues until it gets to the ending point. Due to this limitation, the result would be a larger model with many more connections even though the model will be easier to follow as it is terminated only at one final point.

Exception mechanisms in WS-CDL are poorly explained in general. For instance there is some confusion in the standards documentation regarding whether exceptions impact only certain role types in choreography, or whether all role types are impacted. The likely intention is that choreography coordination is required to notify all role types, but this is inadequately specified in the standards documentation [10], [16].

5.1 Practical Evaluation

In this subsection, both languages are tested in their ability to model a complex healthcare service based process. Scheduled workflow (SWF) is an IHE process that represents the ordering, scheduling, imaging acquisition, storage and viewing activities associated with radiology exams. Consisting of twenty transactions and eight different IHE Actors, SWF is a good representation of the complexity of healthcare processes because of its complexity, large amount of domain communication, and service like behavior. The SWF provides a good test case for the evaluation of our framework.

The SWF establishes the continuity and integrity of basic departmental imaging data. IHE integration profiles discuss the integration needs of healthcare sites and the integration capabilities of healthcare IT products (Site 3). IHE radiology integration profile specifies a number of transactions throughout the imaging acquisition procedure in order to maintain the consistency of patient and ordering information. It also provides the scheduling and imaging acquisition procedure steps [15]. This profile also provides a central coordination scheme for process completion and scheduling notifications. (Figure 2) represents the complexity of transacted messages as well as the large number of involved units in SWF.

As a test base we selected to represent a procedural section of SWF because it has a more service based process like behavior. The admission procedure which includes several transactions and inter-departmental interactions between Order Pacer, Order Filler and ADT, is the most suited fraction of SWF for our purpose. It is complex enough to show the capabilities of modeling languages, and it is the most used process section in SWF, so its reliability and understandability are important. Furthermore, based on the day-to-day evolution of healthcare systems, it is more likely to change than other processes.

The first step in modeling is to understand the process and its requirements, here, participating services. The sequence diagram in (Figure 3) represents the interaction flow and its timing between services on patient admission when a request is made to perform an imaging procedure on a patient. UML sequence diagrams are widely used as a representation language for workflows, especially when there are many transactions involved in the process. In addition, knowing the transmitted messages and their flow will help to create robust communication protocols in choreography which will be discussed later.

The sequence diagram is derived from the one presented in the IHE documentation [15]. The compensation messages are presented for the readability of processes when they are modeled in BPEL and/or WS-CDL. The involved services can be easily identified in the diagram.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

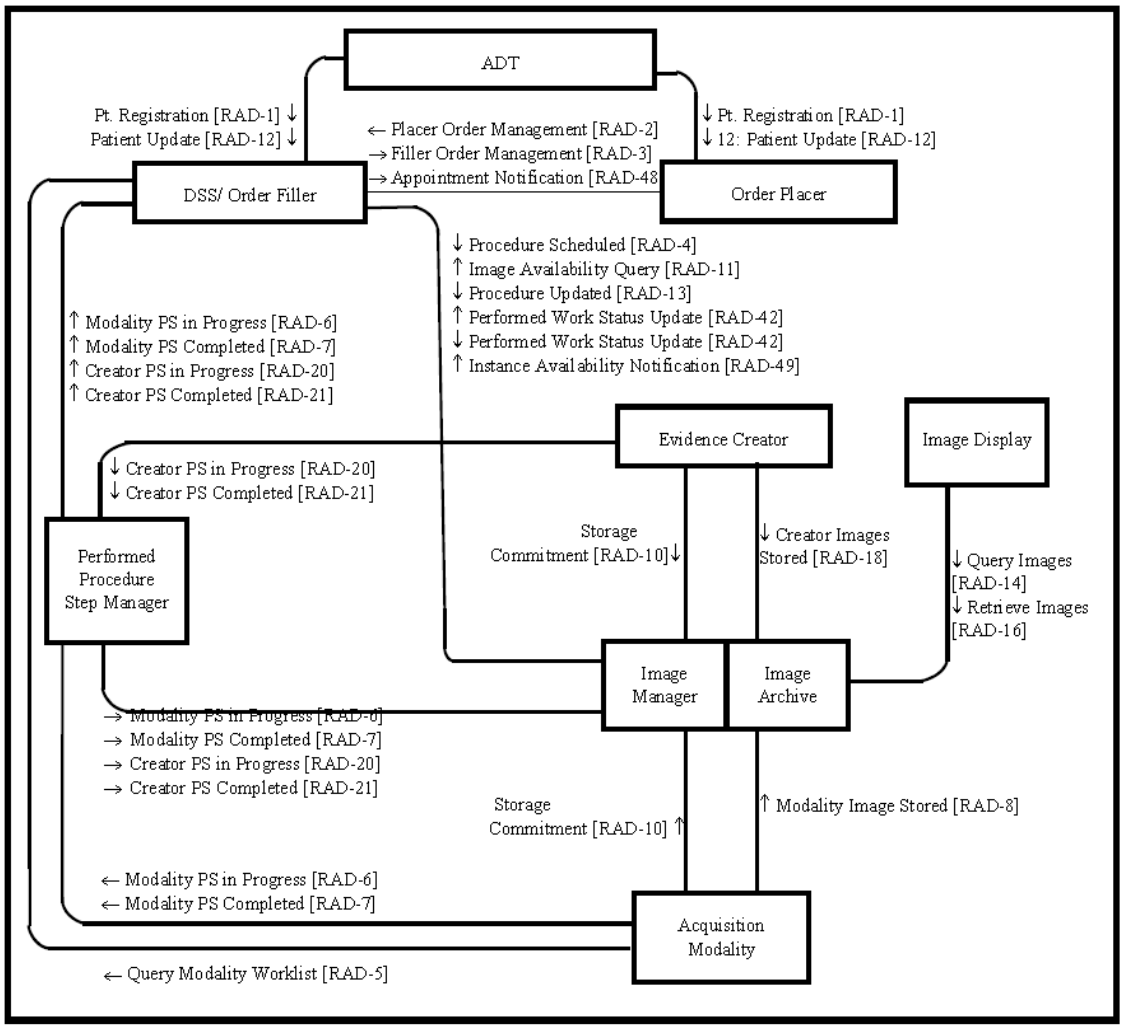

Figure 2: Scheduled Workflow Diagram

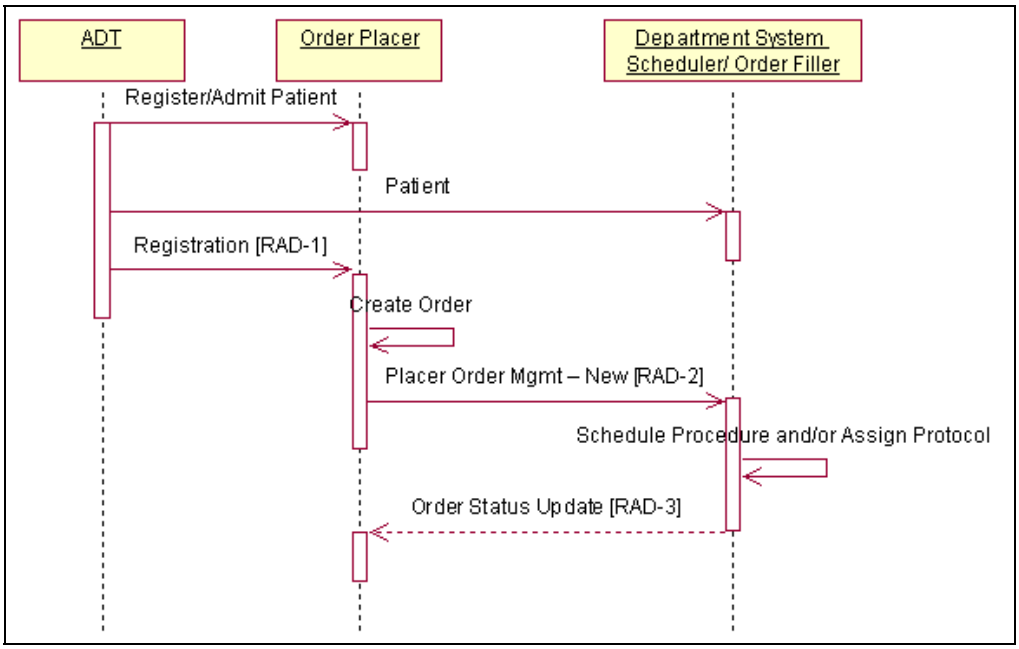

Figure 3: Part of SWF

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

5.1.1 BPEL Implementation

In this step, we derive a BPEL process specification from the *Patient Registration* activity diagram. The BPEL specification contains definitions of types, variables, messages, and correlationSets. Also, business partners and a process using basic and structured activities are defined. The WSDL file contains a portType and a serviceLinkType *Section* to define the web services. Finally, compensation activities are provided using scopes and security issues are outlined. Clearly, to start modeling BPEL processes requires a significant effort. Every part of the process model should be analyzed and pre-designed with a predefined structure. We apply the main steps of BPEL development procedure, and then we represent BPEL features in a test-bed to assess them in accordance with the evaluation performed earlier using our framework.

First, the partners section that covers the implementing application, the supported serviceLinkType, the role in the process as well as messages, and data types should be carefully developed. The next step is the development of the central process which harmonizes the flow of interaction in the whole orchestrated process. To complete the modeling process, transactions, security, error handling, and compensation features are applied. (Figure 4, 5, and 6) represent portions of the coding that has been done for the SWF process and (Figure 7) illustrates the model's graphical view.

With respect to modeling, we faced all challenges that we expected and mentioned in the framework based evaluation. The code sections of models are long and not user readable. They do not help in the presentation of security measures from a modeling perspective. The tight relationship between BPEL's graphical notation and its background coding makes it difficult to optimize representational detail level for training purposes or management use.

```
<message name="RAD-2">
   <part name="OrderInfoTypoe" type="xsd:string"/>
</message>
<message name="RAD-4">
   <part name="ScheduleInfoType" type="xsd:string"/>
</message>
<message name="RAD-6">
   <part name="ModalityProcedureProgressType" type="xsd:string"/>
</message>
<message name="RAD-7">
   <part name="ModalityProcedureCompletedType" type="xsd:string"/>
</message>
<message name="RAD-11">
   <part name="ImageAvailabilityQueryType" type="xsd:string"/>
</message>
<message name="ImageAvailability">
   <part name="ImageAvailabilityType" type="xsd:string"/>
</message>
<message name="RAD-3">
   <part name="OrderStatusType" type="xsd:string"/>
</message>
```

```
<portType name="OrderFillerPortType">
   <operation name="ProcedureScheduled">
      <input name="input1" message="tns:RAD-2"/>
      <output name="output1" message="tns:RAD-4"/>
   </operation>
   <operation name="ModalityProgress">
      <input name="input2" message="tns:RAD-6"/>
   </operation>
   <operation name="ModalityCompletionOperation">
      <input name="input3" message="tns:RAD-7"/>
   </operation>
   <operation name="QueryImageAvailability">
      <input name="input4" message="tns:ImageAvailability"/>
      <output name="output3" message="tns:RAD-11"/>
   </operation>
   <operation name="UpdateOrderStatus">
      <input name="input5"/>
   </operation>
   <operation name="ScheduleProcedure">
      <input name="input6" message="tns:RAD-2"/>
   </operation>
</portType>
```

Figure 4: Message and port type declarations

```
<partnerLinks>
   <partnerLink name="OrderFillerPL" xmlns:tns="..." partnerLinkType="tns:partnerlinktypeOrderFiller" partnerRole="OrderFillerPortTypeRole"/>
   <partnerLink name="OrderPlacerPL" xmlns:tns="..." partnerLinkType="tns:OrderPlacer" partnerRole="OrderPlacerPortTypeRole"/>
   <partnerLink name="Secretary" xmlns:tns="..." partnerLinkType="tns:ADT" myRole="ADTRegistraror"/>
</partnerLinks>
```

Figure 5: declarations of partners and partnerlinks

```
<sequence>
      <receive name="RegistrationInitiation" createInstance="yes" partnerLink="Secretary" operation="PatientRegistration"
xmlns:tns="http://j2ee.netbeans.org/wsdl/ADT" portType="tns:ADTPortType" variable="PatientRegistrationRequestreceived"/>
      <flow name="Flow1">
         <if name="If1" xmlns:tns="http://j2ee.netbeans.org/wsdl/OrderPlacer">
            .
            .
            .
            .
            .
         </copy>
      </assign>
      <reply name="RegistrationCompleted" partnerLink="Secretary" operation="PatientRegistration" xmlns:tns="http://j2ee.netbeans.org/wsdl/ADT"
portType="tns:ADTPortType" variable="PatientRegistrationOut"/>
   </sequence>
```

Figure 6: Central process

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

One of the main issues that came to our attention is model reusability. BPEL models are not reusable in a way that for a small change in the process, the major part of the process will be affected and sometimes it would be necessary to remodel the whole process. This issue is even more important for processes where additions are frequent. BPEL is less capable of adopting new processes or services after the modeling process is finalized. It is worth mentioning that we did not use any extensions to BPEL in order to measure the language's genuine capacities and features.

5.1.2 WS-CDL implementation

(Figure 8) shows the main choreography part of the WS-CDL document for our SWF example. Although WS-CDL provides more features to support healthcare requirements, the long WS-CDL documents are rather hard to analyze and understand for non-professional model users, as can be seen in the figure. Moreover, WS-CDL sometimes fails to provide the description of workflows if the proper WSDL files are not available. Finally, the lack of a standard integrated graphical notation makes the understanding of the described choreographies time consuming.

WS-CDL can be seen as a programming language with limited capabilities. It obviously lacks some of the features of a modeling language (i.e., graphical notation), and it has some extra features (i.e., limited execution) that separate it from other description languages.

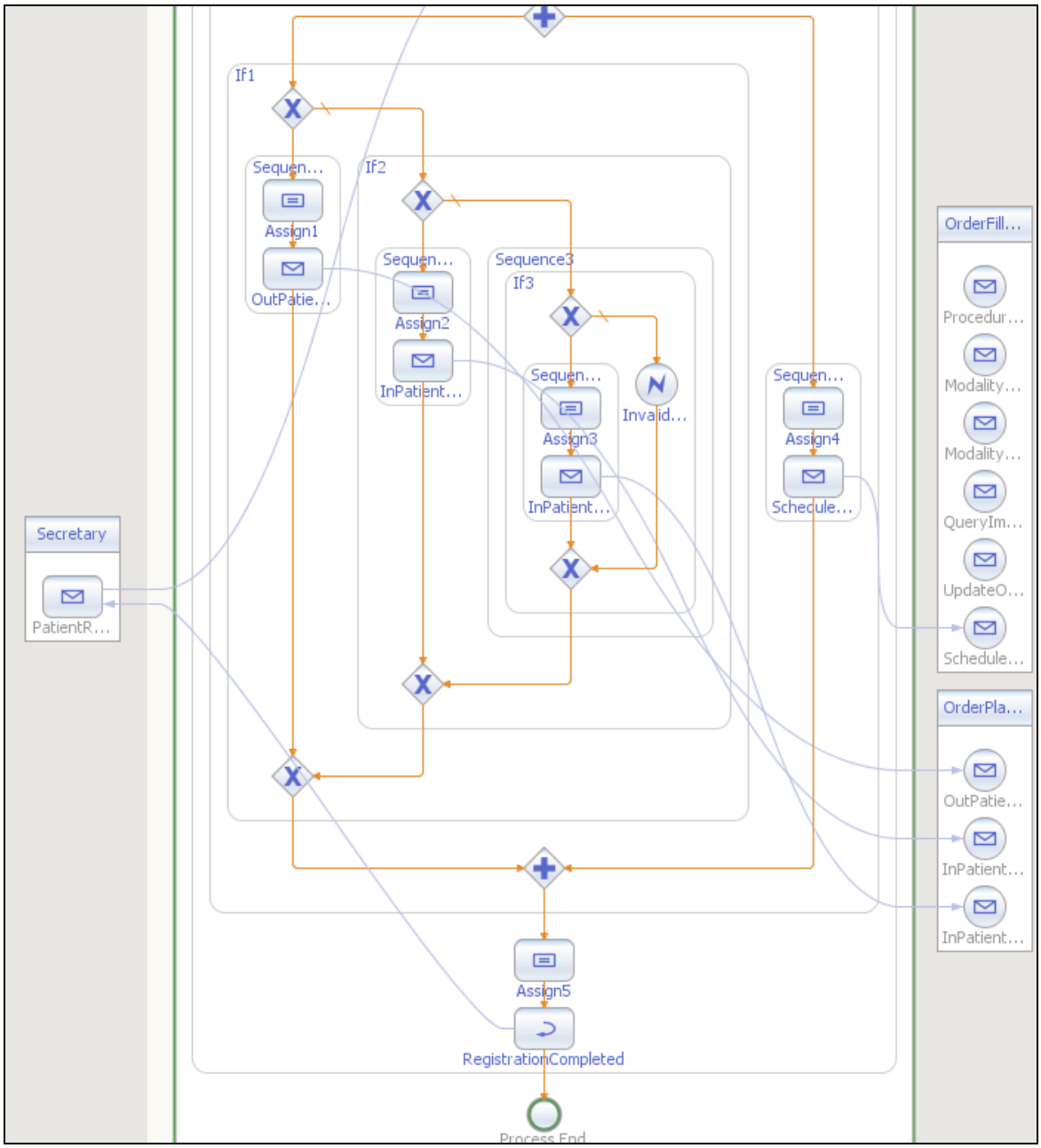

Figure 7: ADT sub-Process and exception handling in BPEL

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

6 Implications for Healthcare Business Process Modeling

In the previous sections we developed choreography and orchestration representations of our test case, using BPEL and WS-CDL, and identified advantages and disadvantages of both languages for modeling healthcare processes. We found that neither of the two languages individually provides the full functionality to satisfy all the healthcare requirements. (Table 3) represents a global comparison between languages. Based on the findings, each language has advantages that can compensate for shortcomings of the other language in modeling specific processes. As mentioned earlier, it is possible to combine two modeling languages to overcome one language's limitations by utilizing the combined advantages of both. This is what we present in this section.

In order to model a system using two different languages (or methodologies), we first need to categorize the processes based on their suitability for modeling in each language (or methodology). The following criteria should help in the task.

Intra-departmental or inter-departmental relations: Intra-departmental processes usually include a tight relationship between processes. There is often a major process which handles and manages the whole department's process. In this case, orchestration would be a better choice since creating standalone processes is difficult and in most cases these processes are dependent on each other. On the other hand, different departments have separate defined tasks that should not overlap. They usually operate as a standalone unit and there is no central dominant process over their operation. In this case, different departments should be modeled as choreographies.

Privacy: If privacy is an important concern then orchestrations would not be a good choice. Orchestrated processes share their procedures with the central process which exposes the process to all participant processes while choreography provides a level of encapsulation for the process so the process remains private at the departmental level.

```
<choreography name="SWFChoreography" root="true">
  <relationship type="tns:ADTOrderPlacer"/>
  <variableDefinitions>
    <variable name="ADTOrderPlacerC"
        channelType="tns:ADTOrderPlacerChannel"
        roleTypes="tns:RequesterRole tns:ProviderRole">
    </variable>
    <variable name="RAD1"
        informationType="tns:RAD1Type"
        roleTypes="tns:RequesterRole tns:ProviderRole">
    </variable>
    <variable name="RAD3"
        informationType="tns:RAD3Type"
        roleTypes="tns:RequesterRole tns:ProviderRole">
    </variable>
    <variable name="fault"
        informationType="tns:FaultType"
        roleTypes="tns:RequesterRole tns:ProviderRole">
    </variable>
  </variableDefinitions>

  <choice>
    <sequence>
       <interaction name="Registeration" operation="Register"
         channelVariable="tns:ADTOrderPlacerC">
         <participate relationshipType="tns:ADTOrderPlacer"
             fromRoleTypeRef="tns:ReqRole" toRoleTypeRef="tns:ProRole"/>
         <exchange name="RegisterationReq" informationType="tns:RegistrationReqType"
             action="request">
           <send variable="cdl:getVariable('ID','','')"/>
           <receive variable="cdl:getVariable('ID','','')"/>
         </exchange>
         <exchange name="RegistrationRes" informationType="tns:RegistrationResType"
             ...
         </exchange>
         <exchange name="UpdateResponse" informationType="tns:UpdateResponseType"
          action="respond">
          <send variable="cdl:getVariable('UpdateResponse','','')"/>
          <receive variable="cdl:getVariable('UpdateResponse','','')"/>
         </exchange>
         <exchange name="UpdateFault" informationType="tns:UpdateFaultType" action="
          respond" faultName="InvalidInfoFault">
          <send variable="cdl:getVariable('faultResponse','','')"/>
          <receive variable="cdl:getVariable('faultResponse','','')"/>
         </exchange>
       </interaction>
    </sequence>
  </choice>
</choreography>
```

Figure 8: Selection of a WS-CDL description

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 4 / ISSUE 2 / AUGUST 2009 / 1-19

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

Table 3: Framework-based comparison of BPMN, BPEL and WS-CDL

	BPMN	BPEL	WS-CDL
Security & Privacy	-	-	-
Pattern Support	17	10	13
Ontological Completeness	16	12	16
Extendibility	+	+	+
Notations	+	+/-	+/-
Modularity	+	-	+
Level of Detail	+/-	+/-	-
Exception Handling	+	+	+

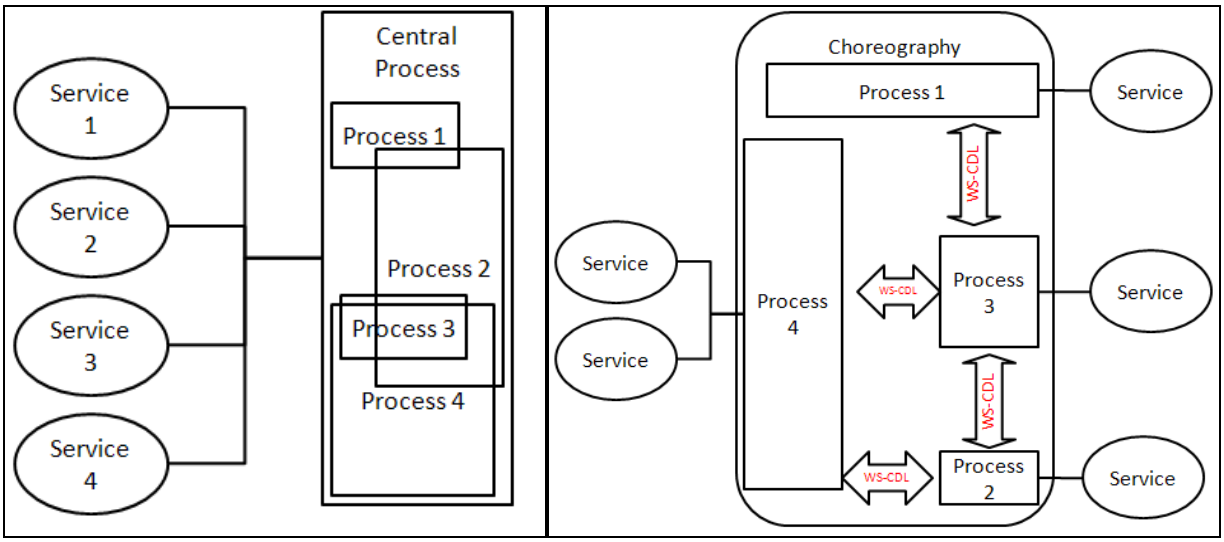

Figure 9: Linear model representation and integrated BPEL+WS-CDL model

Purpose of modeling: Considering there is no standard graphical choreography language, if the model is aimed to support training or managerial decisions, BPEL would be a better choice. However, as one of the discussed problems of BPEL is the high level of detail, if the above mentioned issues are the only reason for developing the models, graphical process modeling notations such as BPMN are better.

Once all processes are chosen, modeling can be done in either top-down or bottom-up approaches considering that the upper level will always be choreography. As we mentioned earlier in this section, the resulting model is partially graphical at the departmental level. Because the inter-departmental interactions are not included in each unit's model, models are smaller and less detailed, so BPEL models could be more understandable. Moreover, a combination of choreography and orchestration can solve one of the important concerns in the IT world which is reusability. Reusability is of major importance in complex business sectors such as healthcare due to their dynamicity, frequent change, large extent of process integration, and standardization. By developing reusable services and process models, modeling web service based processes can be done in a semi-automatic manner.

One of the major issues with BPEL is its lack of modularity. Lack of reusability always accompanies lack of modularity. Since BPEL processes cannot be divided into or composed from standalone processes, literally, all processes should be designed in a linear fashion (Figure 9). As a result, addition, removal, or alternation of even small sections of processes forces a large change on the involved processes. With the help of choreography, here WS-CDL, the BPEL processes can be divided into reasonably smaller parts, and then WS-CDL provides a means of interaction between the parts (Figure 9). Due to the separation of standalone processes, they are less resistant to change since most alternations will be made on the target BPEL standalone process and its relevant WS-CDL file. Additionally, each standalone process can be used in further modeling without any dramatic change. As a result, libraries of models will be available to modelers, and the semi-automatic process for defining web services workflows using web service based composition becomes possible.

In light of all the abovementioned benefits that come from combining the two methodologies, there are still some drawbacks remaining. The final model has within it each language's embedded problems. The codes are still long and detailed, so they can be used mostly for development purposes. Besides, mastery of two different methodologies is required for model creation and interpretation.

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

7 Conclusion

Many research initiatives have focused on studying the ambiguities and difficulties of normalizing models and modeling languages with the aim of improving our ability to design information and communication technologies. Since the specifications of various business sectors and the inherent variety of their processes impose the utilization of a subset of modeling features, modeling languages should be evaluated separately for each sector.

This paper presents our research on evaluating the suitability of service-based process modeling languages for healthcare, taking into account the specificities of the healthcare sector as well as its actors in order to enhance business process modeling and improve the design of information and communication technologies to support complex domains such as healthcare.

The healthcare environment is a challenging area for both design and development. The complexity of healthcare processes, the diversity of actors and healthcare's special needs such as security and privacy are the source of those challenges. Although the challenges have been observed by healthcare system designers and developers, there has been a little effort to identify them systematically. We identified challenges associated with modeling healthcare processes as groundwork for identifying the possible solutions to the challenges.

We introduced a framework for evaluating business process modeling languages in healthcare, defined a set of requirements for healthcare process models, and performed a case study on a complex IHE transaction using BPEL and WS-CDL. So far, results show that none of the studied languages, individually, satisfies healthcare modeling requirements. Basically, considering the advantages and disadvantages of each language, any of these languages is useful for different modeling purposes, yet neither of them is completely suitable for non-modeler target users. Embedded features of both languages are not enough for modeling complex processes, so extensions to existing languages are necessary in order to generate models that can suit healthcare modeling requirements. However, extensions raise more problems in the modeling and model use procedures. As another solution to overcome challenges associated with modeling healthcare, a combination of languages can be utilized. The combination of modeling languages increases the level of satisfaction for the healthcare modeling requirements, and also solves some problems that result from using a single modeling language. The most important problems solved by combining the two languages are reusability some process privacy issues. The final model, created by combination of the two languages, still lacks a complete graphical representation. However, we still recommend it for modeling healthcare processes because it addresses some fundamental healthcare modeling requirements. Healthcare is our domain of focus, but since most complex systems share a set of similar requirements and specifications, findings of this paper can be applied in other complex domains.

As we have illustrated in this paper, modeling languages are not complete. They only implement some of the features required in a process model. Limitations of this research are that we have only tested our framework using one evaluation case (SWF). Future research will involve further evaluation of conceptual cases (i.e., other IHE cases) as well as evaluation in actual clinical environments. Other future research will investigate the integration of modeling languages to overcome their individual limitations, and also combine their advantages. As was introduced in section 6, combining two modeling languages can satisfy some of the requirements of complex healthcare processes discussed in this paper.

Websites List

Site 1: Sparx Systems
http://www.sparxsystems.com.au

Site 2: Workflow Patterns Initiative
http://www.workflowpatterns.com

Site 3: Integrating Health Enterprise (IHE)
http://www.ihe.net

References

[1] R. Anzbock and S. Dustdar, "Modeling medical e-services," in Lecture Notes in Computer Science - Business Process Modeling , vol. 3080, Anonymous Heidelberg: Springer Berlin, 2004, pp. 49-65.
[2] R. Anzbock and S. Dustdar, Modeling and implementing medical e-services, Technical University of Vienna, Austria, 2004.
[3] D. Avison and T. Young, Time to rethink health care and ICT? Communications of the ACM. pp. 50-56, 2007.
[4] A. Barros, M. Dumas and P. Oaks. (2005) A Critical Overview of the Web Services Choreography Description Language (WS-CDL) BPTrends [Online]. Available: http://www.bptrends.com.
[5] R. Cover. (2008) "Business Process Execution Language for Web Services (BPEL4WS)". [Online]. Available: http://xml.coverpages.org/bpel4ws.html.

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

[6] R. Cover. (2003) "Business Process Modeling Language". [Online]. Available: http://xml.coverpages.org/bpml.html.
[7] F. Curbera, R. Khalaf, F. Leymann and S. Weerawarana, Exception handling in the BPEL4WS language," in BPM 2003. 2003, pp. 276.
[8] G. Decker and J. M. Zaha, Pattern-based evaluation of WS-CDL, Queensland University, Australia, [Online] Available: http://sky.fit.qut.edu.au/~dumas/LetsDance/WSCDLEval.pdf.
[9] S. W. Fraser and T. Greenhalgh, Complexity science: Coping with complexity: educating for capability. BMJ. vol. 323, pp. 799-803, 2001.
[10] L. Fredlund. Implementing WS-CDL. In *Procedings* of the II Jornadas Científico-Técnicas en Servicios Web (JSWEB 2006), pages 107-113, Santiago de Compostela, Spain, November 2006,
[11] A. Gemino and Y. Wand. A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering vol. 9, no.4, pp. 248-260, 2004.
[12] P. Green, M. Rosemann, M. Indulska and C. Manning, Candidate interoperability standards: An ontological overlap analysis. Data & Knowledge Engineering. vol. 62, pp. 274-291, 2007.
[13] V. Gruhn and R. Laue, Complexity metrics for business process models," in 9th International Conference on Business Information Systems. 2006, pp. 1.
[14] M. Hafner and R. Breu, Realizing Model Driven Security for Inter-organizational Workflows with WS-CDL and UML 2.0. Model Driven Engineering Languages and Systems. vol. 3713, pp. 39-53, 2005.
[15] Integrating Health Enterprise (IHE). (2007). "IHE Technical Framework, Volume I, Integration Profiles". [Online]. Available: http://www.ihe.net/Technical_Framework/upload/ihe_tf_rev8.pdf
[16] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon and C. Barreto, "Web Services Choreography Description Language Version 1.0", vol. 2008, November, 2005.
[17] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. V. Riegen, P. Schmidt and I. Trickovic. (2005) "WS-BPEL Extension for Sub-processes – BPEL-SPE,IBM & SAP". [Online]. Available: http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-bpelsubproc/ws-bpelsubproc.pdf
[18] M. Laguna and J. Marklund, Business Process Modeling, Simulation, and Design. USA: Pearson Printice Hall, 2004.
[19] R. Lu and S. Sadiq, A survey of comparative business process modeling," in BIS 2007. 2007, pp. 82.
[20] W. Luo and Y. A. Tung, A framework for selecting business process modeling methods. Industrial Management & Data Systems. vol. 99, pp. 312-319, 1999.
[21] J. Mendling and M. Hafner, From WS-CDL Choreography to BPEL Process Orchestration. Journal of Enterprise Information Management. vol. 21, pp. 525-542, 2008.
[22] J. Mendling, H. A. Reigers and J. Cardoso, What makes process models understandable?" in Business Process Management. 2007, pp. 48.
[23] H. Mili, G. B. Jaoude, E. Lefebvre, G. Tremblay and A. Petrenko, Business process modeling languages: Sorting through the alphabet soup, UQAM, 2004.
[24] A. Nadalin, C. Kaler, R. Monzillo and B. Hallam-Baker, Web services security: SOAP message security 1.1, WSS: SOAP Message Security. [Online]. Available: http://www.OASIS-open.org.
[25] A. G. Nysetvold and J. Krogstie, Assessing business processing modeling languages using a generic quality framework," in Workshop on Exploring Modeling Methods in Systems Analysis. 2005, pp. 545.
[26] R. F. Paigea, J. S. Ostroffa and P. J. Brookeb, Principles for modeling language design. Information and Software Technology. vol. 42, pp. 665-675, 2000.
[27] P. E. Plesk and T. Greenhalgh, Complexity science: The challenge of complexity in health care. BMJ. vol. 323, pp. 625-628, 2001.
[28] P. E. Plesk and T. Wilson, Complexity science: Complexity, leadership, and management in healthcare organizations. BMJ. vol. 323, pp. 746-749, 2001.
[29] M. Rani, A. K. Chawla and S. Batra, Web service choreography description language (WS-CDL): Goals and benefits," in COIT. [Online]. Available: http://www.rimtengg.com/coit2007/proceedings/pdfs/49.pdf
[30] A. Rodriguez, E. Fernandez-Medina and M. Piattini, A BPMN Extension for the Modeling of Security Requirements in Business Processes. IEICE Transactions on Information and Systems. vol. E90-D, pp. 745-752, 2007.
[31] M. Rosemann, J. Recker, M. Indulska and P. Green, A study of the evaluation of the representational capabilities of process modeling grammars," in CAiSE, LNCS. 2006, pp. 447-461.
[32] D. Rossi and E. Turrini. What your next workflow language should look like, Presented at 2nd InternationalWorkshop on Coordination and Organization.
[33] W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede, N. Russell, H. M. W. Verbeek and P. Wohed, Life after BPEL?" in Formal Techniques for Computer Systems and Business Processes. 2005, pp. 35-50.
[34] W. M. P. Van Der Aalst. (2003, February). Don't go with hte flow: Web services composition standards exposed. IEEE Intelligent Systems. [Online]. Available: http://www.tm.tue.nl/it/research/patterns/ieeewebflow.pdf
[35] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski and A. Barros. Workflow patterns. Distributed and Parallel Databases vol. 14, no.1, pp. 5-51, 2003.
[36] W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede and P. Wohed. (2002) Pattern based analysis of BPML (and WSCI), Queensland University of Technology, QUT. [Online]. Available: http://xml.coverpages.org/Aalst-BPML.pdf
[37] Y. Wand and R. Weber, On the ontological expressiveness of information systems analysis and design grammars. Information Systems Journal. vol. 3, pp. 217-237, 1993.

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762009000200002

[38] W. Wang, H. Ding, J. Dong and C. Ren, A comparison of business process modeling methods," in Service Operations and Logistics, and Informatics. 2006, pp. 1136.
[39] R. Weber, Ontological Foundations of Information Systems. Melbourne, Australia: Coopers & Lybrand and the Accounting Association of Australia and New Zealand, 1997.
[40] P. Wohed, W. M. P. Van Der Aalst, M. Dumas, A. H. M. Ter Hofstede and N. Russell, "On the sustainability of BPMN for business process modeling," in Lecture Notes in Computer Science - Business Process Management , vol. 4102, Anonymous Heidelberg: Springer Berlin, 2006, pp. 161-176.
[41] P. Wohed, W. M. P. Van Der Aalst, M. Dumas and A. H. M. Ter Hofstede. (2002). Pattern based analysis of BPEL4WS, Queensland University of Technology, QUT. [Online]. Available: http://xml.coverpages.org/AalstBPEL4WS.pdf