Control-Flow Patterns in Converged Services

Andres Benavides, Gustavo Enriquez, Jesus David Ramirez, Cristhian Figueroa, Juan Carlos Corrales
Telematics Engineering Group
Universidad del Cauca
Popayán, Colombia
(andresbenavides, genriquez, jdramirez, cfigmart, jcorral)@unicauca.edu.co

***Abstract*—Nowadays, telecommunication-services providers have focused their interest in developing new converged services at low cost and fast time to market. For this purpose they have adopted composition approaches, which are based on the reuse of existing software components to create more complex services. However the composition of converged services has so far been done at code level which requires expert knowledge, makes the process tedious and complicated, and causes an increase in design and deployment time. This paper reviews the control-flow patterns supported in converged services, proposes a theoretical basis, and defines the most recurrent control-flow patterns present in these kinds of services.**

***Keywords*-Converged services; service composition; control-flow patterns.**

I. INTRODUCTION

Nowadays, telco service providers (hereinafter Telcos) are looking for new methods for service creation in order to react quickly and cost effectively to dynamic market conditions. Furthermore, convergence of networks, services and content is taking place at an increasing speed, and therefore Telcos are forced to reduce services time to market [1].

One of the methods adopted by Telcos to accelerate the time to market is called *composition*; which in Telco environments, is based on the reuse of existing software components in order to create converged services. These services integrate traditional telco services (e.g call forwarding, sms) with web services (e.g payment services, social services) and provide an added value to users willing to pay.

Therefore, Telcos have adopted telecommunications platforms like JAIN SLEE (JSLEE) [2], Sip Servlets [3], etc. to compose converged services keeping telco features (high throughput, low latency, high availability [4]) and additionally supporting web services invocation through specific adapters. These features have allowed the composition of a large amount of converged services; however, this process has been done at code level requiring an expert knowledge. Consequently, the composition becomes tedious and complicated, causing an increase in the service design and deployment time [5].

To address this issue, and in order to facilitate the creation and composition of services, some studies have focused on the separation of the service business logic and its implementation. For example [4], [5], [6] concentrated in the design of converged services using the Business Process Execution Language (BPEL) or the Java Bussines Process Management (jBPM) graphical tools. These services are then adapted to a JSLEE platform, and executed afterwards.

As can be seen, these works use well-known languages to create converged services and model features as execution order and data exchange, which defines the execution flow of these kinds of services. Moreover, these languages allow the use of recurrent structures known as patterns (e.g. split, merge, synchronization) which can be reused in the composition process. These patterns represent repetitive events within an execution flow and have been widely studied in Van der Aalst *et al.*'s [7] work. This study analyzes the execution flow from two perspectives: I) control-flow patterns (CFP) [8] related with services execution order; and II) data patterns [9] used in data flow to represent the information transfer among services. The previously named works have addressed service composition emphasizing the use of model languages and reusable structures, however these approaches have been found to present drawbacks due to the fact that patterns perspectives and typical composition languages, like BPEL and JPDL, were designed to model the execution flow of web processes, but not for converged services. Consequently telco-services execution features are left out.

Our work addresses the study of CFP support in converged services, in order to create a theoretical basis which allows the standardization of processes related with the creation and composition. The use of CFP as a means of categorizing recurrent solutions in the converged services field, brings advantages as: abstraction and description, in a concrete form, of the recurrent structures (e.g. splits, merges, loops) in specific contexts; fulfillment of composition requirements (e.g. define the correct order of services) and reduction of the modeling work as well as development time and cost, through the use of recurrent solutions. Therefore, it is a new insight within the processes related with converged services which allows understanding and systematizing the workflow perspective in this field; thus, we define a list of most recurrent CFP present in these kinds of services in order to take advantage of the impact of the workflow patterns [8].

This paper is organized as follows: Section II describes a method for detecting CFP in converged services; Section III shows the results obtained after patterns detection; and finally, Section IV presents the conclusions.

II. Methodology

The methodology used in this paper focuses on CFP detection in converged services and consists of two main parts: Service Catalog and CFP Detection. In the first one, we classify the services present in a convergent environment defining an architecture for services classification and a list of services which are then represented by a formal model. In the second one, we adapt an algoritm to detect substructures (patterns) in the entire service catalog, allowing to define a set of CFP applicable to the converged services domain. These main parts are described in more detail in the following sub-sections.

A. Service Catalog

We define a service catalog in order to classify a set of services present in a converged environment, the catalog is based on a literature review of service delivery platforms (Mobicents [10], Rhino [11]); some service providers (Vodafone, Telefonica, AT&T), and organizations (ITU, 3GPP, ETSI-TISPAN, OneApi [12]).

1) Service Architecture: The Services Architecture proposed in this paper (Fig. 1), is based on Al-Begain *et al.* [13], which provides an ideal environment for telco and web services composition. This study proposes an architecture divided in five layers: network, control, services, integration and implementation. For the scope of our work, we focuse only on the service layer in which we define the following modules: Telco Services (TS), Web Services (WS), Converged Services (CS), Data access Services (DAS) and Security Services (SS).

- TS module includes traditional telco services known as Basic Services (BS) [14], [15] and complementary functionalities which are called Supplementary Services (SPS) [16]. Additionally, this module includes composite services as result of BS and SPS combination (e.g Call + Call Hold) [17].
- WS module contains traditional web services (TWS) and Web 2.0 services (WS2). The TWS, are static services with simple request-response mechanism; while the WS2, allows a dynamic and interactive knowledge creation on the internet [18]. Additionally, WS module includes composite web services which combine TWS and WS2.
- CS module contains converged services which are defined as services, applications, and content, provided by different networks through different user-terminals [19]. Based on this definition, this module contains services using telco and web capabilities. It is important to bear in mind that a convergent service must include at least a TS and WS.
- DAS and SS modules are transversely located in the architecture as they can be used by TS, WS and CS modules. DAS module allows services to access to certain data using different standards (e.g. Java Data Base Conectivity JDBC); and SS module guarantees that services are safely consumed by users, using protocols such as: Internet Protocol security (IPsec) [20], Secure Sockets Layer (SSL) [21], Secure HyperText Transfer Protocol (S-HTTP) [22], etc.

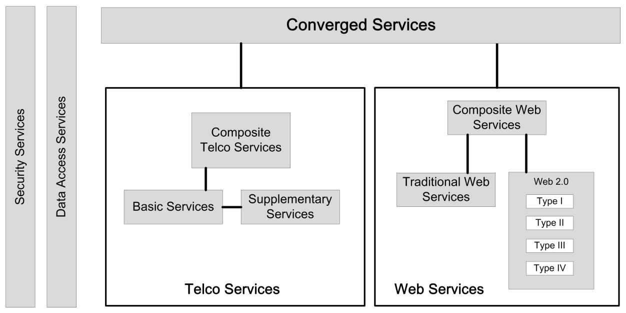


Figure 1. Service Architecture

2) List of Services: As presented at the beginning of this section, the list of services is based on the literature review of different organizations. In this list we show a set of 40 services (15 TC, 15 WS, 10 CS) which are part of a converged environment, however, in Table I due to the space restriction we only present some examples, which are listed according to the classification performed using the architecture shown in Fig. 1.

3) Formal Model: To describe the services contained in the catalog, we used two formal models: Petri Nets and Graphs. The first one, uses a set of places, transitions and edges to describe concurrent, asynchronous, distributed and nondeterministic systems [23]; and the second one, uses nodes and edges to represent and describe the existing relations between system objects [24]. In our work, Petri nets were used to provide a detailed and precise service description. This model allows the association between services of the catalog and CFP [8] which use the same representation. Additionally, we use the graph formal model to represent the services control-flow with a logical approach. With this model is possible to use gates like AND, OR or XOR to describe the structures that define the services control-flow. Fig. 2 and Fig. 3 (right) present the *Basic Call* service using Petri Nets and Graph model. In the first one, Petri Nets allow to represent states (ring, talking, busy recording) and possible events (invite, 200 Ok, busy, bye) of the service, while the second one, shows the service control-flow through logic gates: XOR-JOIN (XORJ) and XOR-SPLIT (XORS).

Table I
LIST OF SERVICES

Telco, Web and Converged Services		
Telco	BS	Call
		SMS
		Video Call
	SS	Call Hold
		Call Transfer
		Explicit Call Transfer
		Conference
		Advice of Charge
	Composite Telco Services	Call + Call-Hold
		Call + Call-Hold + SMS
		Call + Conference
Web	TRS	Payment
		Terminal Location
		Presence
		Terminal Status
		Device Capabilities
	WS2	Facebook Services
		Maps
		Skype Services
	Composite Web Services	Presence + Facebook
		Facebook + Call
		Travel Planning
Web+Telco	CS	Maps + Click to Dial
		Financial Instant Message
		Facebook Events Reminder

B. Control-Flow Patterns Detection

To detect CFP in converged services, a method composed of four main modules was used; this model is illustrated in Fig. 4.

- *Service Graph*: it contains a file that describes the service control-flow.
- *Query CFP*: they are the reference patterns used by the Pattern Detection module. These query patterns are stored in a database.
- *Pattern Detection*: it uses a patterns detection algorithm to define the number of patterns in the service graph.
- *List Of CFP*: represents the number of detected patterns in the service graph.

These modules are described in more detail on the following subsections.

1) Service Graph: Each service described in graph model is associated with a file description, in which the CFP detection is performed.

The file is divided into five sections, as shown in Fig. 3 (left). The first section of the file begins with an identifier (#graph0), the second section describes the number of graph nodes; the third section refers to the nodes labels, where E is an event and T is a task; the fourth section contains the number of edges (it should be at least one), and finally the fifth section shows the edge labels which represents the source and destination nodes, and defines the service control-flow [25].

2) Query CFP: Query CFP are the reference patterns used by the detection module to find recurrent structures in the graph service. In this paper we have defined a set of query CFP based on the evaluation performed in Wohed *et al.*'s [26] work, in which a study of graphical jBPM

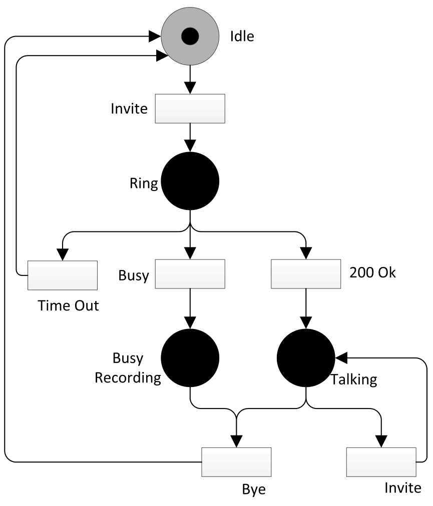
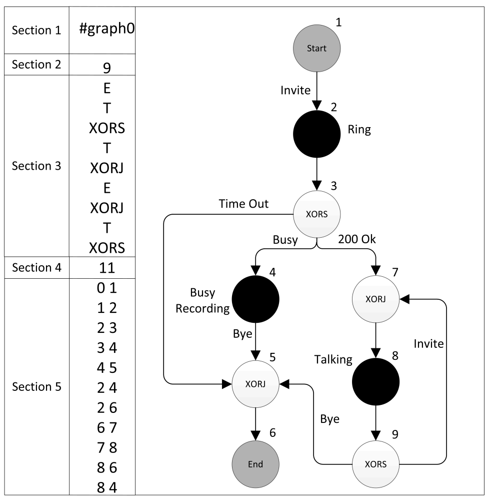


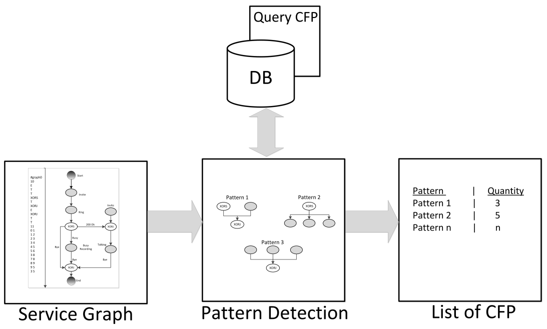
Figure 2. Basic Call Service using Petri Nets model



Section 1	#graph0
Section 2	9
Section 3	E T XORS T XORJ E XORJ T XORS
Section 4	11

Section 5	0 1
	1 2
	2 3
	3 4
	4 5
	2 4
	2 6
	6 7
	7 8
	8 6
	8 4

Figure 3. Basic Call Service using Graph model (right) and its description file (left)



Pattern	Quantity
Pattern 1	3
Pattern 2	5
Pattern n	n

Figure 4. Method to detect CFP

tool v. 3.14 and BPEL language v.1.1 was performed. This evaluation provided a set of 16 patterns directly supported by jBPM and 13 patterns directly supported by BPEL. In addition, we consider the composition and specializations relationships between those patterns [8]. The first one, exists when a pattern represents a more restricted form of another pattern, (e.g. *Multiple-Choice* pattern is a specialized form of *Parallel split* pattern) and the second one, combines two or more patterns (e.g. *Structured Loop* pattern is the *Exclusive Choice* and *Simple merge* patterns combination). Table II shows a subset of twenty-one query CFP, as result of the previous analysis. These set of query patterns are represented through logical gates. Fig. 5 shows some examples of query patterns taking as reference the Vander Aalst *et al.*'s work [8].

Due to the control-flow of some patterns can be similar, they may be associated to the same logical gate.

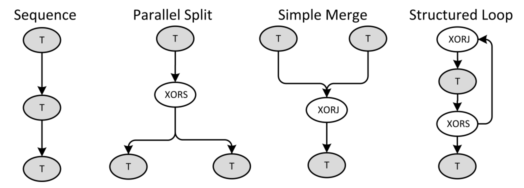


Figure 5. Query CFP represented through logical gates

Table II
QUERY CFP AND ITS ASSOCIATED LOGICAL GATES

CFP	Associated gate
Sequence	-
Simple merge	XORJ
Multi choice	ORS
Multi merge	ORJ
Parallel split	ANDS
Implicit termination	
Synchronization	ANDJ
Transient trigger	
Persistent trigger	
Exclusive Choice	XORS
Deferred choice	
Multiple instances without synchronization	
Structured synchronizing merge	ORS+ANDJ
Local syncrhonizing merge	
Generalized AND-join	ANDS + ANDJ
Critical Section	
Interleaved Routing	
Arbitrary Cycles	XORS + XORJ
Structured Loop	
Cancel Case	XORS or XORJ
Cancel Task	

3) Pattern Detection: CFP detection within the service graph (Section II-B1) was performed using the indexing graph method called GraphBlast [27] which allows searching for patterns within a graphs database. This method includes a subgraph isomorphism algorithm denominated VF2 [28] which executes comparisons between graphs, and determinate similarity among their nodes and edges structures (isomorphism), or whether a graph (sub-graph) is contained within another one.

4) List Of CFP: A list of CFP is the number of detected patterns in each service. For example, for the Basic Call service, as shown in Fig 3 (right) a set of two XORJ gates and two XORS gates were detected, which corresponds to the patterns: *simple merge, deferred choice, structured loop* and *cancel case.*

The result of the pattern detection is shown in Section III.

III. RESULTS

In this section, we present the result obtained after the CFP detection. Table III shows the set of patterns found in the entire list of services (Section II-A2).

The CFP detection is based on the results provided by the proposed method (Section II). Each service is analyzed individually and afterwards a general analysis is performed.

For example, the Facebook Events Reminder service, represented by a graph model as shown in Fig. 6, is a converged service which alerts the user when a Facebook event takes place. This service invokes a web service called Facebook User Events, which has the basic information of the user events (date, time, place). Then a Presence service is used in order to check and determinate the user availability; in this way, if the user is available, the service call him to give the event information by means of a voice recording, but if the user is not available, a SMS with the necessary information is sent.

The patterns detected in Facebook Events Reminder service are: *secuence* (nodes 1, 2 and 3), *deferred choice* (nodes 4 to 5 and 6), *cancel case* (nodes 7 to end) and *simple merge* (nodes 5-6 to 7). As can be seen the *secuence* pattern represents the invocation of Facebook-, HTTP-Manager- and Presence- Service, while the *deferred choice* pattern provides the ability to defer the moment of choice in the service based on user status (unavailable or available), additionaly, *simple merge* pattern provides a means of merging call and SMS services without synchronizing them; finally *cancel case* pattern ends the service.

Another example is the Telco service composed by : Call + CallHold + Call Transfer + Conference wich has five patterns where the most recurrent pattern is *simple merge*. Furthermore, according to the services features, it also has patterns like: *deferred choice, cancel case, exclusive choice* and *structured loop* which are shown in the Fig. 7. The service graph model for this service is not present in this paper due to space restriction.

The analysis in the previous two examples was performed to the entire 40 services presented in the catalog (Section II-A) and it is illustrated in Fig. 8.

This result is based on a recurrence rate analysis performed to the CFP found in the entire amount of services.

As can be seen, the most recurrent patterns are *simple merge* (P5), *defered choice* (P8), *parallel split* (P2) and

Table III
FOUND CFP IN THE SERVICE CATALOG

ID	Pattern
P1	Sequence
P2	Parallel split
P3	Synchronization
P4	Exclusive choice
P5	Simple merge
P6	Generalized And-join
P7	Multi-Choice
P8	Deferred Choice
P9	Cancel task
P10	Cancel case
P11	Structured loop

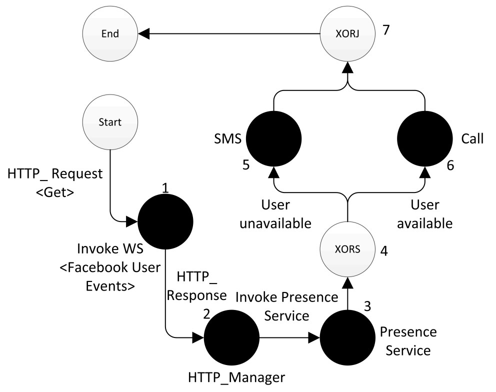


Figure 6. Graph model of Facebook Events Reminder service

cancel case (P9); while the the less recurrent patterns are *secuence* (P1) and *synchronization* (P3).

The result allows to identify the most recurrent CFP in converged services. In a practical environment, the CFP found in this paper can be implemented at code level once, and reuse them when needed; in this way it can be used as a code template in the processes of creation and composition of converged services. Thereby, the use of these recurrent structures (patterns) and a Service Logic Execution Environment like JSLEE, keeping telco features and supporting web services invocation, can provide a usable tool for developing new services in an easy way and with less time consumption.

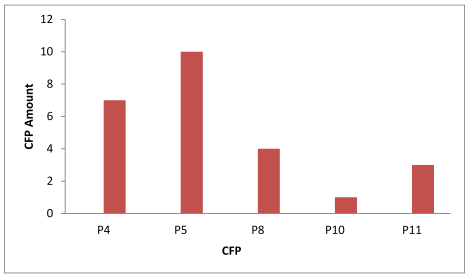


Figure 7. CFP in Call + Call Hold + Call Transfer + Conference service

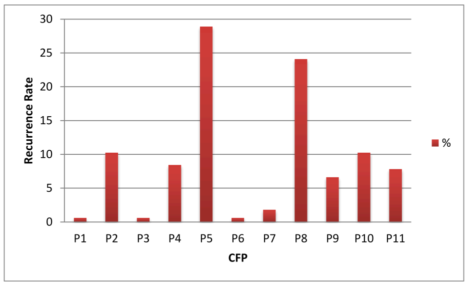


Figure 8. Recurrence rate of CFP

IV. CONCLUSION

This paper presented a study of the supported CFP in a converged services environment through the analysis of most recurrent CFP found in a set of services. The result obtained in this work serves as a theoretical basis to contribute to the standardization of practices related with creation and composition of converged services.

The CFP detection approach for converged services is based on two formal models: the first one is a graph model, which provides a clear description of the control-flow through logic gates; and the second one, is a Petri nets model which provides a more specific service description and allows the association between services in the catalog and CFP proposed in [8].

Additionally, a detection algorithm was adapted to find the number of CFP in each service, which results in a set of ten patterns applicable to the converged services domain. For this set of patterns, an analysis of recurrence rate was performed, concluding that the most common CFP within converged services are the *basic, stated based, cancellation* and *Structural* Patterns.

The next step in this work is a prototype development to experimentally evaluate the effectiveness of using CFP in the converged services environment in terms of time and ease. For this reason we are currently developing a graphical tool using GEF (Graphical Editing Framework) and JSLEE in order to create and compose converged services in an easier way. Fig. 9 shows the user interface within the Eclipse IDE; here services and patterns are illustrated as a blue building blocks, which can easily added to the workspace with a simple drag and drop functionality.

In addition, this study could be extended using data, resources and exception patterns defined in [7], in order to study other issues of converged services environment.

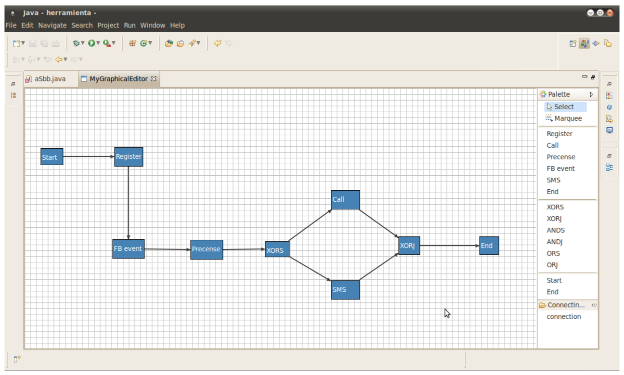


Figure 9. Graphical Composition Tool using GEF and JSLEE

V. ACKNOWLEDGMENT

The authors would like to thank University of Cauca and Telematics Engineering Group for supporting this research work.

REFERENCES

[1] P. Falcarin and C. Venezia, “Communication Web Services and JAIN-SLEE Integration Challenges,” *International Journal of Web Services Research*, vol. 5, no. 4, pp. 59–78, 2008.

[2] Community Development of java Technology Specifications - Oracle, “Jain Slee (JSLEE) v1.1,” June 2012. [Online]. Available: http://jcp.org/en/jsr/detail?id=240

[3] J. Deruelle, V. Ralev, and I. Ivanov, *JBoss Communications Platform 5.1 - SIP Servlets Server User Guide*, 2011.

[4] M. Femminella, E. Maccherani, and G. Reali, “A software architecture for simplifying the JSLEE service design and creation,” in *Software, Telecommunications and Computer Networks (SoftCOM)*, vol. 22, 2010.

[5] T. Eichelmann, W. Fuhrmann, U. Trick, and B. Ghita, “Enhanced concept of the TeamCom SCE for automated generated services based on JSLEE,” in *Eighth International Network Conference (INC)*, 2010.

[6] D. Zhu, Y. Zhang, B. Cheng, B. Wu, and J. Chen, “HSCEE : A Highly Flexible Environment for Hybrid Service Creation and Execution in Converged Networks,” *Journal of Convergence Information Technology*, vol. 6, no. 3, pp. 264–276, 2011.

[7] “Workflow Patterns Initiative,” June 2012. [Online]. Available: http://www.workflowpatterns.com/

[8] W. van der Aalst, A. Hofstede, N. Russell, and N. Mulyar, “Workflow Control-Flow Patterns, A Revised View,” in *BPM Center Report BPM-06-22*, 2006.

[9] W. Van der Aalst, N. Russell, A. Hofstede, and D. Edmond, “Workflow Data Patterns,” *QUT Technical report, FIT-TR-2004-01*, 2004.

[10] Open Source Cloud Communications Platform, “Mobicents,” June 2012. [Online]. Available: http://www.mobicents.org/slee/docs.html

[11] Open Cloud, “Rhino’s SIP resource adaptor (RA), services and components,” May 2012. [Online]. Available: https://developer.opencloud.com

[12] GSMA, “OneAPI,” June 2012. [Online]. Available: http://oneapi.aepona.com/

[13] K. Al-Begain, C. Balakrishna, L. A. Galindo, and D. Moro, *IMS : A Development and Deployment Perspective*, Chichester, West Sussex, U.K, 2009.

[14] ITU, “Definition of Teleservices, ITU-T Recomendation I.240,” 1989.

[15] ITU-T, “Definition of Bearer Services, ITU-T Recomendation I.230,” 1988.

[16] ITU, “Definition of Supplementary Services, ITU-T Recomendation I.250,” 1988.

[17] 3GPP, “3GPP TS 27.173 - Multimedia Telephony Service and supplementary services,” Tech. Rep. Release 9, 2010.

[18] S. Shang, E. Li, Y. Wu, and O. Hou, “Understanding Web 2.0 service models: A knowledge-creating perspective,” *Information & Management*, vol. 48, no. 4-5, pp. 178–184, 2011.

[19] F. R. Mejia, *Evolución de los centros de acceso público a las TIC*, serie cepa ed., 2009.

[20] P. Asadoorian, “An Introduction to IPsec,” May 2012. [Online]. Available: http://www.pauldotcom.com/IPSEC.pdf

[21] GNU, “The GNU Transport Layer Security Library,” 2012. [Online]. Available: http://www.gnu.org/software/gnutls/documentation.html

[22] Network Working Group, “HTTP Over TLS - RFC 2818,” May 2012. [Online]. Available: http://tools.ietf.org/html/rfc2818

[23] J. L. Peterson, *Petri net theory and the modeling of systems*, prentice-h ed., 1981.

[24] H. Bunke, “Graph Matching: Theoretical Foundations, Algorithms, and Applications,” pp. 82–88, 2000.

[25] D. Rivas, D. Corchuelo, C. Figueroa, J. Corrales, and R. Giugno, “Business Process Model Retrieval based on Graph Indexing Method,” *Lecture Notes in Business Information Processing*, vol. 66 Part 3, pp. 238–250, 2011.

[26] P. Wohed, B. Andersson, A. H. Hofstede, N. Russell, and W. Van der Aalst, “Patterns-based Evaluation of Open Source BPM Systems : The Cases of jBPM , OpenWFE , and Enhydra Shark,” *Information and Software Technology*, vol. 51, no. 8, pp. 1187–1216, 2009.

[27] A. Ferro, R. Giugno, M. Mongiovi, A. Pulvirenti, D. Skripin, and D. Shasha, “GraphBlast: multi-feature graphs database searching,” in *NETTAB 2007 Workshop*, 2007.

[28] L. P. Cordella, C. Sansone, and M. Vento., “Graph Isomorphism Algorithm for Matching Large Graphs,” *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.