POLITECNICO DI MILANO
Facolt`a di Ingegneria dell'Informazione

POLO REGIONALE DI COMO

Master of Science in Computer Engineering

Association, storage, and retrieval of semantically described Business Processes using ontology

Supervisor: Prof. MARCO BRAMBILLA

Master Graduation thesis by JALAL UDDIN AHAMMAD

STUDENT ID: 722869
Academic year – 2009/2010

Table of contents

List of Figures

Abstract

The primary objective of the semantic web technology is to provide knowledge oriented approach for describing things for machines so that they can communicate among themselves effectively without human intervention. However, this technology can be used for storage and retrieval of information semantically in a diversified field of study. Business processes which are described semantically can be discovered more effectively than the traditional way [1]. To describe the business process semantically, the concept of abstract business process has been introduced. Based on this concept, we implemented a prototype to store the abstract business processes in the ontology and proposed three different categories of possible associations between business processes and algorithms for finding them and updating them in the ontology. We also showed how to discover the business processes from the ontology based on the associations we proposed.

Chapter 1

Introduction

Process based organization management is not an unprecedented idea. The genesis of process oriented procedure has been evolved since 1980s and eventually shaped to Business Process Model (BPM) in the early 2000s. BPM is the current trend of describing and enacting internal and external procedures of an organization. A Process is defined as a set of activities connected with a set of flows in order to achieve a successful task of an organization. Since process modelling is a costly and time consuming effort, the process reusing concept came into being. Reusing the process models, those exist in an organization, is an important aspect of business process management system for constructing new processes i.e. processes exist serve as building blocks for the new processes. However, success of this paradigm is depends on how well users can locate the right process which would be relevant and effective to meet their need.

Reusing artefacts and sharing knowledge are key concepts in software development field nowadays. A good source of software models and code repositories, which are the contributions of the skill developers, can sustain a lot in the development of the further software projects. From the broad point of view, software project repositories, which serve the storage of code diffusion and design accumulations from the multifarious sources, can be exploited for further use with ease and efficiency. Searching of code or design of software project is pretty simple and unequivocal, since they are stored in the repositories as they are searched for. Nonetheless, searching high level models -such as business process models- does not acquiesce the project repository searching mechanism which acts upon source code level.

Business process reuse is a utile idea in BPM. However, finding process similarities and discovery of the processes based on these similarities in the process model repository is a strenuous task to be accomplished with efficiency, found by the researchers. Consequently, many researchers contributed in inventing methods applicable to storing process models considering the ease of storage and discovery. Model repository, though, the concepts of model repository is unequivocal in all research outcomes, techniques available with different approaches and flavours are multifarious. RepoX, which developed in METEOR project, introduced XML based process repository that used query language based on SQL [10]. Liu et al introduced object oriented approach to store enterprise work-flows in order to manage process information, run-time information and to cater browse and modification facilities of enterprise work-flows [9]. Eyal et al proposed a visual query language to retrieve business processes, which are modelled in BPEL [9]. Similarity metrics were employed in the development of a framework for discovering work-flows and their relationships by Goderis et al [10]. Repository for integrated Process Management (IPM) is a business process models storage that provides functionalities like storing and retrieving business process, version and configuration management [6].

The above described papers introduced model repositories based on many approaches to store business processes with the purpose of discovering the processes later for any opportune use. Business processes which are described semantically can be discovered more effectively than the traditional way [1]. To describe the business process semantically, the concept of abstract business process has been employed from the publication too.

With the purpose of discovering the business processes from the repository with ease and efficiency, we exploit ontological approach, i.e. define business process in the form of concepts in the ontology, to describe the business process semantically in the notion of abstract business process. By storing the semantically described process, we also proof that a set of useful associations between the business processes can be described in the ontology in conceptual approach.

In our thesis, we use a modeller to describe a set of business processes. Then, we developed a prototype which can transform the business process files extracted from the business process modeller into the concepts in the form of ontology class to store in the ontology. Our application can update the relationships, which are defined in the publication, among the business processes in the ontology in an automatic fashion. Nevertheless, we emphasized on three different categories of associations: structural, content based and complete associations mainly in this thesis.

To update the associations in the form concepts in our business process ontology, we proposed a set of algorithms to search these three disparate associations exist among the business processes and immediately can update the ontology with these relationships. To validate the proposed algorithms, we implement them in our prototype, and then perform an experiment with twenty five different business processes; the result of which is explained in chapter five.

The organization of the thesis report is as follows. In chapter 2, the background of the thesis and works related to it are tersely described. The concepts of Business Process Management, Process Model Repository and Model searching works are the key topics in this section. BPM is not based on an unprecedented idea. Though, BPM did not come into being until early 2000s, the genesis of this idea towards process based organization came from TQM-Total Quality Management- that appeared in 1980s. To store and to share the process models defined in an organization, model repositories are used. A brief discussion about ontology models available nowadays depending on the functionalities and platforms is made under this topic. Since the searching criteria and mechanism is not standardized due to the incongruity exist among the business process ontologies in structure and format, we construed briefly the works done for searching different ontologies with distinctive approaches. In chapter 3, we presented the core work performed in our thesis work for BPM search in semantic repository. The overview of the approach for storage and discovery of abstract business processes from the publication is concisely expressed here, later we proposed our association categories of the business processes and their storage in the ontology; and the similarity algorithms for updating the associations in the ontology. Chapter 4 underscored the implementation aspects of the proposed thesis. In this section, we described the technology used, requirements of the system in terms of use case and activity diagram, and finally the design of the system using class diagram pithily. In order check the conformity of our thesis work and the implementation done, validation is performed and explained briefly in chapter 5. A handsome number of business processes have been taken to perform the validation of our work. The results are illustrated from general to specific point of view with the help of the graphical presentations.

Chapter 2

Background and related work

2.1 About BPM

Towards the process based organizational efficiency, numerous efforts have been done prior to BPM, acronym of Business Process Management; those led the BPM to coming into being. It was a cumbersome roadmap from TQM- Total Quality Management – to BPM that deserved strenuous efforts to make it happen. Total Quality Management (TQM), which was pioneered in 1980s, overtook by Business Process Reengineering (BPR) prompted by Hammer and Champy in early 1990s. The mixture of excellent successes and failures of BPR exited until late 90s. ERP systems were unable to resolve the process issues after being come into work after late 90s. In early 2000s, CRM systems were introduced, which were responsible for front office processes rather back office crucial processes. The next big thing evolved is BPM. To make the BMP sustainable and efficient, multitudinous works have being performed for several years. The following diagram shows the BPM Hype cycle, which lasted for the last 20 years, gives a terse scenario of the process cycle.

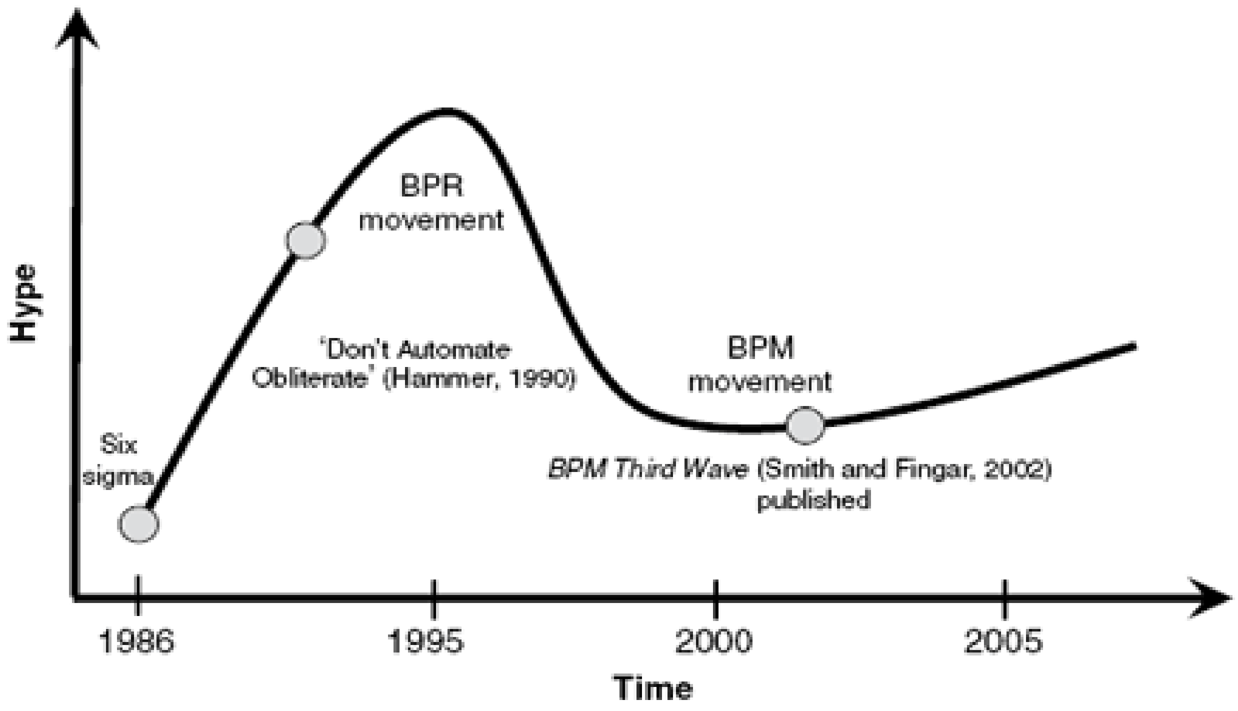

Fig-2(1) BPM Hype Cycle

The process awareness was inaugurated by the invention of Six Sigma in 1986. Business Process Reengineering movement was begun by the article “Don’t Automate, Obliterate” by Hammer and Champy in 1990. With the publication of the paper: BPM Third Wave by Smith and Fingar in 2002 the created a potential interest of the business process later time.

BPM is not simply modelling but it also involves the implementation and execution of the process models with sufficient analysis. “The Objectives of a BPM implementation range from the strategic goals of the organization through to the individual process goals. It is about the achieving the business outcomes or objectives”[3].

Generation of benefits for business is the core strategic goal of a business. And the implementation BPM should comply and substantiate this idea as well. The final product of a company is the integral outcome -a service, a product- of the sequence of activities performed. The activities involve actors, flows, rules, place where to execute the activities and, more complex issues like temporal aspects etc. Hence the organization of these activities is important. Business process is a tool to organise these activities effectively. Business process management is the management of business processes with the help of the information technology and the resources available to accomplish the goal assign for. For precisely, BPM can be defined as “Business process management include concepts, methods and techniques to support the design, administration, configuration, enactment, and analysis of business process”[12].

Coordination of enactments in the business process can be handled manually, automatically using software system known as Business process management system, or both. Business process management system is a software system that stores business process models with the purpose of being enacted with proper coordination of the software system. The storage or representation of the business processes in the software system can be textual, however, the flows of activity execution is either difficult to represent or time consuming to delve through. Hence, the graphical representation of business process is popular and easy to understand at a glance. Though there are many graphical tools available today, the aim of them is obviously the same of representing the business process. It is noticeable that graphical notations are used to represent the business processes and participants associated with activities regardless of the technical aspects of their realizations, i.e. the definition of business processes using graphical notations is independent from implementation strategies and platforms.

A business process can have several instances, and so the activities of the business process can have activity instances too. A business process a.k.a. business process model, comprises activities a.k.a. activity model, and constraints associated with each activity.

There are a plethora of BPM applications available nowadays that could be used to improve business process management of a company. An enterprise BPM application is a collection of business software that can be integrated in a company that does not have any BPM tool yet. Process specific BPM application tool is also available for the company which is reluctant to overheads of managing servers. So this application facilitate the use of business process with as simple as installing typical apps on the workstations. A company can also benefit employing hosted BPM application intending to outsource some of it's business processes like human resource or marketing.

2.2 Model repository

The definition of repository by Bernstein and Dayal is “a shared database of information on engineered artifacts which are produced or used by an enterprise”. According to them, a repository manager is tool that composed of functionalities like modelling, retrieving, and managing the objects in its repository. Inferring form the concept of database by Bernstein and Dayal, Injun Choi, Kwangmyeong Kim and Mookyung Jang defined process repository as “a repository that stores information of processes and provides functions to manage them”.

Introducing business process models in a company from the scratch is an onerous task to pull into functioning. A company can benefit from already developed models. It will cut off the extra efforts and money required to design new models from the beginning. It is worth mentioning the boon, when two companies, which have its own business processes and descriptions, merge into a single enterprise, is received from the existing models. A handful of researchers contributed in finding methods applicable to storing process models considering the ease of storage and discovery. Model repository, though, the concepts of model repository is unequivocal in all research outcomes there are plethora of techniques available with different approaches and flavours.

BPM model repositories are different from each other depending on the functionalities they offers [4]. In addition, the author worked with the general repositories with an extension for storing and managing business process models based on the concept describes by Bernstein and Dayal. The extended BPM model repository comprises process data model, process function model and process management model where process data model describes how model data can be stored internally in the BPM model repository – it composed of meta model, storage model, and index model- ;process function model composed of storage functions -functions includes create, update or delete of processes or part of the processes- , retrieval functions- functions includes navigate, query and search-, integration functions – function to integrate process repositories so as to communicate with external applications-;process management functions are process specific management functions like version management, configuration management, view management; and general repository management functions like access management, integrity management, notification management and context management.

The process Handbook project, which was established in 1991 by MIT, was developed to assist in designing and sharing of business processes. It culled business processes from several organizations and classified them depending on own criterion, criteria of the organizations, and compound one. It stores business processes in text format and does lack of a formal definition language. And therefore, the analysis and searching of the business process is done on the simple textual business process.

There are a multitude of process repositories based on XML available today. One of them is RepoX developed in METEOR project[10]. It has a query language based on SQL, but finding process similarity using this query language is not possible, or even did not keep in find while designing the query language.

Object oriented principle is introduced for storing enterprise work-flows by Liu et al in 1996. The aim was to manage process information, run-time information and to provide browse and editing facilities of enterprise work-flows.

Using the concept of software library form where people get help, process library can be introduced so that people can benefit in the process management area. “We define a process library as a collection of code assembled to perform a set of related coordinating and computing tasks”[5]. The main goal of this paper is to define a mechanism to provide an easier and flexible process management application using codes as building blocks.

Repository for integrated Process Management (IPM) is a business process models storage that provides functionalities like storing and retrieving business process, version and configuration management [6]. It is a business process management approach, that can integrate process using XML and can cater to design and to analyse business processes, and to manage business process knowledge for the entire process life cycle. Process analysis and optimization(PAO)-validation and performance estimation of new processes are carried out by this component using analysis and simulation-, Process knowledge management(PKM) --,Process modelling and integration(PMI)- component responsible for integration of process definitions and related data using Extensive mark-up language-, process automation and control(PAC)-process instantiations and execution are done by PAC-, and Process-oriented integration(POI), are various life cycle requirements of IPM. Each of these components interacts with each other to provide life cycle support for BPM. Moreover, five repositories namely Process Repository, Process Instance Repository, Process Knowledge Repository, Process Rule Repository, and Process Resource Repository are introduced in IPM to achieve the desire goal of the project.

2.3 Model search

Every major company can have a set invaluable business processes, which they need in its day to day activities. If a company is big enough, then it has to manage a large set of processes, which they may prefer to store in the repository with the intension of searching and reusing with ease and effectively. However, the searching criteria and mechanism is not standardized due to the incongruity exist among the business process repositories in structure and format. And therefore, a lot of searching algorithms based on the repositories are being proposed taking the complexity, approach and usage into account. The following discussions are about the approaches, we have got while doing our research for this thesis work.

Injun Choi et al introduced a process query language called IPM-PQL for IPM – an integrated process management approach for business process management [6]. IPM-PQL (IPM process query language) is based on XML-based process query language for effective retrieval of the information regarding processes required in the management of business processes. It also provides four types of searching options: whether process has attribute, process contains activity, process has sub processes or process is transitional.

Antoon Goderis et al proposed a query language called BP-QL for searching business processes [9]. BP-QL is developed to work on the process model constructed in Business Process Execution Language (BPEL). This paper depicts query language proposed as well the formal model on which the query processor is based on. They also presented implementation which is compatible with XML specification and web services.

Et al describes a workflow discovery tool, which was developed based on graph matching technique for ranking the workflows and validating them in a real workflow environment. In the requirement analysis phase, they conducted a survey to understand the criteria used for discovering workflows. In the workflow discovery process, they found from the survey that workflow structure would be an invaluable part. Hence they also proposed a workflow component to support the structural aspect in the discovery process. Finally they developed the workflow discovery tool to work in Taverna workbench.

Chapter 3

BPM search in semantic Repository

3.1 Overview of the approach

An abstract business process (ABP) is a representative of a set of equivalent business processes [1]. The equivalence of two business processes solely depends on the activities and the flows they share .Unlike concrete business process which is capable of being executed, abstract business processes are non executable business processes that composed of generic tasks -task descriptions shared by one or more concrete business processes, which can be semantically described and associated in the ontology to provide information about the concrete process activities such as ability of the processing units' performing activities, description of input and output data of the processing units. In order to utilize the semantic behaviour of these descriptions, concepts from ontology have been employed. Hence we can annotate the abstract business processes in terms of concepts and activities as task ontologies. To provide the relationship among activities; among flow operators of business processes and the concepts of ontologies, task descriptions are encoded in form of annotation. The concepts can be related in different binary relationships based on the semantics encoded by the ontology.

After defining the concepts in the ontology, we are able to update the associations among the concepts in the ontology. For example, if two concepts, which represent two abstract business processes, in the ontology have relationship such as equivalent, being part of or overlapped in terms of their task and flow ontologies, then we can update the ontology depending on the relationship we infer.

However, the relationships that we defined above did not consider all aspects such as structural point of view. The relationship up to now is defined solely depending on the task and flow ontologies of the concept in the ontology. So we were in need of looking for the relationships not only from the contents point of view, but also from the structural and complete point of view. Therefore, the extension regarding the association among abstract business processes has been proposed in this thesis.

The extension of the relationship among business processes is done by categorizing the association among business processes in three different types: i) Content based association ii) Structural association iii) Complete association. And each of these categories sub-divided into equivalent, part of, and overlap relationships. Consequently, the ontology, we are defining now, can update all the mentioned associations among abstract business processes in terms of concepts, and hence, it is possible to get a comprehensive searching mechanism for the discovery of the business processes as intended to.

3.2 Ontology for describing business process

Business process ontology denoted as θ_{BP} is an ontology that describes business processes. Now the question arises how we can describe business processes in ontology as we do not have any tool which can translate executable business process into ontology concepts. From the erudition we gleaned from research done, we exploited the conceptualization of the ontology to represent the abstract business processes, which is the non executable form of the business process intending to describe the business processes. A concept in the ontology is the representative class of the abstract business processes. The ontology is an explicit specification of a conceptualization [2]. An ontology, denoted by θ, can be well described by a collection of concepts, literally $\theta = \{c1, c2, c3, .., cn\}$. The concepts can be specific or general, i.e. specific concepts are linked to general concepts and these relationships are constructed using the sub concept relationships. The following conceptions about the business process ontology in this section are borrowed from the publication [1] which our thesis is based on.

Since a concept represents the abstract business processes in the ontology, a function getABP is proposed for a given concept c from the business process ontology θ_{BP} retrieving the abstract business process classified by c. The signature of the getABP is as follows.

$$\text{getABP} : \theta_{BP} \rightarrow \text{ABP}$$

To annotate the activities of a business process semantically, task ontology, denoted by θ_{task}, has been introduced. This ontology is responsible for acquiring information of the action carried out by of activities of different business processes with a same interest. In order to retrieve the task ontology i.e. annotation of the activities within a domain of interest, a function called task() is defined which takes an ACTIVITY of the business process as a parameter. The function can be stated as follows.

$$\text{task: ACTIVITY} \rightarrow \theta\text{task, where}$$

ACTIVITY represents the domain of business process activities.

An abstract business process abp is literally represented by the pair of terms within angle brackets as follows.

$$\langle T,CF \rangle\text{, where}$$

T is a collection of tasks comprises an abp : $T \subseteq \theta$task.

CF $\subseteq$ (T x OP) $\cup$ (OP x T) is the control flow relating the tasks in T.

Mapping of the tasks of the abstract business processes are performed with the help of two classes of functions, the domain of which are represented by MapEquiv and MapSpec[Publication]. The functions of MapEquiv are delegated to map the tasks of a supplied abstract business process to the tasks of another that execute exactly the same or equivalent tasks. If abp1 and apb2 are two abstract business processes, then the mapping function which maps the tasks of abp1 to those of abp2 can be expressed as f_{map}: abp1.T $\rightarrow$ abp2.T.

$f_{map} \in$ MapEquiv iff:

$$\forall\ t \in \text{abp1.T; task } (f_{map}(t)) \equiv \text{task } (t)$$

The construction of the abstract business processes abp corresponding to a busness process bp is done by employing a function denoted by abstractBP(). The function abstractBP can be stated as follows

$$\text{abstractBP: BP} \rightarrow \text{ABP}$$

BP denotes the domain of business processes and ABP the domain of abstract business processes.

Creating and populating the business process ontology in an automatic fashion proposed in [1] is used in this thesis. The idea of semantic annotations, which describe the tasks of constituent activities of a given set of business processes and definition concepts of the business process ontology, is also taken from it. The proposed technique of generating business process ontology given by the publication is as follows.

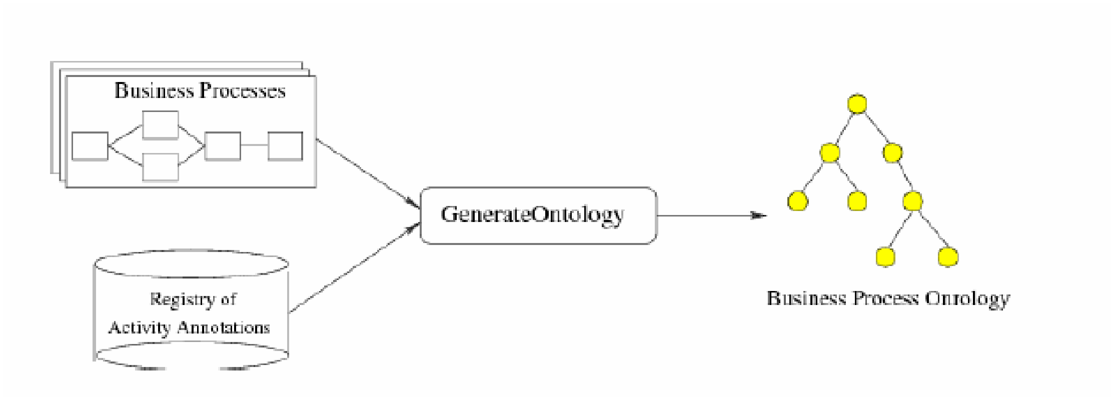

Fig-3(1): Generation of the business process ontology

The algorithm for generating business process ontology [1] is given below.

```
Algorithm: GenerateOntology
input : BP
output : $\theta_{BP}$
begin
1  for each bp $\in$ BP do
2      abp = abstract(abp)
3      if ($\exists$ c $\in \theta_{BP}$, abp = getAbstractBP(c))
4      then
5          addInstance(bp,c)
6      else
7          c := defineConcept(abp)
8          addInstance(bp,c)
9          deriveAndAssertProperties(c)
end
```

3.3 Concepts and relationships

The concepts, representative classes of abstract business processes, in the process ontology θ_{BP} are associated with each other using binary properties that encode relationship among abstract business processes. The identified three binary properties to encode process relationships are equivalence, overlap and part of. A prototype has also been developed based on these conceptual relationships. But later on, we observed that the associations could be extended considering other view points. An abstract business process, that represents the non executable form of concrete business process can be associated with another abstract business process based on three different approaches namely content based similarity, structural matching and complete associations. In a nutshell, we categorized the associations in three different ways and each of these approaches is divided into equivalence, overlap and part of relationships.

3.3.1 Content based Similarity

A concept in the ontology can be associated with another concept in terms contents such as the tasks belong to the abstract business processes and represented by the concept in the ontology. To update the ontology with this type relationship, an algorithm has been proposed which is discussed later in the algorithm section.

3.3.1.1 Content Equivalence among abstract business processes

Two abstract business processes are said to be content equivalent if the constituent tasks of both abstract processes are equivalent disregarding sequence of activities performed in the corresponding concrete business processes and the bindings of the tasks i.e. which control flow connects which tasks in abstract business process. If c1 and c2 are two concepts from the business process ontology $\theta_{BP,}$ the corresponding two abstract business processes abp1 and abp2, then, can be found using the function getABP i.e. abp1 = getABP(c1) and abp2 = getABP(c2) [1]. The concepts c1 and c2 are content equivalent, iff there exists relationships as follows.

$$abp1.T - abp2.T = \emptyset \text{ and } abp2.T - abp1.T = \emptyset,$$ where

abp1.T and abp2.T are the set of tasks of abstract business processes abp1 and abp2 respectively.

3.3.1.2 Content Partof relationship among abstract business processes

Two abstract business processes are associated with one another with the content part of relationship if the constituent tasks of one abstract process are contained in the another abstract business process where order, in which the tasks are connected with each other, and the bindings of the tasks, which control flow connects which tasks in each abstract business process, are not taken into consideration. For concepts c1 and c2 in the business process ontology $\theta_{BP,}$ if there are two corresponding abstract business processes abp1 and abp2, then the content partof relationship between these two concepts may exist. Concept c1 is a content part of concept c2, iff there exist relationships as follows.

i) $abp1.T \subset abp2.T$

3.3.1.3 Content Overlapping among abstract business processes

Two abstract business processes are associated with one another with the content overlap relationship if both of the abstract business processes have at least a task in common. If c1 and c2 are the concepts of the business process ontology θ_{BP}, the corresponding abstract business processes are abp1 and abp2, then the concepts c1 and c2 are said to be content overlapped, iff there exist a relationship as follows.

$$Abp1.T \cap abp2.T \neq \emptyset$$

3.3.2 Structural Similarity

To find the structural and the complete similarities between abstract business processes, we introduced hierarchical tasks concept. This concept emerged from the transitional behaviour of the tasks. For instance, every abstract business process has a certain set of tasks which are connected in a fashion that every task has a pre control flow -a flow coming from previous task-, a post control flow- a flow going out of the task-, and one or more next tasks- tasks those are to be executed next to this task in the concrete business process. To the best of our ken, the better way to explain the hierarchical structure of tasks of an abstract business process is graphical presentation of tasks. For better clarification let us take an abstract business process Enrolment, whose BPMN representation is the below.

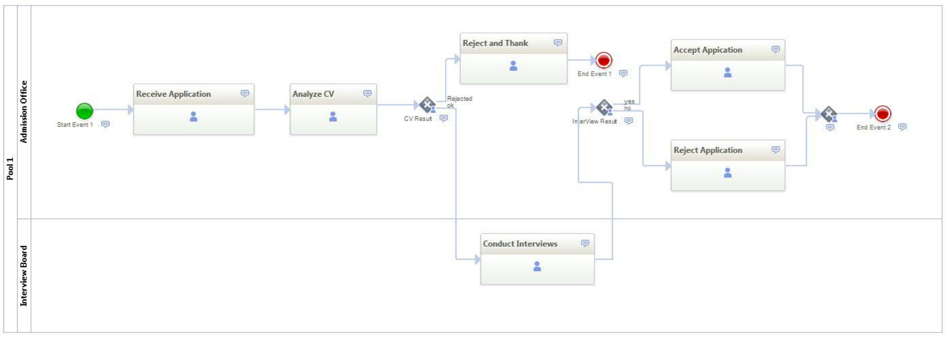

Fig-3(2): BPMN diagram of Enrolment process

A hierarchical tree diagram of the tasks of an abstract business process Enrolment can be shown as follows.

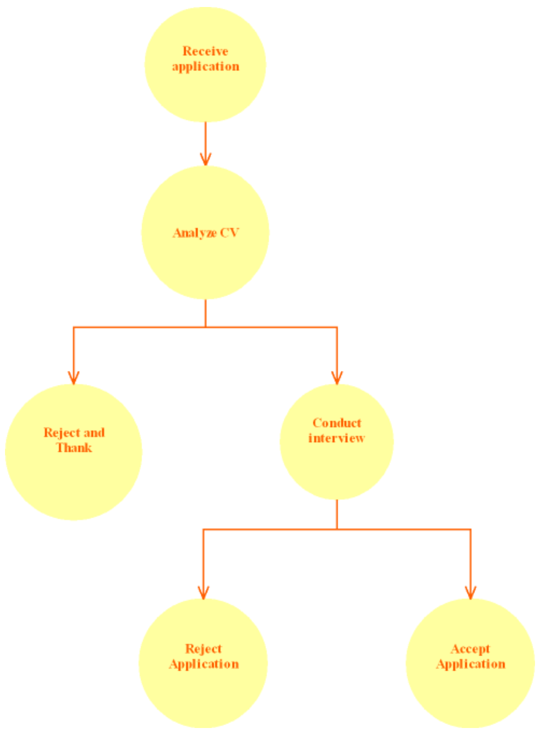

Fig-3(3): Graph for BPMN process Enrolment

After representing the abstract business process in hierarchical tree, we are in need of graph traversal algorithm to visit all the nodes, which represent the tasks of the abstract business process, to find the pre-control-flow, post-control-flow and the adjacent tasks of the each task. Since we are avoiding the cyclic nature of the business process, we introduced breadth first traversal algorithm here to complement our tasks. For the above ABP Enrolment, the breadth-first traversal begins with the task Receive application. Then it will add "analyse CV" in the queue to be visited next. A vertex, which represents task, is only added to the queue -data structure that holds the tasks those are not yet traversed-, if there exists a path between the previous vertex and the vertex to be added in queue. After the starting task being traversed, "Analyse CV" task is picked up and add its adjacent tasks in the queue and so forth. It is important to mention that the task that has been already traversed or has been added in the queue will not be added in the queue again, hence preclude the propagation of acyclic nature in the graph. While traversing each node, a hierarchical task – an object that can represent pre-control-flow, post-control-flow and the tasks that are in proximity to the task being traversed- is created for each and put in a linked list to retain the graphical presentation also in the hierarchical tasks list.

In order construct a complete graph to meet our need, every vertex must be traversed conceptually. While visiting each hierarchical task, each Htask acquires its pre-post flow operators and the adjacent tasks that are not yet visited. After being traversed all nodes, the hierarchical tasks' tree will be conceptually looked like the below.

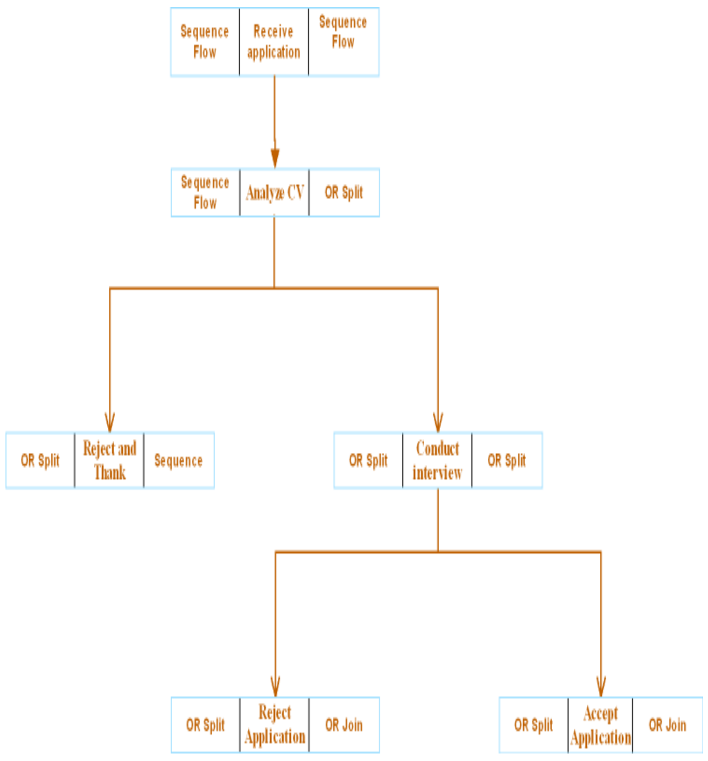

Fig-3(4): Graph for BPMN process Enrolment with pre and post flow operators

The comparison is done between two abstract business processes in terms of graphical representation by comparing the tasks and control flows in each level of both graphs. The level is the each hierarchical step in the abstract business process graph. For instance, "Receive application" task is in the first level of the Enrolment graph, while "Reject application" and "Accept application" are the tasks available in the last level of the hierarchical tree.

The maximum number of levels is the maximum number of acyclic transitions an abstract business process takes to describe the activities' of a concrete business process from start event to the end event. It can also be defined as the longest path a business process instance takes to travel from one activity instance to another towards the termination of the process instance without considering a cyclic transition in the path.

To get the level, an integer represents the number of transitions available in an abstract business process, a function getLevelof BP(abp) is introduced. The notation of this function can be well described as

$$\text{getLevelofABP(abp)}\ \theta_{BP}\text{: level}$$

3.3.2.1 Structural Equivalence of abstract business processes

Two abstract business processes are called structurally equivalent if there are an equal number of tasks in corresponding level of both hierarchical trees of abstract business processes with the same pre-control-flow(s), post-control-flow(s) and same number of adjacent tasks. If c1 and c2 are two concepts from the business process ontology θ_{BP} and the corresponding two abstract business processes are abp1 and abp2, hierarchical tasks tree GraphABP1 and GraphABP2, hence, can be found using the function constructGraph i.e. GraphABP1 = constructGraph (abp1) and GraphABP2= constructGraph (abp2). We defined that two abstract business processes are structurally equivalent if the both the hierarchical trees of the processes are structurally equivalent i.e. regardless of the contents of each task of the trees. Two hierarchical trees GraphABP1 and GraphABP2 are equivalent, iff they comply all of the following conditions.

i) The number of levels of GraphABP1 is equal to that of GraphABP2.

ii) There are same number of tasks in each corresponding level of both GraphABP1 and GraphABP2.

iii) For each level of GraphABP1, there is a mapping function which can map each task to the task of the corresponding level of the GraphABP2 structurally, i.e. contents of the task for example, name of the task, role, activity to be performed, are not taken into account.

3.3.2.2 Structural PartOf similarity between abstract business processes

Two abstract business processes are associated with each another with the structurally part of relationship if the graph of one abstract business process is contained structurally in the graph of the another abstract business process where the order how the tasks are connected with each other and the bindings of the tasks, which control flow connects which tasks, in each hierarchical tree, are taken into consideration, however, the contents of each task are disregarded. For better understanding the association, let us take two hierarchical trees created form tow business processes named Credit Application and Enrolment.

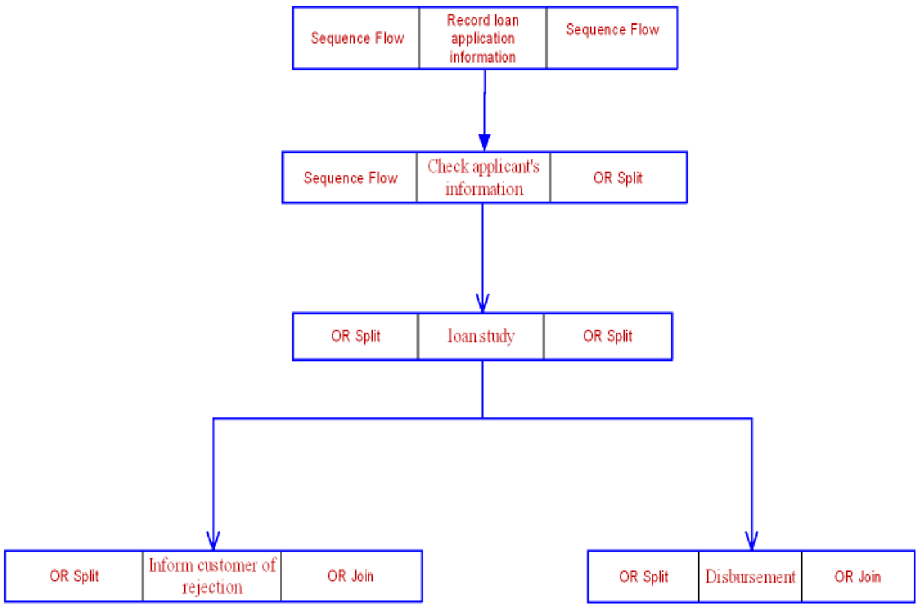

Fig-3(5): Graph for BPMN process Credit Application with pre and post flow operators

After being constructed and traversed the graph of Credit Application, we can see a hierarchical tasks tree as above which represents the acyclic hierarchical organization of the tasks with their pre-control-flows and post-control-flows. Each node of this graph is an object called hierarchical task which is capable of storing its pre and post flow; its adjacent tasks -tasks not yet traversed. Pre-flow is a control flow operator of any type that comes in from another task while post-flow is a flow operator that go out of the task.

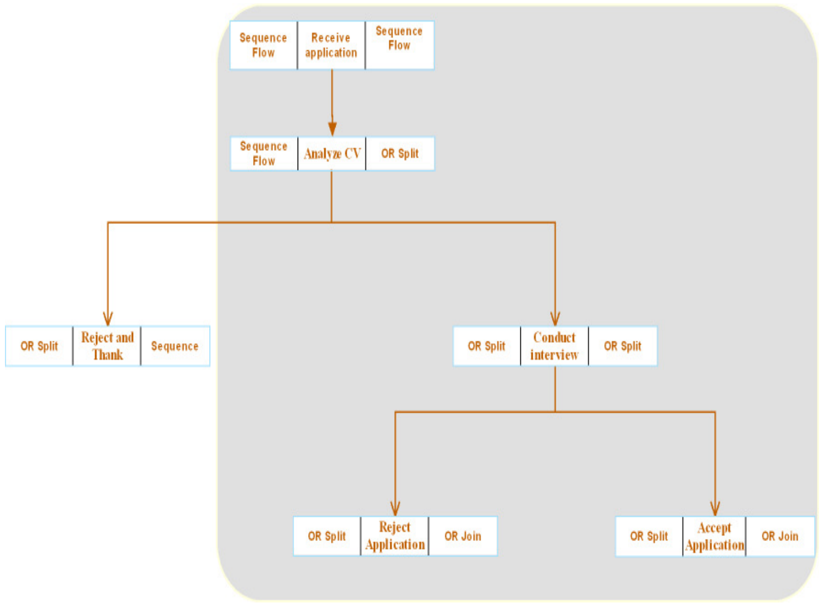

Fig-3(6): Association in terms of graph

From the above diagrams, we observed that the graph of Credit Application has the similar structure to the sub graph of Enrolment graph without considering the contents, visibly the names of the tasks. In every level of the Credit application, for every task there exists a corresponding structurally similar task with the same pre-control-flow and post-control-flow in the same level of the graph of Enrolment.

If c1 and c2 are two concepts from the business process ontology θ_{BP} and the corresponding two abstract business processes are abp1 and abp2, hierarchical tree GraphABP1 and GraphABP2 can then be retrieved using the function getConstructedGraph i.e. GraphABP1 = getConstructedGraph (abp1) and GraphABP1 = getConstructedGraph (abp2). Now we can define the structural part of relationship just considering the graphs of the abstract business processes. An abstract business process abp1 is structurally part of abp2 if abp1 is structurally contained in the abp2 i.e. the graph of abp1 is a sub graph of graphABP2. The graph graphABP1 is a sub graph ,iff they comply all of the following conditions.

i) GraphABP1.getNumebrOfTask < HTask2.getNumberOfTask

ii) GraphABP1.getMaxLevel() <= HTask2.getMaxLevel

iii) $\exists$ Htask $\in$ GraphABP2 **:** Htask $=_{\text{Struct}}$ GraphABP1.rootTask,
GraphABP2.subGraph(Htask) $=_{\text{Struct}}$ GraphABP1

For any task in GraphABP2, which is structurally similar to the root task of GraphABP1, there must be any sub graph of GraphABP2 with this task as a root task and this sub graph is structurally equivalent to GraphABP1.

3.3.2.3 Structural Overlapping similarity between abstract business processes

Two abstract business processes are associated with one another with a structurally overlap relationship if both of the hierarchical trees of the abstract business processes have at least a task in common without considering the contents such as task-name, actor, role etc. However, the pre and post control flows and the number of adjacent tasks of the both tasks must be taken into account. If c1 and c2 are two concepts from the business process ontology θ_{BP} and the corresponding two abstract business processes are abp1 and abp2, then hierarchical trees GraphABP1 and GraphABP2 can be constructed using the sub-routine getConstructedGraph. Two graphs GraphABP1 and GraphABP2 are structurally overlapped, iff they comply all of the following condition.

$$\text{GraphABP1.Htask} \cap_{\text{Struct}} \text{GraphABP2.Htask} \neq \emptyset$$

The intersection operation is not responsible to check the contents of the tasks; it only confirms the existence of any task in the graph. If two graphs are not empty i.e. each graph has at least on hierarchical task, does not matter where and what they are, both graphs are structurally overlapped, and hence the abstract business processes are.

3.3.3 Complete Similarity

Two abstract business processes are called completely equivalent if there are an equal number of hierarchical tasks in corresponding level of both hierarchical trees of abstract business processes and for each of the task in each level of a hierarchical tree, there must a hierarchical task in the same level in the other hierarchical tree with the same contents, same pre-control-flow, post-control-flow and number of adjacent hierarchical tasks. In other words two abstract business processes are completely equivalent if they are structurally as well as content equivalent with the same sequence of tasks. From the definition, a process is composed of a set of activities that are connected in the right way in a particular sequence to produce the desired outcome.

3.3.3.1 Complete equivalence between business processes

Two abstract business processes may be structurally equivalent as well as equivalent based on content, but these two similarities do not always guarantee that these processes are completely equivalent. This is because of the possible dissimilarity in the sequence of the tasks in the business process. For example, let us consider two BPMN diagrams below namely Employee Joining with medical Check-up first and Employee Joining with medical check-up last. For the sake of simplicity, same business process is considered here just toggling the activity sequence. However, in real case, two processes may have different domains of execution, means that similar activities such as choose material; create invoice; stock inventory; may be common in different business process in different manufacturing process, however, their sequence of execution may vary from one business process to another.

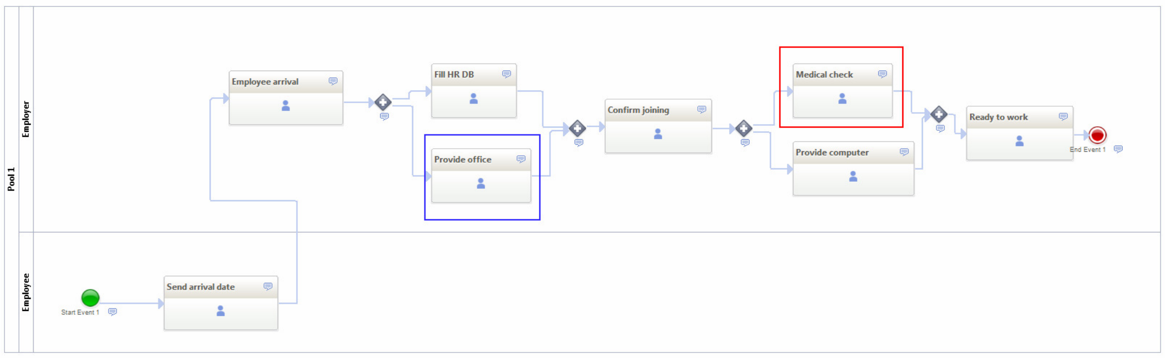

Fig-3(7): Employee Joining with medical Check-up first

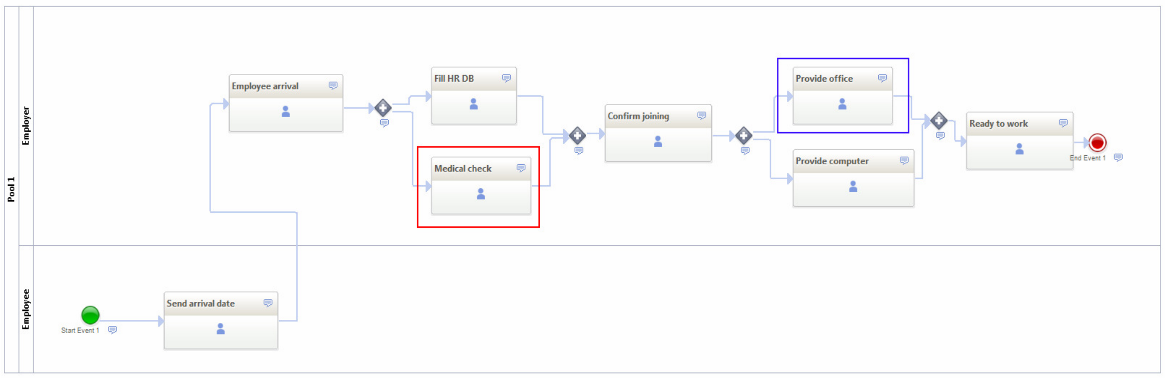

Fig-3(8): Employee Joining with medical check-up last

So, the problem arises from the mismatch in the sequence of activities to be performed in the concrete business processes, which are represented as tasks in our abstract business processes. The activity Medical check in the business process Employee Joining with medical Check-up first is executed after providing the office to the employee while in the business process Employee Joining with medical check-up last, medical Check-up is performed first before providing the office. Though, both processes have the same set of activities and same set of control flows, the sequence of the activities in both business processes are not same. Consequently, despite of being content equivalent and structural equivalent, both business processes are not completely equivalent.

By the way, we could manage to compare two business processes to find the structural similarity and the similarity based on contents. But it is impossible to say certainly that two business processes are completely equivalent, if both business processes are structurally similar and similar based on contents. So a new algorithm called completely match algorithm which can compare two abstract business processes considering also the sequence of the activities of the business process has been proposed. This algorithm is also designed retaining the idea of hierarchical structure of the business process. The algorithms for matching complete resemblance among business processes are the algorithms for structural matching algorithms with some significant modifications.

If abp1 and abp2 are two abstract business processes whose representations in the ontology are defined by c1 and c2, then we can construct corresponding graphs GraphABP1 and GraphABP2, using the sub-routine constructGraph(ABP). We defined that two abstract business processes are completely equivalent if the both the hierarchical trees of the abstract business processes are completely equivalent. In other words, we can say that two abstract business processes are completely equivalent if both abstract business processes are structurally as well as content based equivalent taking the sequence of the tasks into account. Two hierarchical trees GraphABP1 and GraphABP2 are completely equivalent, iff they comply all of the following conditions.

1. GraphABP1 is structurally equivalent of GraphABP2.

2. GraphABP1 is content equivalent of GraphABP2.

3. The sequence of tasks in both hierarchical trees is same.

3.3.3.2 Complete part of association among business processes

Two abstract business processes are associated with each another with the completely part of relationship if the hierarchical tree of one abstract business process is contained structurally and based on contents in the hierarchical tree of the another abstract business process where the order how the hierarchical tasks are connected with each other and the bindings of the tasks, which control flow connects which tasks, in each hierarchical tree, are also taken into consideration. We defined an abstract business process is completely part of another if both the hierarchical trees of these abstract business processes are equivalent. In another way, we can say that an abstract business process, abp1, is completely part of the other, abp2, if both abstract business processes are structurally as well as content based equivalent taking the sequence of the tasks into account. GraphABP1 is completely part of GraphABP2, iff the following conditions hold.

1. GraphABP1 is structurally part of GraphABP2.

2. GraphABP1 is content part of GraphABP2.

3. The sequence of tasks in both hierarchical trees is same.

3.3.3.3 Complete overlapping association among business processes

Two abstract business processes are associated with one another with the completely overlap relationship if both of the hierarchical trees of the abstract business processes have at least a task in common without considering the orders or positions of the tasks. In other way, we can define completely overlapping relationship as an abstract business process, abp1, is completely overlapped with another abstract business process, abp2, if abp1 is also both structurally and content overlapped with abp2. Two hierarchical trees GraphABP1 and GraphABP2 are structurally overlapped, iff the following conditions meet.

1. GraphABP1 is structurally overlapped with GraphABP2.

2. GraphABP1 is content overlapped with GraphABP2.

3.4 Business process matching algorithm

3.4.1 Content Based match making algorithm

Content based similarity is the concept of having an idea, how business processes share contents in the real life, and how to utilize the existing business process model just seeing the contents that you think are appropriate for your desire task without being started from the scratch or even not delving through the complete structure of the business process. One might need a business process which is already exist and functional. Instead of designing the whole process, it is possible just defining abstract business process to see whether there is any such process in the ontology to avoid reinventing the wheel.

3.4.1.1 Content based ABP match making algorithm

Content Based ABP match making algorithm is relatively simpler than Structural and complete match making algorithm for abstract business processes because we have design the algorithm considering only single dimensional objects, so called tasks of the abstract business process. Moreover, the algorithm mainly exploits the operations from set theory. The unique algorithm, we call it ContentBasedABPMatch, to check for three different content based association similarities of abstract business processes and to update the corresponding abstract process ontology with matched association(s) is given below in pseudo code.

3.4.1.2 Content Based Equivalence Matching Algorithm

Algorithm: ContentBasedABPMatch(Abstract Business Process abp1)

```
BEGIN
1  IF(There is not such concept that represents abp1)
2     Create a new concept that associates abp1
3  END IF
4  tasksOfabp1 = get tasks from abp1
5  FOR each other abstract business process in the ontology
6     tasksOFabp2 = get tasks from otherABP
7     IF (tasksOFabp1- tasksOFabp2 = $\emptyset$)
8        set property = contentEquivalence
9        update ontology (property, abp1, apb2)
10    ELSE IF ( tasksOFABP1 $\subseteq$ tasksOFabp2 OR tasksOFABP2 $\subseteq$ tasksOFabp1)
11       set property = contentPartOf
12       IF (tasksOFabp1.length < tasksOFabp2.length)
13          updateOntology(property, abp1,abp2)
14       ELSE
15          updateOntology(property, abp2,abp1)
16       END IF
17    END IF
18    IF (tasksOfAbp1 $\cap$ tasksOfAbp2 != null)
19       set property = contentOverlapped
20       update ontology(property, abp1, apb2)
21    END IF
22 END FOR
END
```

In the next few lines we will construe the Content Equivalence Matching Algorithm. Since we have defined concepts as the representatives of abstract business processes in the business process ontology, In line 1, we look for a concept that represent the abstract business process taken as an input of the algorithm. If no such concept is present in the ontology, predefined procedure, described in the publication the thesis is based on, should follow to create a concept. In line 2, taskOfabp1 is a set of type task ontology which receives and temporarily stores the tasks for further set operation to be carried out. Line 7 expresses explicitly the condition that must be met to say the abstract business processes, which are being treated currently, are content equivalent. The content equivalence association update operation is stated in line 9, where property specifies the association type; abp1 and abp2 are the two currently examined abstract business processes. Line 10 reveals two major criteria of finding content part of relationship among the processes. Besides, which process is content part of other depends on the size of the tasks each process has. If both processes have at least one task in common, the condition of which is specified in line 18, we can say they are content overlapped of each other.

For better understanding, let us illustrate the algorithm with the following BPMN diagrams.

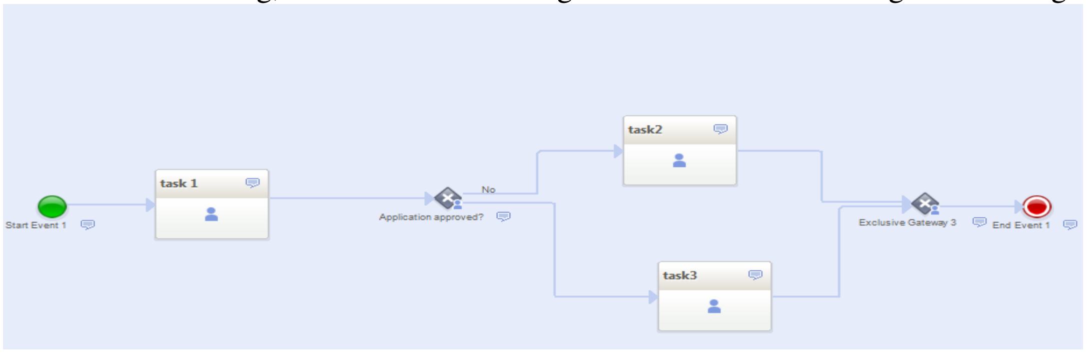

Fig-3(9): BP Content Similarity

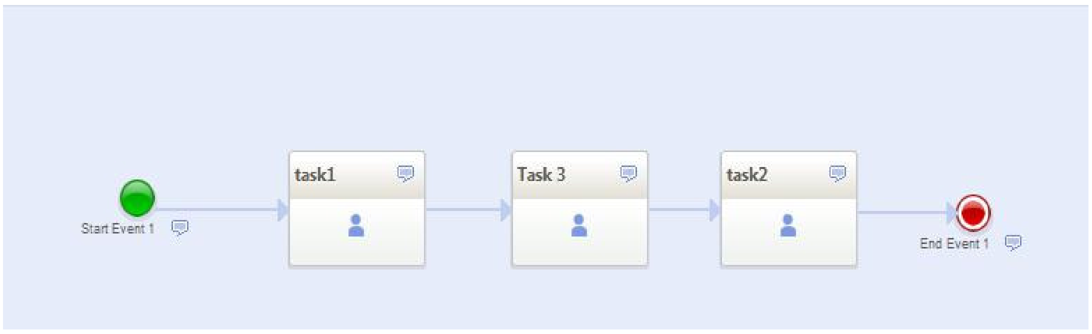

Fig-3(10): BP assumed to be Content Equivalent

Both processes above have the same set of tasks- task1, task2, and task3 of figure Fig-3(9) and of figure Fig-2(10) are assumed similar, therefore, our algorithm can identify them as content overlapped as well as content equivalent business processes.

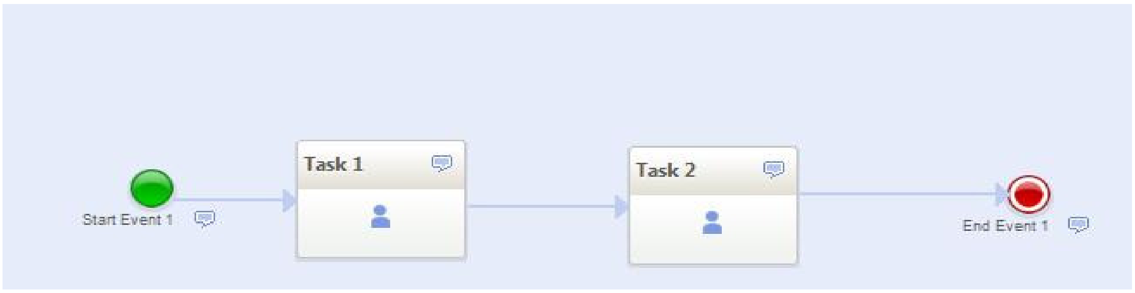

Fig-3(11): BP assumed to be Content Part

The business process depicted by figure Fig-3(11) is contained in both the processes portrayed by Fig-3(9) and Fig-3(10), as task1 and task2 both are belong to the subset of the tasks of both processes.

3.4.2 Structural match making Algorithm

Three algorithms have been proposed to compare the business processes for structural equivalence, structural part of and structural overlapping associations. Each of which has one or more sub-routines, most important sub-routines of them are also proposed below.

3.4.2.1 Structural Equivalence Matching Algorithm

3.4.2.1.1 Algorithm: MatchSturctEquivABPs(Abstract Business Process abp1)

```
BEGIN
1  GraphABP1= constructGraph(abp1)
2  For each abstract business process ontology class stored in the ontology
3     GraphABP2= constructGraph(abp2)
4     IF(GraphABP1.getSize() == GraphABP2.getSize() )
5        IF( GraphABP1.levelLength == GraphABP2. levelLength )
6           IF(isStructurallyEquivalent(GraphABP1, GraphABP2))
7              set property = StructurallyEquivalentce
8              update ontology(property, abp1, apb2)
9           END IF
10       END IF
11    END IF
12 END FOR
END
```

ConstructGraph, which is a subroutine destined to construct a graph of Hierarchical tasks, tasks are ordered as they are executed in the concrete business process, of a given abstract business process, is called in the line 1. To ensure that both abstract graphs are same, so are abstract business processes, the number of nodes and level numbers - each generation of the graph starting from the root node is defined as level of the graph - are compared in line 5 and line 6 respectively. Another subroutine named isStructurallyEquivalent, which takes two graphs as input, is invoked in line 7 in order to check whether both graphs have the same structure. If the sub-routine returns true, line 8 is executed to update the structural equivalence association among the abps in the ontology.

3.4.2.1.3 Sub-Routine for structural equivalence matching algorithm

```
FUNCTION: isStructurallyEquivalent(Graph1, Graph2)
IF(areTasksStructurallySimilar(Graph1.rootHTask, Graph2.rootHTask))
    FOR each corresponding next level of both Graph1 and Graph2
        If(Graph1.getHtask(level).size != Graph2.getHtask(level).size)
            Return false;
        END IF
        taskMatched[HTask] = null;
        FOR each hierarchical task hTask of Graph1
            FOR each hierarchical task hTask2 of Graph2
                IF(hTask2 !ϵ taskMatched)
                    IF(Pre-Fow and Post-flow of hTask and hTask2 are equal)
                        IF(NumberOfDescendents of hTask and hTask2 are equal)
                            taskMatched.add = hTask2;
                        END IF
                    END IF
                END IF
            END FOR
        END FOR
        IF(taskMatched.size != HTree.Abp1.getHTask(level).size)
            Return false;
        END IF
    END FOR
    return true;
ELSE
    Return false;
END IF
END FUNCTION
```

This sub-routine is responsible for comparing two graphs supplied by the main routine of structural equivalence. Before going through all the levels of the graphs, the routine must check the root tasks of both business processes first. This sub-routine delegates the job of finding structural similarity of the root hierarchical tasks of both graphs –supplied from the main routine- to another sub-routine called areTasksStructurallySimilar routine. The sub-routine areTasksStructurallySimilar return true if both root tasks have same pre-control-flow, post-control-flow and same number of adjacent hierarchical tasks. Hence we can proceed with all subsequent level next to search for structural resemblance.

3.4.2.1.3 Sub-Routine for structurally matching two hierarchical tasks

```
FUNCTION: areTasksStructurallySimilar(hTask1, hTask2)
    IF(Pre-Fow and Post-flow of hTask1and hTask2 are equal)
        IF(NumberOfDescendents of hTask1and hTask2 are equal)
            RETURN true
        END IF
    END IF
RETURN FALSE
END FUNCTION
```

This sub-routine confirms that the both hierarchical tasks, which are represented as vertices in graph, have same basic pre and post flow operator- for example, ORSplit; ANDSplit; ORJoin; ANDJoin- specified in line 1, and same number of adjacent vertices that are still to be traversed in line 2.

For better understanding the action of the algorithm, let us have a look to the business processes depicted below.

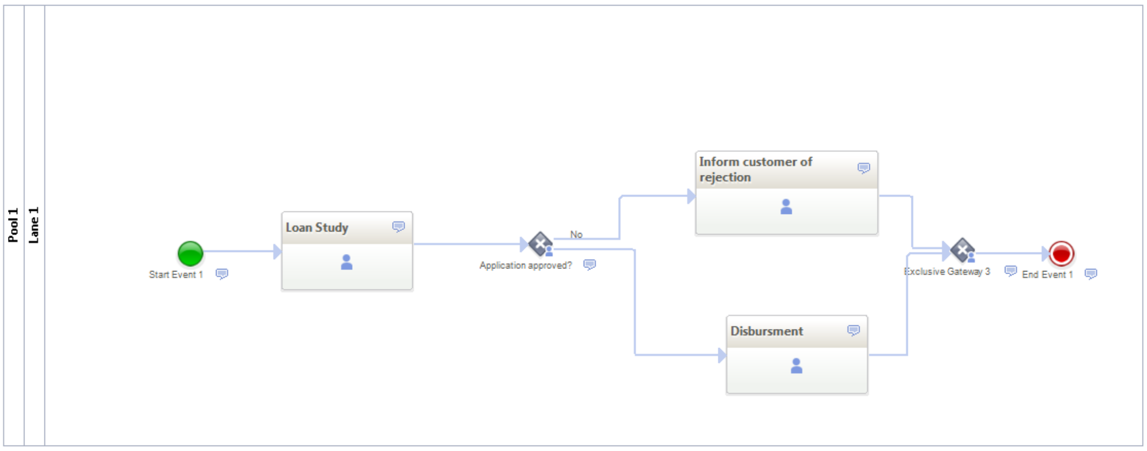

Fig-3(12): BP Credit application sub

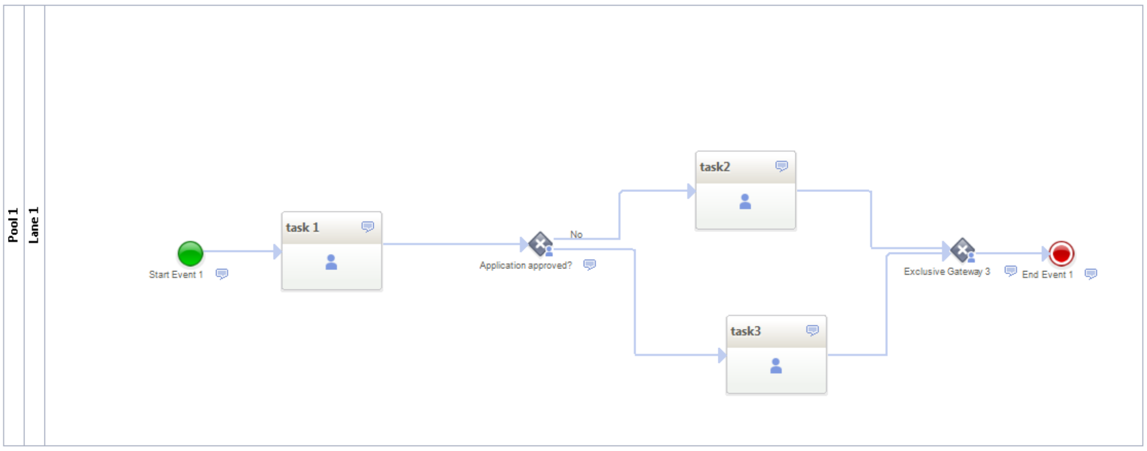

Fig-3(13): BP assumed to Structural Equivalent

To say whether two business processes –process represented by Fig-3(12) and Fig-3(13)- are equivalent or not, our algorithm checks the presence of tasks and flows in the same corresponding positions of the business processes. In the processes above, we see that both processes have the same number of tasks and same set of flows except the name of the task which represents that the tasks are not similar, and hence, it can find them structurally similar.

3.4.2.2 Structurally Part of matching algorithm

3.4.2.2.1 Algorithm: MatchStrucPartABPs(Abstact Business Process abp1)

```
BEGIN
1  GraphABP1 =  getConstructedGraph(abp1)
2  For each abstract business process stored in the ontology
3      GraphABP2=  getConstructedGraph (abp2)
4      IF(GraphABP1.getSize() > GraphABP2.getSize())
5          IF(isStructurallyPartOf(GraphABP2, GraphABP1))
6              set property = StructurallyPartOf
7              update ontology(property, abp2, apb1) //abp2 is Struct part of abp1
8          END IF
9      ElseIF(GraphABP1.getSize() < GraphABP2.getSize())
10         IF(isStructurallyPartOf(GraphABP1, GraphABP2))
11             set property = StructurallyPartOf
12             update ontology(property, abp1, apb2) //abp1 is Struct part of abp2
13         END IF
14     END IF
15 END FOR
END
```

After constructing graph for abstract business process abp1, it iteratively construct graph for each other abstract business process already exist in the ontology in the form of concepts defined in line 3. Line 4 and line 9 ensure the difference between the tasks size which does matter among abstract business processes to decide which graph would be the sub graph. A sub-routine, which returns true if the graph supplied by first parameter is structurally part of the graph supplied by the second parameter, is called in line 5. Line 7 updates the ontology as abp2 is structurally part of abp1 while line 12 updates the ontology as abp1 is structurally part of abp2.

3.4.2.2.2 Sub-routine for structural part of Matching Algorithm

```
FUNCTION isStructurallyPartOf(Graph1, Graph2)
1  LevelBeingTraversed = 1
2  WHILE(Graph2.levelLength – LevelBeingTraversed >= Graph1.levelLength )
3      FOR each hTask at level LevelBeingTraversed of Graph2
4          IF(areTasksStructurallySimilar(hTask,Graph1.rootHTask))
5              IF(isStructurallyEquivalent(Graph1, Graph2.getSubGraph(hTask)))
6                  RETURN TRUE
7              END IF
8          END IF
9      END FOR
10     increment LevelBeingTraversed by 1
11 END WHILE
12 RETURN FALSE
END FUNCTION
```

As Graph2 has the larger set of vertices known from the main routine MatchStrucPartABPs, it is probable that Graph1 may contain structurally in any portion, defined as sub-graph, of the Graph1. So, we have to traverse the second graph staring from the first generation i.e. from the root of Graph2. To visit the first hierarchical task in Graph2, the level, LevelBeingTraversed, is set to 1 in the first line of the sub-routine. The method getSubGraph of the Graph2 is invoked in line 5 to get a complete graph where the root is hTask which is passed as a parameter. If the structural similarity is found in Graph1 and sub-Graph of Graph2, the subroutine returns true to the main routine, hence to substantiate the update of the ontology.

Let us consider the following two business processes for structural part of similarity matching algorithm simulation.

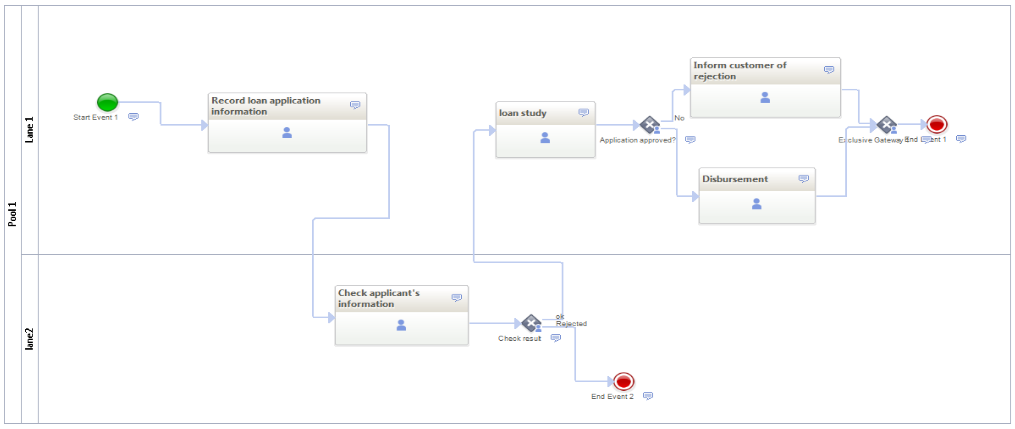

Fig-3(14): BP Credit card Application

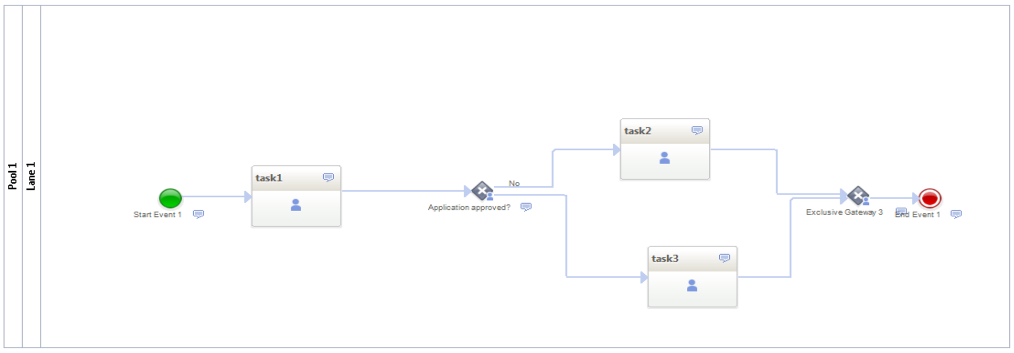

Fig-3(15): BP assumed as Structurally Part of

Our algorithm can identify the association exist between above processes is part of, i.e. the process depicted by Fig – 3(15) will be found as a part of the process shown by fig – 3(16). If we consider the processes for the following structural overlapping algorithm, we will see that the later algorithm will find the structural overlapping relationship among the business processes.

3.4.2.3 Structurally Overlapping Algorithm

3.4.2.3.1 Algorithm: MatchStructOverLappedABPs(Abstact Business Process abp1)

```
BEGIN
1  GraphABP1 =  getConstructedGraph(abp1)
2  For each abstract business process in the ontology
3      GraphABP2=  getConstructedGraph (abp2)
4      IF(isStructurallyOverlapped(GraphABP1, GraphABP2))
5          set property = StructurallyOverlapped
6          update ontology (property, abp1, apb2)
7      END IF
8  END FOR
END
```

The method getConstructedGraph invoked in line 1 and line 3 is concurred to retrieve the graph already constructed for structural equivalence comparison, otherwise create the graph using the abstract process ontology. The ontology will be updated as abp1 is structurally overlapped with abp2and vice versa, if the sub-routine which is called in line 4 returns true.

3.4.2.3.2. Sub-routine of Structural overlapping Algorithm

```
FUNCTION: isStructurallyOverLapped(Graph1, Graph2)
1  FOR each level levelG1 of Graph1
2      FOR each hierarchical task hTask1 at levelG1
3          FOR each level levelG2 of Graph2
4              FOR each hierarchical task hTask2 at levelG2
5                  IF(hTask1.getTaskOntology != null & hTask2.getTaskOntology != null)
6                      Return true;
7                  END IF
8              END FOR
9          END FOR
10     END FOR
11 END FOR
12 RETURN FALSE
END FUNCTION
```

The delegated task, task from the main routine of structural overlapping, is to check if there exists at least one task is structurally common in both graphs. It seems to be a complex situation to overcome, nevertheless, the idea is as simple as to check whether the graphs are non-empty i.e. both graphs have at least one vertex.

3.4.3 Complete match making algorithm

3.4.3.1 Complete Equivalence Matching Algorithm

3.4.3.1.1 Algorithm: MatchCompleteEquivABPs(Abstract Business Process abp1)

```
BEGIN
1  Graph1 =  constructGraph(abp1)
2  For each abstract business process in the ontology
3      GraphABP2=  constructGraph(abp2)
4      IF(Graph1.getSize() == GraphABP2.getSize() )
5          IF( Graph1.levelLength == GraphABP2. levelLength )
6              IF(isCompletelyEquivalent(Graph1, GraphABP2))
7                  Set property = CompletelyEquivalentce
8                  update ontology(property, abp1, apb2)
9              END IF
10         END IF
11     END IF
12 END FOR
END
```

As a hierarchical tree represents vertices in orders starting from the primitive ancestor to the lowest descendent, we can rely on the graph matching technique to compare two abstract business processes for complete equivalence which considers not only structural and content similarity but also the sequence of the activities to be executed in concrete business processes. Graph construction sub-routine, constructGraph which is invoked in line 1 and line 3, is responsible to construct a graph for an abstract business process which is already defined in the process ontology. Sine the order of the abstract business process is represented in the graph, it is crucial to check whether both graphs have the same number of generations which is done in line 5. To compare the every detail in the corresponding vertices of the graphs, the algorithm delegates the duty to another sub-routine known as isCompletelyEquivalent that takes both graphs as parameters.

3.4.3.1.2 Sub-routine for complete equivalence algorithm

```
FUNCTION : isCompletelyEquivalent(Graph1, Graph2)
1  IF (areTasksSimilar(Graph1.getRootTask, Graph2. getRootTask))
2      FOR each corresponding next level of both Graph1 and Graph2
3          IF(Graph1.getHtask(level).size ≠ Graph2.getHtask(level).size)
4              Return false
5          END IF
6          taskMatched[HTask] = null;
7          FOR each hierarchical task hTask of Graph1
8              FOR each hierarchical task hTask2 of Graph2
9                  IF(hTask.getTask.equals(hTask2.getTask) & hTask2 ∉ taskMatched)
10                     IF(Pre-Fow and Post-flow of hTask and hTask2 are equal)
11                         IF(NumberOfDescendents of hTask and hTask2 are equal)
12                             taskMatched.add = hTask2;
13                         END IF
14                     END IF
15                 END IF
16             END FOR
17         END FOR
```

```
18          IF(taskMatched.size  ≠ HTree.Abp1.getHTask(level).size)
19              Return false;
20          END IF
21      END FOR
22      return true;
23 ELSE
24      Return false;
25 END IF
END FUNCTION
```

The complete comparison of the vertices is done by comparing the tasks mentioned in line 9, pre and post control flows of the tasks in line 10 and number of adjacent tasks that are not yet visited in line 11. It is worth mentioning that the level or generation of the comparing vertices must be same for both graphs to sustain the order of the activities. If the number of tasks matched in any level is not congruent, then the sub-routine must return false to the main routine which is mentioned in line 18 and 19.

3.4.3.1.3 Sub-routine for matching two hierarchical tasks

```
FUNCTION: areTasksSimilar(hTask1, hTask2)
1  IF(hTask1.getTask().equals(hTask2).getTask())
2      IF(Pre-Fow and Post-flow of hTask1and hTask2 are equal)
3          IF(NumberOfDescendents of hTask1and hTask2 are equal)
4              RETURN true
5          END IF
6      END IF
7  END IF
8 RETURN FALSE
9 END FUNCTION
```

Since we are considering the vertex of a graph as a task of the business process, hence the comparison between vertices – names of the task- is not to say the vertices are similar. Therefore, we have considered here the vertex as a Hierarchical task object having pre-flow-operator, post-flow-operator, and the number of adjacent which are not yet visited. So, to say two HTasks are similar, line 1, lin2, and line3 compare tasks, pre and post control flow and number of descendents not yet traversed respectively.

3.4.3.2 Completely Part of Matching algorithm

3.4.3.2.1 Algorithm: MatchCompletePartABPs(Abstact Business Process abp1)

```
BEGIN
1  GraphABP1 =  getConstructedGraph(abp1)
2  For each abstract business process ontology class stored in the ontology
3      GraphABP2=  getConstructedGraph (abp2)
4      IF(GraphABP1 .numberOfNodes > GraphABP2. numberOfNodes )
5          IF(isCompletelyPartOf(GraphABP2, GraphABP1 ))
6              set property = CompletelyPartOf
7              update ontology(property, abp2, apb1) //abp2 is complete part of abp1
8          END IF
9      ElseIF(GraphABP1 .numberOfNodes <GraphABP2. numberOfNodes )
```

```
10          IF(isCompletelyPartOf (GraphABP1 , GraphABP2))
11              set property = CompletelyPartOf
12              update ontology(property, abp1, apb2) //abp1 is complete part of abp2
13          END IF
14      END IF
15 END FOR
END
```

In line 4 and line, the comparison between the abstract processes' number of nodes are made, to distinguish the abstract business processes. The distinction is such that one abstract business process would be part of another and the other business process would contain another business process. In line 5 and line 10, a sub-routine – isCompletelyPartOf- is called to compare the passing graphs as parameters. Depending on the outcome of the sub-routine as true, the ontology is updated with the association type of CompletelyPartOf.

3.4.2.2.2 Sub-routine for Completely part of Matching Algorithm

```
FUNCTION CompletelyPartOf (Graph1, Graph2)
1 Set LevelOfFirstMatchedHtask to 1
2 While (Graph2.levelLength – LevelOfFirstMatchedHtask >= Graph1. levelLength )
3     FOR each graph node hTask2 at level LevelOfFirstMatchedHtask of Graph2
4         IF (areTasksSimilar(Graph1.getRootTask, hTask2))
5             IF(isCompletelyEquivalent (Graph1, Graph2.SubGraph(HTaski)))
6                 Return true
7             END IF
8         END IF
9     END FOR
10 Increment LevelOfFirstMatchedHtask by 1
11 END WHILE
12 return false;
END FUNCTION
```

The root task of Graph1, which having the smaller set of vertices, is compared with each task of Graph2 starting from the root vertex until a match is found. Line 1 set the level of the root vertex of the graph 2. A condition is set in line 2 to check whether it is still feasible to get a part of relationship between both processes. Another two sub-routines, areTasksSimilar and isCompletelyEquivalent, are also invoked from this routine. In line 4, sub-routine is called to find the similarity between the root task of Graph1 and the task of Graph2 just being traversed. A subGraph of Graph2 is retrieved to pass as parameter of second sub-routine in line 5 to exploit the behaviour of the sub-routine isCompletelyEquivalent here in CompletelyPartOf routine. Let us consider the following business process, Credit Card Application Sub.

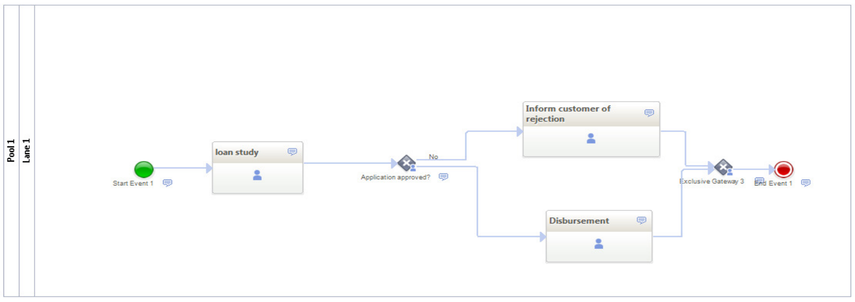

Fig-3(16): BP Credit Card Application Sub

Since the process depicted in Fig-3(16) is content part of and structurally part of the process shown by Fig-3(14), and sequence is also the same, the algorithm, MatchCompletePartABPs, will find that Credit card application sub is completely contained in the Credit Card application process.

3.4.2.3.1 Completely Overlapping Algorithm

```
Algorithm: MatchCompletelyOverLappedABPs(Abstact Business Process abp1)
BEGIN
    1 GraphABP1 =  getConstructedGraph(abp1)
    2       For each abstract business process in the ontology
    3           GraphABP2=  getConstructedGraph(abp2
    4           IF(isCompletelyOverLapped ((GraphABP1, GraphABP2))
    5               set property = CompletelyOverLapped
    6               update ontology(property, abp1, apb2)
    7           END IF
    8       END FOR
END
```

After getting constructed graph for abp1, the algorithm gets constructed graph for every other abstract business process. Though the complete overlapping association is same as content overlapping, the aim of this algorithm is to show that content overlapping or complete overlapping is also possible using the hierarchical tree structure of the business processes. However, constructing graph for updating complete overlapping is not feasible and optimistic. If the construction is already made for other complete associations – complete equivalence and complete part of -, which are completely different approach from content equivalence and content part of, we can exploit it as well for updating the complete overlapping association.

3.4.2.3.2 Sub-routine for Complete overlapping Algorithm

```
FUNCTION: isCompletelyOverLapped(Graph1, Graph2)
1   FOR each level levelG1 of Graph1
1       FOR each hierarchical task hTask at levelG1
3           FOR each level levelG2 of Graph2
4               FOR each hierarchical task hTask2 at levelG2
5                   IF(hTask.getTaskOnotolgy.equals.(hTask2.getTaskOnotolgy))
6                       Return true;
7                   END IF
8               END FOR
```

```
9           END FOR
10      END FOR
11 END FOR
12 Return false;
```

END FUNCTION

Chapter 4

Implementation

4.1 Technologies

The Semantic Web vision, which gained high value and praise, was emerged by Sir Tim Berners-Lee. To transform his vision to reality, many researchers have been working with multifarious efforts to make semantic web functioning.

The United States Department of Defense (DoD) primarily led the research work on semantic web with its Defense Advanced Research Products Agency (DARPA) organization. The first research outcome of DARPA on semantic Web technology was the DARPA Agent Markup Language (DAML) which fostered defining the conceptualization of semantic web. EU also showed interest in semantic representation of the information. Consequently, a EU research product, Ontology Interface Layer (OIL), had been developed. A later merge of DAML and OIL from US and EU was made to form a new language called DAML + OIL.

Since, there had been different organizations engaged in research on semantic technology, uniformity in representing ontology, i.e. meaning of the information and relationship among them, was required. Eventually, to work on standardization of semantic web language, W3C begun its activities with the help of web ontology working group. The group underscored on the standard ontology language elements to sustain the development of the semantics.

Web Ontology Language known as OWL is designed to foster the use of machine interoperability. Using OWL, information coded in the documents can be processed by the applications. So, the intention of OWL is to represent terms with their meaning and the relationships among them to facilitate the machines to share information without human intervention. OWL provides the way to define terms and their relationships, so as to web contents, more comprehensively than others: XML – syntax without semantic constraints-, RDF-data model with simple semantics represented in XML syntax-, and RDF-S -vocabulary that describes the properties and classes of RDF resources. OWL is up-to-the-minute part of the evolving stack of Semantic Web recommended by w3C recommendation. XML is the most primitive syntax for making documents.

OWL is a semantic Web standard which is a framework got final approval from World Wide Web consortium in February 2004, to provide a way of sharing and reusing information among machines. A set of documents have been introduced by W3C to explain the OWL language; each of which satisfy a different purpose. For example, OWL overview document describes brief introduction and language features; OWL Reference document construe all modelling primitives of OWL; normative definitions of the language are officially describes in the OWL Semantics and Abstract Syntax document.

The OWL language is sub classified in three different sublanguages so called species of OWL -OWL Lite; OWL DL; OWL Full- for satisfying purpose of specific communities of ontology developers and users. Brief discussion on each of them regarding the difference between them is as follows.

OWL Lite is the most limited featured OWL language among three. OWL Lite supports only a limited version of cardinality. The cardinalities that are allowed in OWL Lite are 0 or 1. OWL Lite is adhered to all restrictions that are also laid on by OWL DL language.

OWL DL emphasizes two key factors: computation completeness and decidability of reasoning system that must be ensured while using the language. All OWL DL constructors are well supported by OWL DL; however, certain restrictions are imposed on their usage. For instance, a class may become a subclass of one or more classes, but a class can not be added as an instance of another.

OWL Full belongs both the complete set of constructs of OWL language and RDF constructs. Class definition from both OWL and RDF, i.e. owl:class and rdf:class are equivalent, where as owl:class is a proper sub-class of rdf:class in both OWL Lite and OWL DL. Hence, a class created in OWL Lite and OWL DL is sub class of RDF by default. A class, which is a sub class of another class, can also act as an instance of that sub-class. For instance, a class owl: Ferrari, which is a sub-class of owl:car, can act both as a class as well as individual of the class owl: car.

4.2 Requirements in UML

After being understood the context, the identified problem, and the solution proposed, it is important to understand the system and user requirements and the relationship between the model objects before proceeding to the implementation part. Use case diagram is depicted towards this goal.

4.2.1 Use case

To design the use case diagram, we considered all stakeholders involved in the system implementation, use and the use cases that are invaluable part for our system design, the relationships between the stakeholders and the use cases.

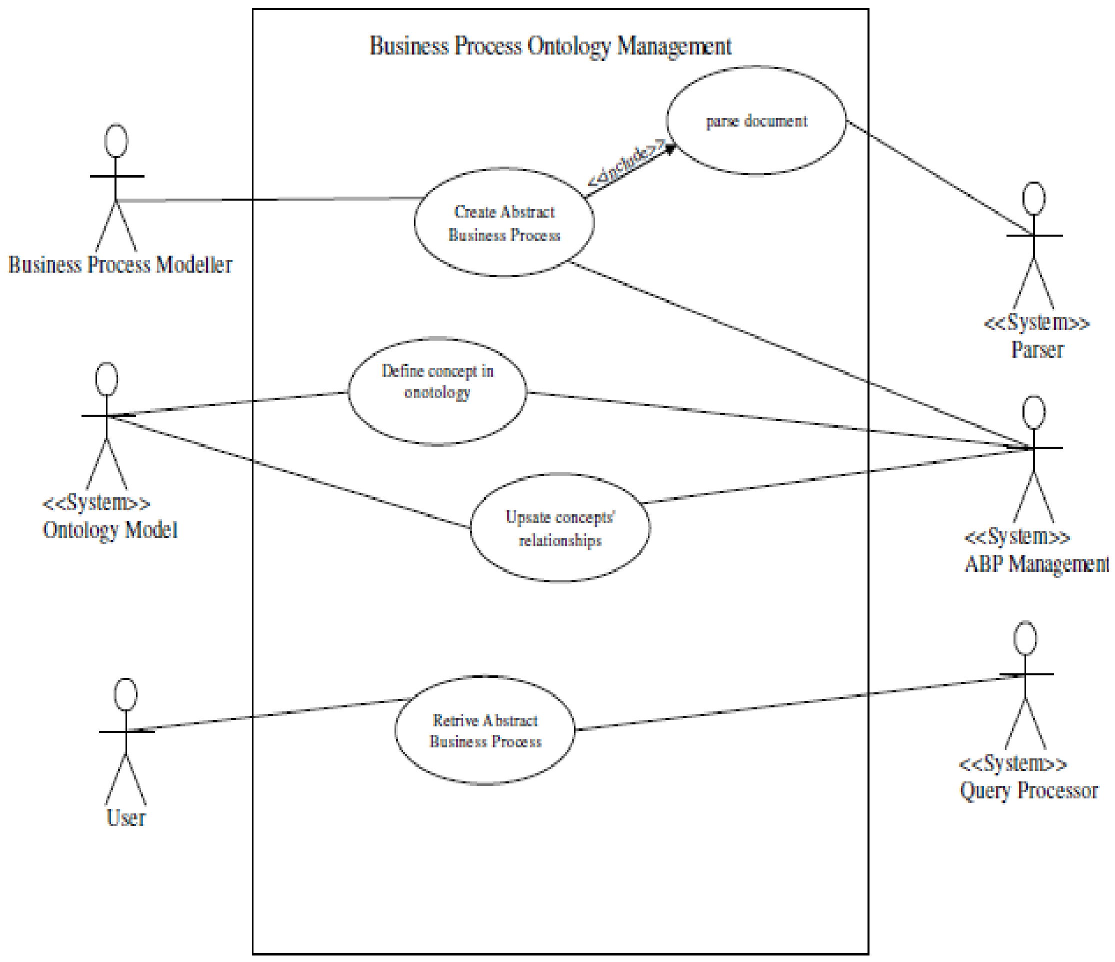
Business Process Ontology Management
parse document
<<include>>
Create Abstract Business Process
Business Process Modeller
<<System>> Parser
Define concept in onotology
<<System>> Ontology Model
Upsate concepts' relationships
<<System>> ABP Management
Retrive Abstract Business Process
User
<<System>> Query Processor

Fig- 4(1): Use Case Diagram - Business Process Ontology Management

The user interacts with the system to retrieve the searching result from the system using the "Retrieve Abstract Business process" use case. The use case diagram "create abstract Business process" involves stakeholders: Business Process Modeller, ABP Management and Parser that includes the parsing operation.

4.2.2 Activity Diagram

The activity diagram, which is useful to construe the flow of operations involved in a use case diagram, is depicted below for the use case "update concepts' relationships" illustrated in the use case diagram. Most of our contributions in this thesis involve the associations among the business processes and their update operations in the business process ontology. As the activity diagram depicted below describes the same things, we are interested only in it.

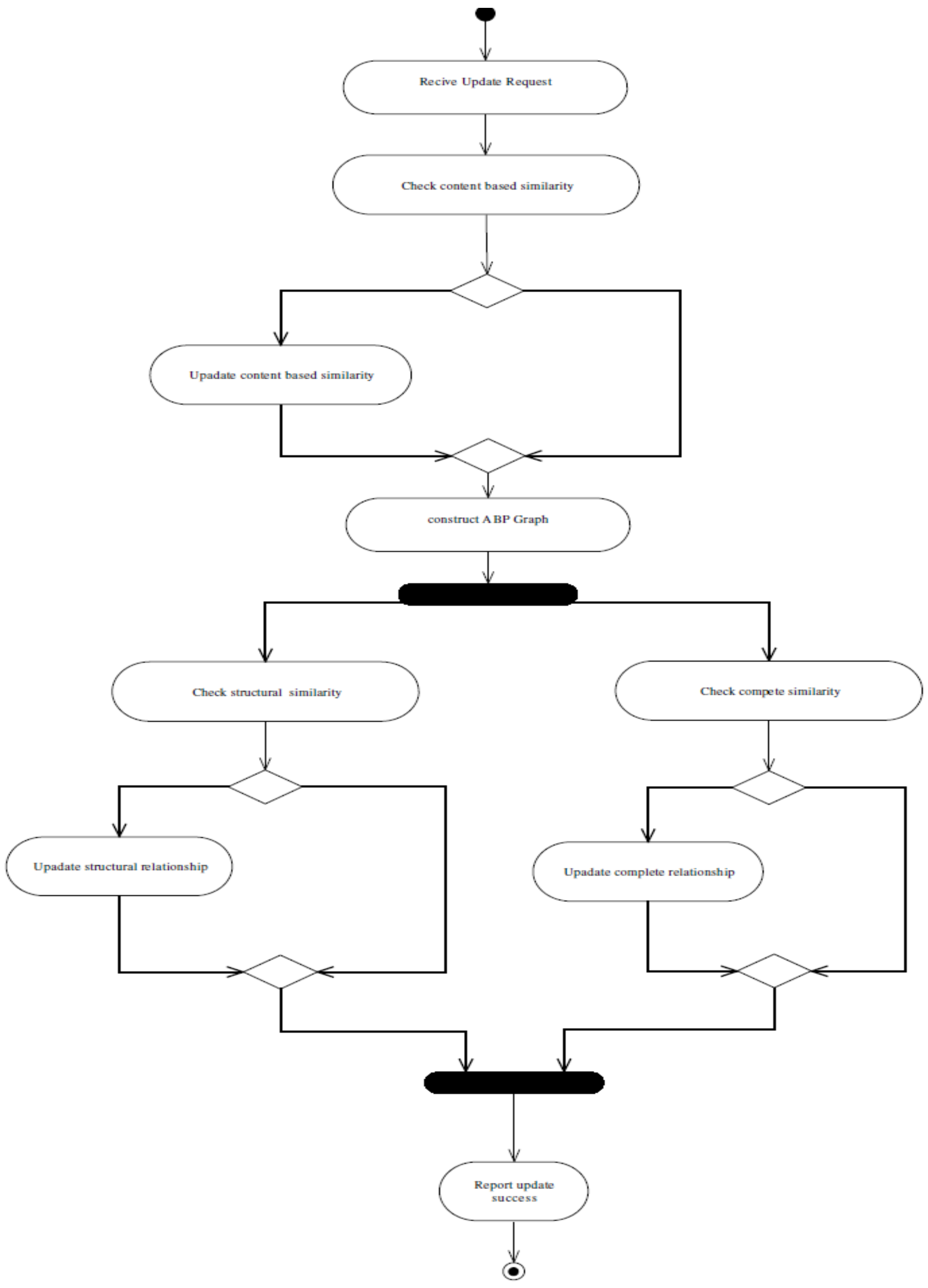
Recive Update Request
Check content based similarity
Upadate content based similarity
construct A BP Graph
Check structural similarity
Check compete similarity
Upadate structural relationship
Upadate complete relationship
Report update success

Fig- 4(2): Activity diagram update concepts' relationships

4.3 Design

The implementation of the thesis work is based on the object oriented approach. Class diagram has been introduced for object oriented modelling required for our wok; the classes, their attributes and methods, and the relationships between the classed are illustrated well.

4.3.1 Class Diagram

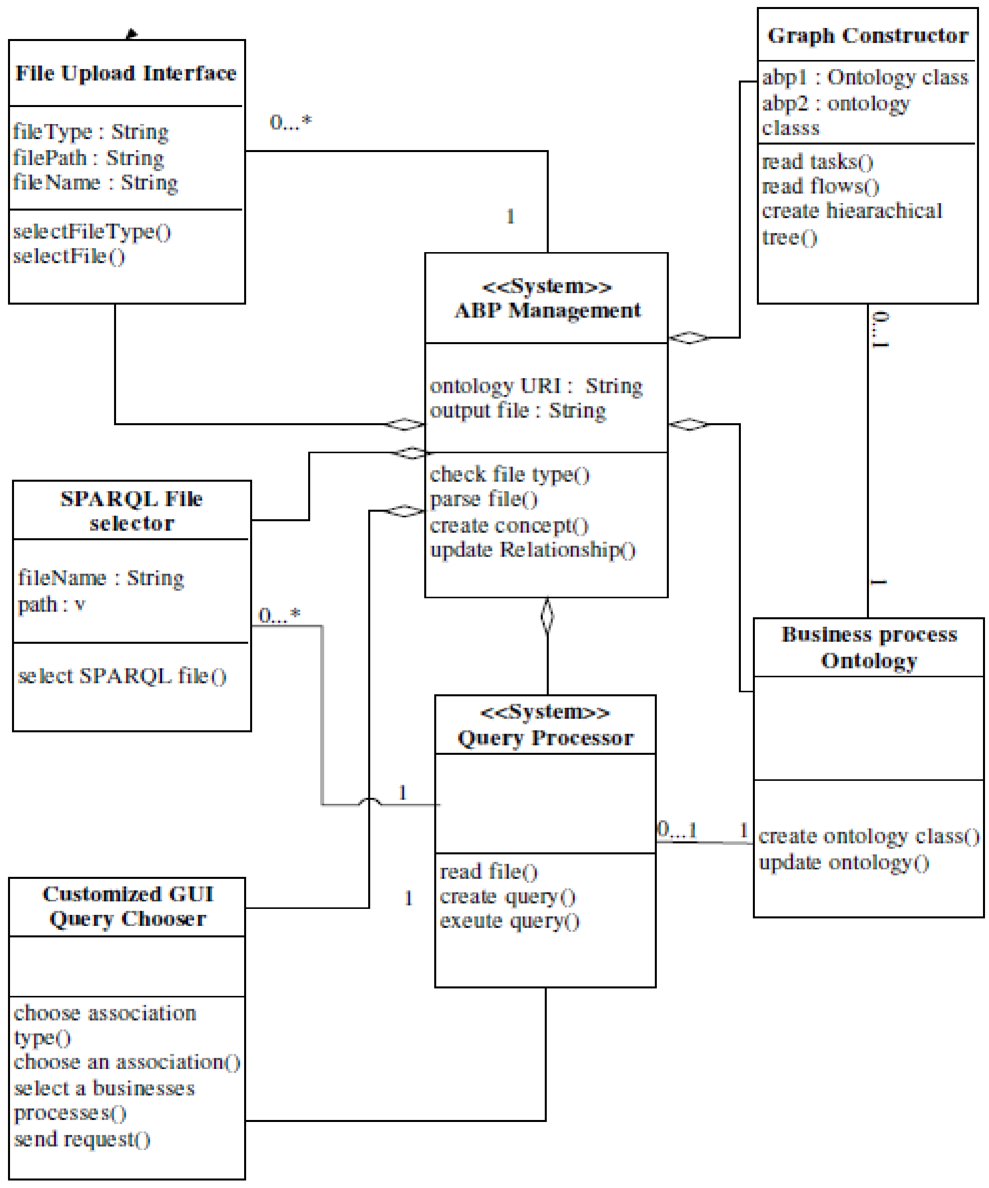

Fig- 4(3): Class diagram of Business Process Ontology Management

An object of ABP Management class works as an organizer of the whole set of objects created by other classes in order to make the system functional. The multiplicity among the classes is important, and hence, can affect the efficiency of the system to be developed. Note that, for any association that involves the business process ontology, there is always only one object of the ontology exists.

4.4 Implementation Details

4.4.1 Implementation tools

To make our application function, we had to go through decisive activities to pick the right tool for the right purpose. Firstly, since we do not have a standard format, which describes business processes modelled by all disparate modellers, we had to use an abstract business process extraction technique to import business processes in our application. Secondly, from manifold APIs available nowadays for Ontology model, it is important to select a good API for ontology modelling. Finally, we had to select a reasoner to check the ontology consistency.

4.4.1.1 Abstract Business Process Extraction

The extraction technique for creating abstract business processes, which was employed in the previous work, is kept the same, as this thesis work is an extension of the work is to develop more robust system in terms of more comprehensive associations among the business processes which are based on structural similarity, content based proximity and completeness. Though extraction technique has been elaborately described in the preliminary work, we discussed the technique briefly for the sake of comprehensibility of the thesis.

Since there are multifarious file formats that are used in disparate business process models, an intermediate XML file has been proposed to create abstract business processes from this type of file. Moreover, the modeller the thesis work used is WebRatio. However, the file created from the WebRatio business process model is not provided by itself, but we surmised the file format with some hopeful conjectures. The file , so called BPM file , is created using XLST with the hope that if any changes done in BPM files can be compensated just changing the XLST file. Nevertheless, we also takes primitive XML file format as input, as it is a well known structured document format that can be accessed conveniently.

4.4.1.2 Ontology API

To build semantic web applications, there are several APIs available today. Each API has its own merits and characteristics, though, the general concept is to provide framework to substantiate the use of ontology for applications. Jena, which was developed in HP lab, is used for out thesis work. For our application, why we assimilated Jena API is briefly discussed below.

Jena is java based framework for ontology applications which is an open source product. It provides APIs for manipulating RDF, RDFS, OWL, SPARQL queries. To execute SPARQL queries, it has build in SPARQL query engine. It also integrated a rule based inference engine. Jena framework also included In-memory and persistent storage.

4.4.1.3 Ontology reasoner

A reasoner is required in our thesis work to check the inconsistency of a set of axioms stored in the ontology. Pallet – written in java- provides reasoning facilities of OWL-DL ontology. It is workable with approximately all axioms declared in OWL1 and OWL2. Since it holds dual-licensed- both for open source and commercial use- and being OWL-DL compatible, we have chosen it in our academic purpose.

4.4.2 Key aspects in coding

The construction of abstract business process concept, its task ontology and flow ontology is kept same as the prior thesis work, while the relationship update mechanism has been changed significantly. Hence, we underscored mainly in describing the relationship update in the ontology.

To update the content based relationships among business processes, we proposed a single algorithm. The java implementation of this algorithm is pretty straight forward and hence, we do not like the same thing in another form, which could sound monotonous. Therefore, we acquiesced to describe the implementation of the key aspects of the thesis i.e. the different types of association among the business processes.

4.4.2.1 Graph construction

An abstract business process is content equivalent with other, if they have the same number of tasks and same set flows i.e. we may have tasks that describe completely different activities of the concrete business process, however, they must be connected using the same set of flows with same order. To do this, we have proposed the notion of business process graph.

To construct a graph from an abstract business process stored in the ontology in the form of concept, a method called constructGraph has been developed. This method takes current context, task and flow ontology classes of the process as input and returns a graph represented by a concurrent hash map object of java language.

This method finds the start event of the process, and generates the complete graph taking the first task as a root vertex. The construction of the graph is done with following java method implementation.

```
public static ConcurrentHashMap<Integer, ArrayList<HierarchicalTask>>
constructGraph(ABPUpdate context,Set<OntClass> tasks,
Set<OntClass> flows) {
        GraphConstructor.context = context;
        GraphConstructor.tasks = tasks;
        GraphConstructor.flows = flows;
        GraphConstructor.tasksvisited = new HashSet<OntClass>();
          OntClass startEvent =
          ABPUpdate.ontology_base_model.getOntClass(ABPUpdate.ontology
          URI + "#Start");
          hierarchicalTasks = new ConcurrentHashMap<Integer,
          ArrayList<HierarchicalTask>>();
            if(startEvent == null)
            {
            System.out.println("Start Event not found");
            return null;
            }
        checkRootVertex(startEvent);
        return hierarchicalTasks;
        }
```

In the checkRootVertex method, a snippet finds the flow operator that is connected to the start event, is given below.

```
        if(task.equals(FlowOnt.getFlowTask(context, indivualFlow))){
                if(FlowOnt.flowDirection == IN_DEGREE) {
                        startFlow = FlowOnt.getFlowOperator(context,
                        indivualFlow);
                        break;
                }
        }
```

After the operator being found, this method finds the root task- first task of the business process- which is connected to the start event with this flow operator. If such a root task is found in the business process, then it calls a recursive method, *popualteDescendents,* which digs up all descendents of the root task and construct a graph. The lines of code that perform this operation are as follows.

```
if(!task.equals(FlowOnt.getFlowTask(context,iflow))
&&FlowOnt.getFlowOperator(context,iflow).equals(startFlow)){
        if(FlowOnt.flowDirection == OUT_DEGREE) {
                if(!tasksvisited.contains(
                FlowOnt.getFlowTask(context, iflow))){
                                popualteDescendents(FlowOnt.getFlowTask(con text,
                        iflow),new Integer(1));
                                }
                        }
                }


void popualteDescendents(OntClass task, Integer level){
        Integer presentLevel = level;
        OntClass startFlow=null;
        OntClass indivualFlow = null;
        HierarchicalTask hierarchicalTask = new HierarchicalTask();
        Set<OntClass> flowsCopy = flows;
        for(Iterator<OntClass> flowIterator = flowsCopy.iterator();
        flowIterator.hasNext();)
                {
                        indivualFlow = flowIterator.next();
                        if(task.equals(FlowOnt.getFlowTask(context, indivualFlow)))
        {
                                if(FlowOnt.flowDirection == IN_DEGREE){
                                        hierarchicalTask.setTask(task);
                                        hierarchicalTask.setGenericTaskName(presentLevel);
                                        startFlow = FlowOnt.getFlowOperator(context,
indivualFlow);
                                        hierarchicalTask.setFlowOperator(startFlow);

        hierarchicalTask.setBasicFlowOperatorName(startFlow.getSuperClass());
```

```
                                tasksvisited.add(task);
                                break;
                        }
                }
        }

        if(startFlow== null)
                return;



        for(Iterator<OntClass> i = flowsCopy.iterator(); i.hasNext();){
                OntClass iflow = i.next();
                if(!task.equals(FlowOnt.getFlowTask(context, iflow)) &&
FlowOnt.getFlowOperator(context, iflow).equals(startFlow)){
                        if(FlowOnt.flowDirection ==OUT_DEGREE){
                                if(!tasksvisited.contains(FlowOnt.getFlowTask(context,
iflow))){


    hierarchicalTask.addTaskNext(FlowOnt.getFlowTask(context, iflow));
                                }
                        }
                }
        }
        if(hierarchicalTasks.get(presentLevel)==null){
                ArrayList<HierarchicalTask> hierarchicalTaskArray = new
ArrayList<HierarchicalTask>();
                hierarchicalTasks.put(presentLevel, hierarchicalTaskArray);
        }
        hierarchicalTasks.get(presentLevel).add(hierarchicalTask);
        for(OntClass tn : hierarchicalTask.getTasksNext()){
                System.out.println(tn.getLocalName());
                if(tn != null)
                        if(tn.getLocalName() !="End"){
                                popualteDescendents(tn, presentLevel+1);
                        }

        }
    }
```

4.4.2.2 Updating relationships in ontology

Whenever a new business process is to be populated in the ontology, the system always compares this business process with each of business processes already persisted in the ontology other the business process itself to update the associations. The association may be one of the three types: equivalent, part of and overlapped under each of three categories: content based, structural, and complete.

For instance, to compare two graphs for structural equivalence, a method, isStructurallyMatch, is invoked which takes both graphs as inputs and return to the main function a Boolean value depends on which, the ontology will be updated later. Up to now, we saw that we can compare the structural relationship between two business processes using our proposed algorithm. But now how can we update the business process ontology if we get a structural relationship between two business processes. Since ontology property is used to represent the relationship between resources, we adopted this concept of representing relationship in our business process ontology as well. A set of Java classes provided by Jena is used in our application to manipulate the ontology properties handily. For instance, to create an ontology property for structural equivalence relationship, we used the method, createObjectProperty method of Jena ontology model.

To update the ontology with the structural equivalence relationship between two business processes abp1 and abp2, a method called addStructuralEquivalence is invoked, which exploit the java classes of Jena to update the relationship based on ontology property.

To create an object property of structural equivalence, the following java statement has been coded.

```
ObjectProperty isStructEquiv = context.GetOntModel().CreateObject
Property(ABPUpdate.ontologyURI + "#isStructurallyEquivalentOf");
```

To find the components of all values from restriction on object property isStructEquiv, we have a snippet of code using Jena API as follows.

```
        for (ExtendedIterator<OntClass> i = abp1.listSuperClasses(true); i.hasNext();)
{
                OntClass oc = i.next();
                if (oc.isRestriction()) {
                    Restriction res = oc.asRestriction();
                    if (res.onProperty(isSpecialisation)) {

                        ocListSpec =
res.asAllValuesFromRestriction().getAllValuesFrom().as(UnionClass.class);

                    }
                }

        }
```

As the business processes abp1 and abp2 are structurally equivalent, each instance of the business process abp1 is also equivalent to each of the individual of the business process abp2. To update the ontology with the structural equivalence relationships among the individuals of the business process abp1 and abp2, we wrote a small set of codes as follows.

```
        for (ExtendedIterator<? extends OntResource> i = abp1.listInstances();
i.hasNext();) {
                                Individual indi1 = i.next().asIndividual();
                                for (ExtendedIterator<? extends OntResource> i2 =
abp2.listInstances(); i2.hasNext();) {
                                        Individual indi2 = i2.next().asIndividual();
                                        indi1.addProperty(isStructEquiv, indi2);
                                }

                        }
```

Chapter 5

Validation of the work done

After the prototype implementation, it is important to validate for measuring its efficiency and rationality. Since we proposed many algorithms and their corresponding implementations using Java language, multifarious business process examples were being sought to validate our application. After being found a handsome number of business processes in BPMN notion, we used our abstract business process extraction mechanism to make them compatible with our prototype.

5.1 Experiments

Using the file upload interface, we imported all the files, those describes the business processes in our hand. We did not see any anomalies defining concepts and updating ontologies with relationship found except some typos. Then, we executed all queries of every category for each business process stored in the ontology, and obtained the results for each. Based the experiment done, we illustrated the results using visual tool - math lab- from the different perspectives for the sake of visual clarification.

Serial #	Business Process Name	Short name	# of tasks in task ontology	# of flows in flow ontology
1	Procure Article	abp1	6	10
2	OnlinePurchase	abp2	5	8
3	JobPosting	abp3	6	10
4	BookShop	abp4	6	10
5	Media selection	abp5	6	10
6	EmployeeJoiningFitnessPriority	abp6	10	18
7	CreditApplication	abp7	7	13
8	Recruitment	abp8	7	13
9	SoftwareRequirementChange	abp9	7	13
10	BankTransfer	abp10	7	13
11	Discussion	abp11	7	13
12	DinnerPreparation	abp12	8	14
13	BicycleManufatory	abp13	8	14
14	Enrollment	abp14	8	14
15	AdwordsToSearchEngine	abp15	8	15
16	LeaveApplicationFormProcess	abp16	8	15
17	Advertisement	abp17	9	17
18	TravelBooking	abp18	9	17
19	CarRental	abp19	10	19
20	EmployeeJoining	abp20	10	19
21	ColorectalCancerReferral	abp21	10	20
22	LoanSanction	abp22	10	22
23	ElearningCenter	abp23	11	22
24	BookSub	abp24	13	25
25	CreditApplicationSub	abp25	15	28

Table - 1 : List of Business processes stored in the ontology

To make the experiment result more comprehensible, let us consider the following table that describes the list of business processes, known with given short names, stored in the ontology with the number of task ontologies and the number flow ontologies for each process. It seems a bit bizarre that the total number of flow ontology is quite more than that of the task ontology. For example, the total number of tasks present in the process, JobPosting, is 6, but the number of flows is 10, however the number flows expressed in the BPMN diagram is 8. Let us explain the cause behind the scenery with the BPMN diagram of JobPosting.

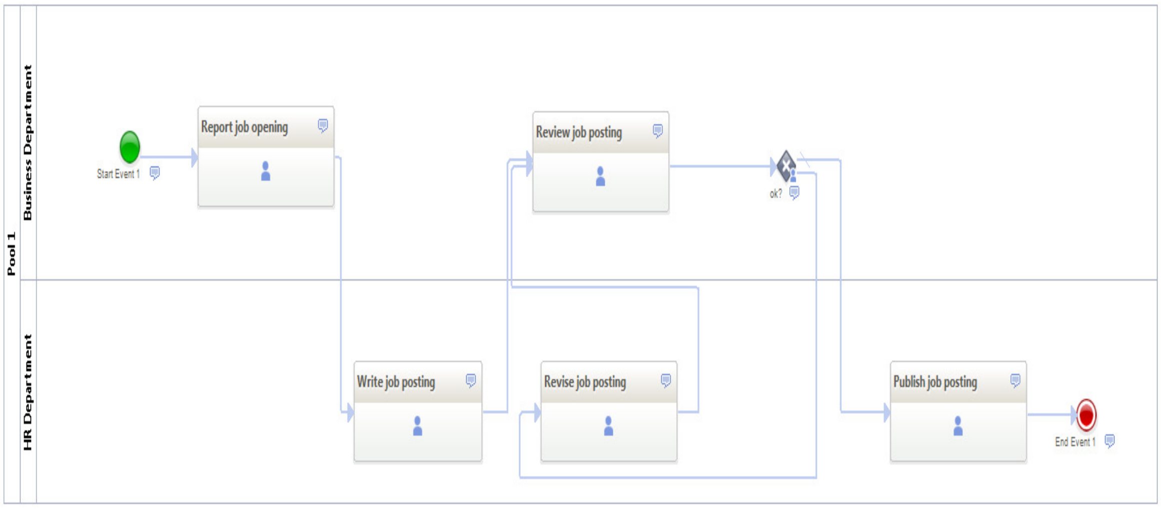

Fig - 5(1) : BPMN diagram of JobPosting

The number of flows increases in the ontology due to the sequence operator defined in our application. If a task is connected directly with another task, then, we consider that the operator lie between these two tasks is sequence operator. For example, the task Report job opening and the task Write job posting are connected directly with each other. So, it seems there is no operator lie in between. But, with our approach, a hidden sequence diagram is present between these two tasks. Consequently, every other tasks those are connected directly with each other have the same affect. Beside, we obviate the cyclic nature of the process, so we cut off the connection lie between tasks Revise job posting and ORSplit operator. In a nutshell, the number flow ontologies is more than the flows in a business process due to the presence of hidden sequence operator between two directly connected tasks represented by the BPMN diagram.

The ratio of the number of task ontology to number of flow ontology could impose some extra operations in our algorithm, however, the affects we believe is just from the calculation point of view rather the matching output as a whole. To illustrate the phenomenon, we kept a list of the processes and did graphical analysis retaining the list in X axis of the graph. Let us have a look the list graphically to observe how much increment is there due to the hidden sequence operator present in each business process of the ontology.

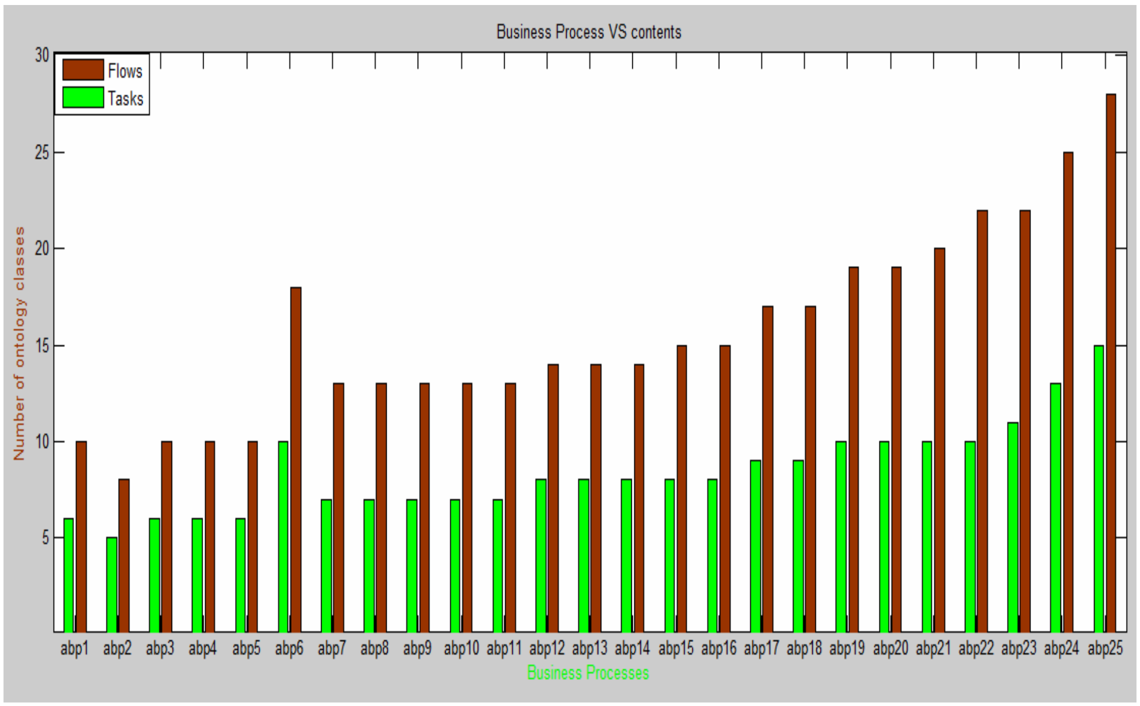

Fig - 5(1) : Business processes and number of tasks and flows in ontology

In the graph above, we observe that the increment of the number of flow ontologies is remained approximately double of the number of task ontologies of the each business process in the ontology due to the sequence operator defined my the application.

5.2 Results

5.2.1 Total number similarities for each business process

The total number of similarities – combining all relationships found for each category for every business process- of twenty five different business process concepts stored in the ontology is shown in the following graph. Structural overlapping is the most common and simple similarity, which is found among business processes, those having at least an activity to be executed – that has been defined as task ontology in our business process model. Therefore, it is also common in finding the number of structural overlapping similarities between the business processes maximum among other associations.

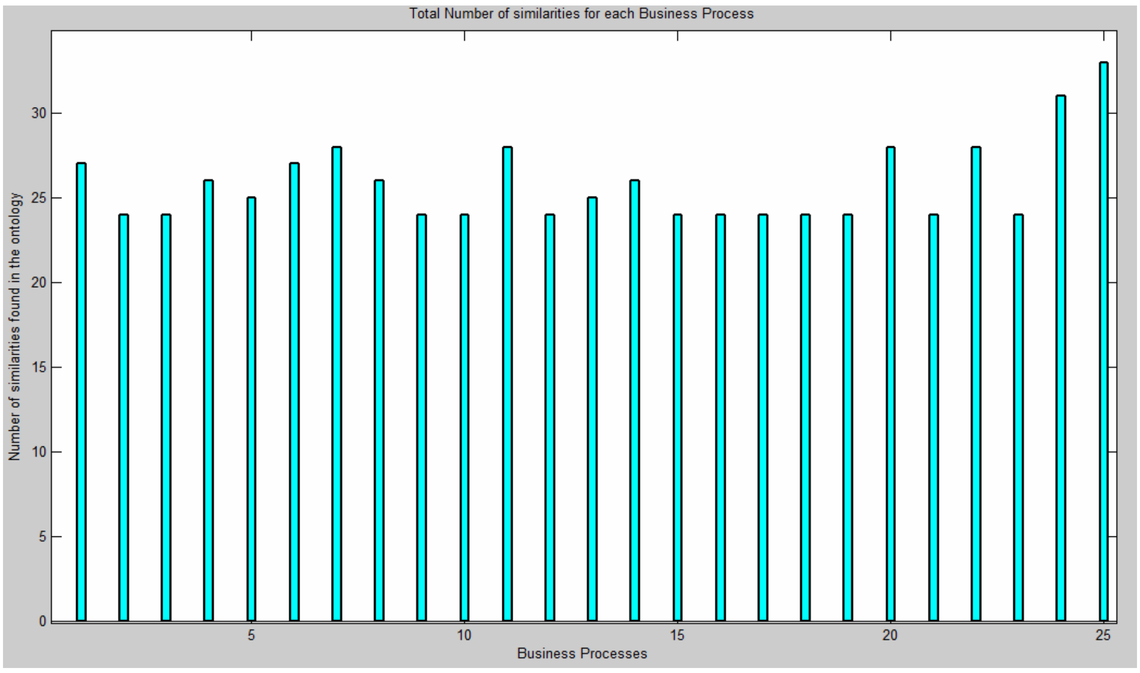

Fig - 5(2) : Total number similarities for business process

We can observe from the above graph that every business process, abp#, is structurally overlapped with every other business process stored in the ontology, and therefore, the number of structurally overlapped similarities of each business process is the number of business processes stored available in the ontology minus 1.Hence we set the base line of the graph 24, which is the number of structurally overlapped relationships exists in the ontology for the each of the business processes[abp1, abp2, ….,abp25].

We can also see form the illustrated graph that total number of relationships found in the ontology for the disparate business processes, processes with a wide range of activities and domain, are varied from business process to business process depending on the similarities found in different categories. For better clarification, let us see few more graphs those describe the variation in the number of relationships based on different categories.

5.2.2 Categorized associations' overview

We basically underscored the proposed algorithms in this thesis for the relationships between the business processes in the broad categories: Completeness in the similarity, content of the business processes present and the structural similarities. Consequently, we emphasized the experiments to adhering to our aim of the thesis. And we got the resultant data we concurred to our algorithms proposed. For example, to show how and to what extend our proposed algorithms can benefit is show by the graph prepared from the resultant data, we got from our application using a handsome quantity of business processes.

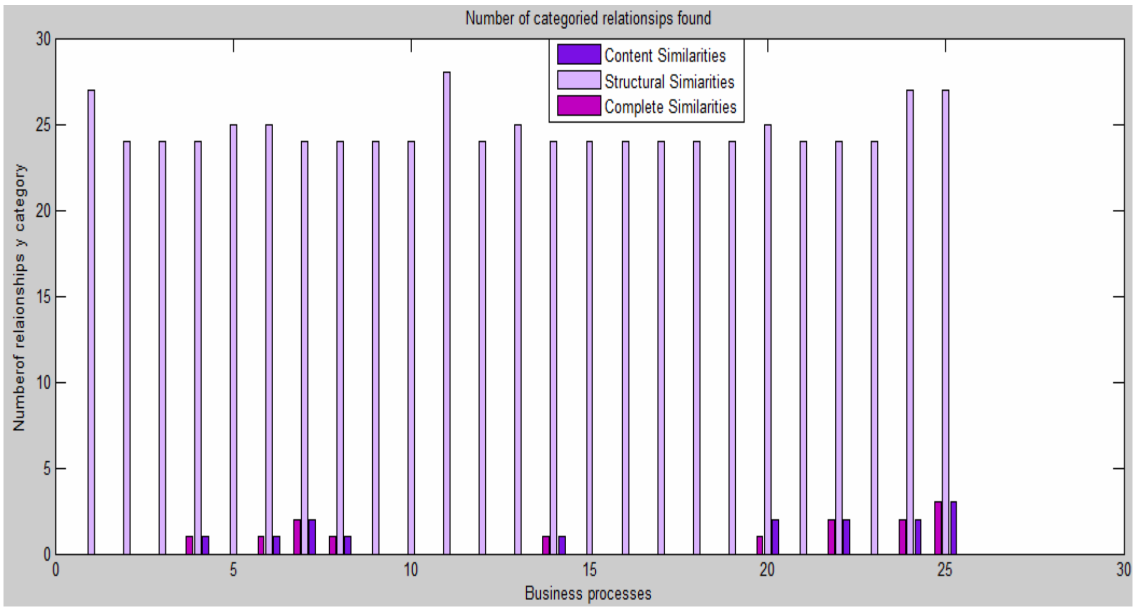

Fig - 5(3) : Number of categorized relationships VS Business Process

From the domination of the green line in the graph construes that the availability of the structural similarities in business process ontology is quite high. However, it is also difficult to overlook the amount of similarities between business processes based on contents. On the other, complete similarity is quite difficult to match and it is also inferred to the number of processes found. As the complete similarity is not just only finding structural and content similarity, it is worth mentioning that the sequence of tasks defined in the ontology does matter as well. Finding the number of similar processes is important based on the three different categories, Nevertheless, finding which sort of relationships with which they are related to is also crucial. For example, a business process, for which we are looking similar processes, defined can have similar contents in another business process in the ontology. Obviously, we can exploit the contents, tasks in the abstract business process, but, it would be fantastically beneficial, if it also have some sort structural similarities between the processes.

5.2.3 Associations based on each specific type

After explaining the total overview, now we are interested in more specific look for each type of associations, we proposed. In order to delve into the relationships- how and to which extend, business processes may be related - is explained with the following sub sections.

5.2.3.1 Structural similarities among Business processes

From the structural point of view, each business process is structurally overlapped with other, if each of them has at least a task to be carried out. Therefore, a straight green line is seen in the graph, which expresses that each business process has the same degree of possibility of being structurally overlapped, as there is no empty business process, process without any task, defined in the ontology.

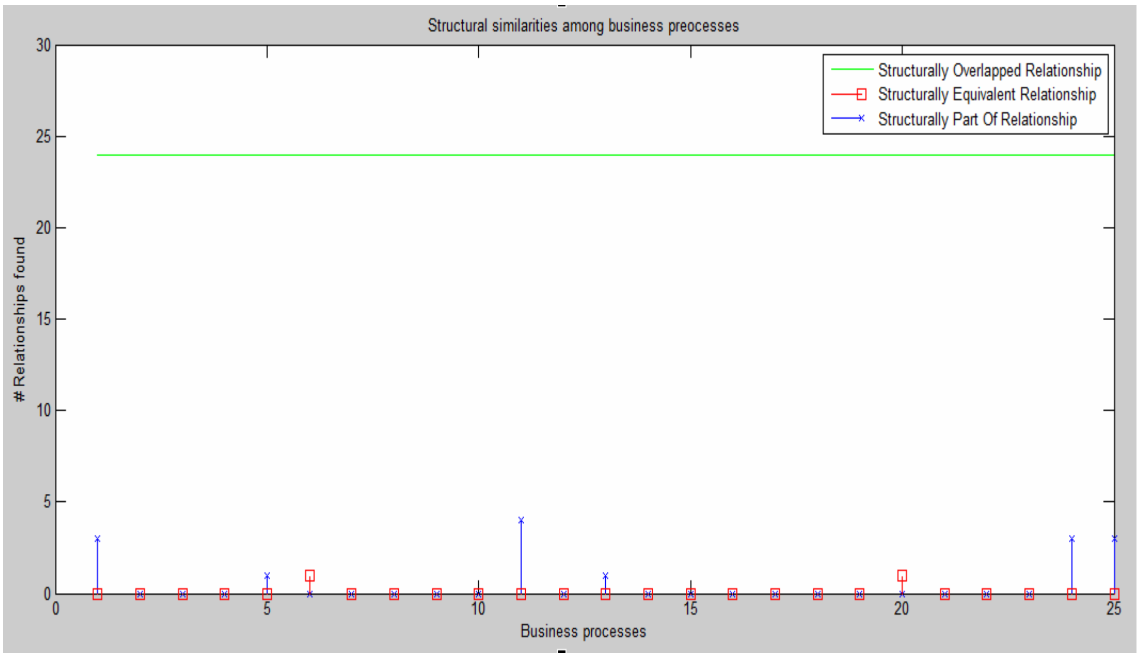

Fig - 5(4) : Structural similarities among Business processes

There were several structural part of relationships found in the ontology, but structural equivalence were difficult to find with this limited set of business processes- collected from disparate sources- available in the ontology. For the sake of the algorithmic congruency, we defined a structural equivalent process as an experimental task, to validate our application's ability to perform structural equivalence comparison. Complete equivalence is rare to find for its three dimensional search criteria. Nevertheless, both structural equivalence and content based equivalence together may serve a great opportunity to employ an existing business process with some modifications without of being tired of preparing it again. In order see how is it probable to meet near about similar process to engulf, it is crucial to check also how contents are similar with each other. To better illustrate it, let's have a look the following graph with a bit explanation.

5.2.3.2 Content Based similarities among Business processes

In the following graph, the content overlapping found in the business processes CreditApplication, LoanSanction are CreditApplicationSub are prominent. Since the actions involved in theses processes lie under the same domain, i.e. actions related to banking, it is likely to have similarities among some of actions of these business processes. It would be more likely to get content based overlapping similarities, if we could add other processes such as purchase on credit, loan management in our ontology too. However, the probability is still depends on how the activities are defined in the business process ontology, i.e. how we defined the activities in XML or BMP files. In addition, if the same activities carried out in different business processes are defined in the input file with different name and contents, it would be unfeasible to guaranty the success of our approach. In this case, convention in defining input files could play a vital role.

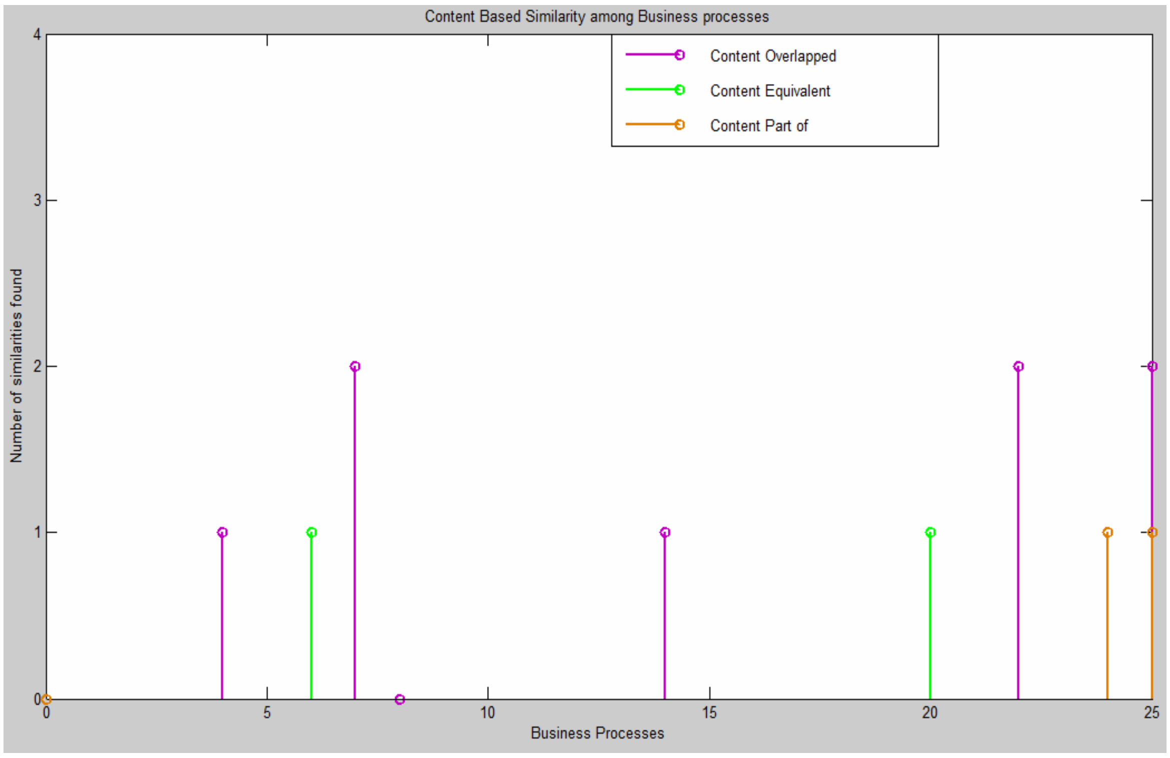

Fig - 5(5) : Content Based similarities among Business processes

Though the theoretical view and simulation accorded with the approach, we proposed, it should be scrutinized well to define a convention of preparing input files so that a long range and diversified sources of business processes in the pragmatic usages could utilize the application with the destined goal precisely.

5.2.3.3 Complete similarities among business processes in the ontology

Before going through the description of the graph drawn below, let us discuss about the algorithms, we used in our application. Content based algorithm solely depends on the constituents of the process ontology, while the complete matching is done taking content similarities, structural similarities as well as sequence of the activities in the concrete business process. Nevertheless, it is apparent that the graph for complete similarities and content based similarities are same except two green lines present on the business process 6 and 20 of content based similarity graph. However, the fact is that the process which has complete relationship is also has content based relationship, but the reverse is not true.

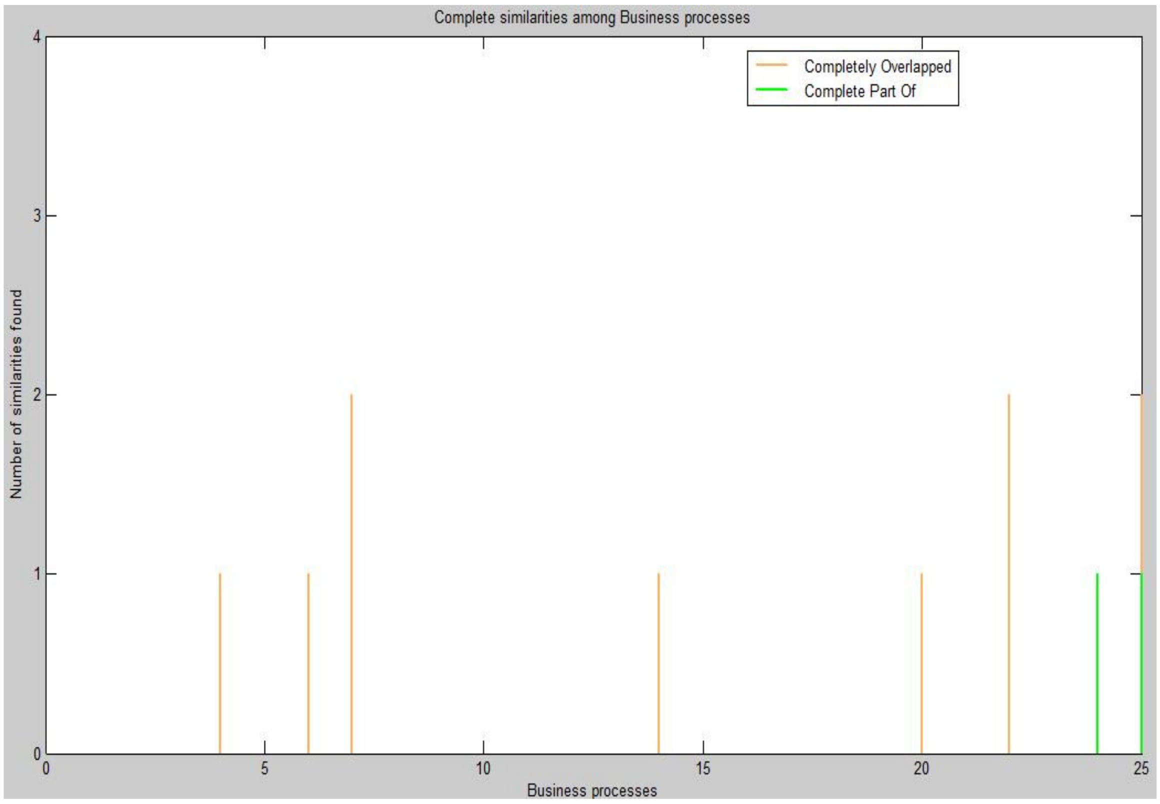

Fig - 5(6) : Complete similarities among Business processes

The business processes, denoted by 6 and 20 on the X axis, in the complete similarity graph have overlapping relationships, and it is also reflected in the content based overlapping associations. The same business processes are equivalent in the content based similarity graph, but, they are not equivalent based on completeness due to the concept defined and implementation done based on different algorithms in our thesis work.

5.2.4 Big Picture at a glance

In the above description, we tried to clarify how the business process ontology can be utilized discovering business processes with our application and the amount of relationship found in the total discovery. Besides, it would be interesting to see how each process is related in 9 different types of associations with each other process in the ontology. The following graph, where both axes labelled business processes, illustrates how the relationships are with different markings.

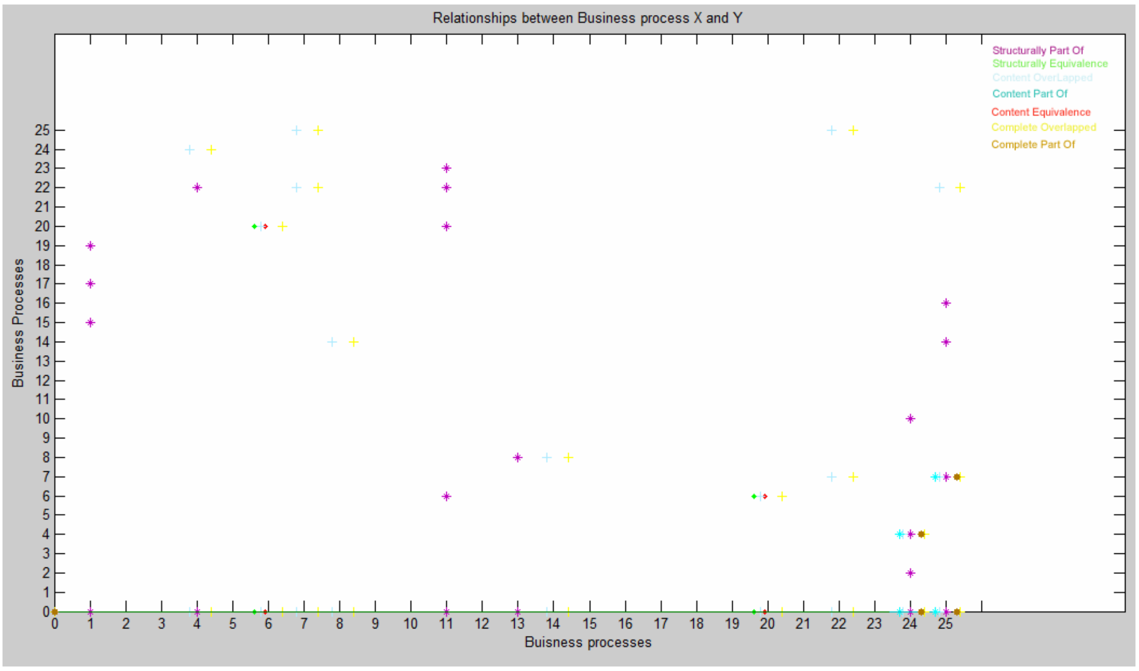

Fig - 5(7) : Total associations among business processes at a glance

From fig: constant line, we could say that the most number of similarities found is of structural overlapping. However, for the sake of visual unequivocalness, the graph is kept clairvoyant avoiding the structural overlapping completely. It is perspicuous in the graph that, the maximum number of relationships, marked by violet starts, among business processes obtained is for structural part of. For example, Discussion,11 ticked, and BookSub, 24 ticked, in the X axes are contained structurally in 4 and 3 processes respectively shown in the presentation. However, we see other type associations between business processes in the scatter graph as well. For a better view, another picture of this graph in different angle is also given below.

In the graphs above, we can have a better look of the associations lie between the business processes. In our thesis work, we would be gratified, if we could add some graphical tools to provide visual presentation of the associations our application can represent. With these tools, users could manage viewing all associations at a glance with a pithy compact way without searching through all the searching criteria present in the current application. Nevertheless, we are looking forward to someone, who can extend the work we just mentioned so that it could not only make us happy but also made convenient for the users exploiting the application.

6. Conclusion

In this thesis report, we discussed about the storage of the business processes, their relationships, and discovery of the processes from the repositories in the background and the related works section. Then, we briefly illustrate the overview of the abstract business process and their relationships. Afterward, we paraphrased the semantic approach of the abstract business process and ontology for storing the semantically described processes from the publication, on which the thesis is based on. The extended works on the relationships between the concepts are described and proceeded to explain our proposed algorithms using pseudo code in two subsequent sections. Finally the technological aspects and implementation of the proposed work have been described.

To validate the work done, we used twenty five disparate business processes in our prototype to get the resultant data for analysis. The conformity of the prototype has been met after analysing those date, and, hence, the proof of this is illustrated in chapter 5. From the result we observed, we can infer that a handsome amount of abstract business processes, business processes described abstractly, stored in the ontology can provide a useful catalogue of business processes of diversified knowledge domain from which a users can search business processes of their interest and associations they prefer to.

Semantically described business processes are more convenient to store abstractly in the ontology than the traditional approach of storing the concrete one. An abstract business process represents a class of concrete business processes those have the same knowledge domain regardless of the execution place of the processes. The concept of abstract business process, which facilitates the reuse of the business process by searching the processes semantically, can obviate the onerous task of implementing the business processes again which already exist in the ontology. Using the business process searching GUI, users can search the ontology specifying the association: structural, content based or even complete category type as their requirements, and consequently get the result of that type.

Future works include optimization of the algorithms proposed for structural and complete associations, a tool for query results presentable in graphs, and extension of the work so that cyclic nature of the business process activities can be taken into account. Since the algorithms for structural and complete association based on the hierarchical tree that is constructed from semantically described processes in the ontology, the reduction of the complexity regarding the graph construction and traversing would be a useful pathway to the optimization of the algorithms. For the sake of simplicity, the cyclic nature of the business processes is disregarded in our thesis work. So the consideration of cyclic graph while comparing the business processes would be an interesting topic for the future work. To have a look at a glance of the search results and to interact visually, a graphical tool can be developed taking references from the validation chapter, where graphs are used for validation purpose for showing various associations among the business processes for analysis.

7. Bibliography

1. Marco Brambilla, Khalid Belhajjame. Ontological Description and Similarity-based Discovery of Business Process Models
2. Business Process management 100 success secretes by Gerard Blokdiijk
3. Business process management: practical guidelines to successful implementation By John Jeston, Johan Nelis.
4. Zhiqiang Yan, Remco Dijkman, Paul Grefen. Business Process Model Repositories - Framework and Survey
5. G. Yang. Process library. Data & Knowledge Engineering
6. Injun Choi, Kwangmyeong Kim and Mookyung Jang. An XML-Based Process Repository and Process Query Language for Integrated Process Management.
7. Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes.
8. Antoon Goderis, Peter Li, and Carole A. Goble. Workow discovery: the problem, a case study from e-science and a graph-based solution.
9. Chengfei Liu, Xuemin Lin, Xiaofang Zhou, Maria Orlowska Building a Repository for Workflow Systems.
10. Song M, Miller JA, Arpinar IB. 2001. REPOX: an XML repository for workflow designs and specifications.
11. A Translation Approach to Portable Ontology Specifications
12. Business Process Management, concepts, languages and architecture by Mathias weske, Hasso Plattner Institut an der Universität Postdam.
13. http://www.w3.org/TR/owl-ref/. OWL Web Ontology Language Reference.
14. http://www.webratio.com/
15. http://jena.sourceforge.net/documentation.html. Jena – A Semantic Web Framework for Java.
16. clarkparsia.com/pellet/. Pellet: The Open Source OWL 2 Reasoner