Sādhanā Vol. 36, Part 4, August 2011, pp. 441–461. © Indian Academy of Sciences

Complex networks and SOA: Mathematical modelling of granularity based web service compositions

S CHATLA[1], S KADAM[1], D KOLLURU[1], S SINHA[1], A VISWANDHUNI[1] and A VAIDYA[2,*]

[1]Prithvi Information Solutions Limited, Prithvi Corporate School, Hyderabad 500 032, India
[2]Department of Mathematical Sciences, Montclair State University, Montclair, NJ 07043, USA
e-mail: vaidyaa@mail.montclair.edu

MS received 9 July 2010; revised 19 January 2011; accepted 14 June 2011

Abstract. Service Oriented Architecture (*SOA*) can be defined as a way of defining and implementing enterprise applications that deals with the intercommunication of loosely coupled, coarse grained (*business level*), reusable artifacts (*services*). In this paper, we attempt to mathematically model the preliminary steps in the larger problem of providing an optimal architecture. The problem is treated as a complex network, particularly a process-task-network. We employ statistical and graph-theoretic methods namely, Jaccard's distance analysis, Multiple Correspondence method and the Minimum Spanning Tree method, to find appropriate clusters. These methods are used to cluster tasks across business processes to propose services. Additional properties and features of these clusters are discussed. We propose a leverage factor which demonstrates the importance of a task within the service and its impact on service composition.

Keywords. Complex networks; SOA; Jaccard's distance; macro clusters; leverage.

1. Introduction

The study of complex networks in sociological, biological and technological contexts is definitely on the rise. While the interdisciplinary characteristics of network science continues to attract scientists from various disciplines (Porter *et al* 2006; Porter *et al* 2007; Zarei & Samani 2009), it has yet to be applied to the area of software architecture which is itself a relatively new area. This paper is devoted to the modelling aspects of the larger problem of Service Oriented architecture as a complex network (Albert & Barabassi 2002; Newman 2003; Pastor–Satorras & Vespigiani 2001) and the application of several techniques from this field to understand a highly nonlinear and sophisticated problem.

*For correspondence

Software Architecture of a computing system can be defined by a set $\{C, P, R\}$. The set C is the set of components of system. The set P represents the externally visible properties of different components in C and set R represents the relationships between different components. The externally visible properties essentially define the input and output parameters of different components. The quality of any software system is determined by its architecture, and hence designing software architecture is a very important and critical activity in the development of a software project.

Services oriented architecture (SOA) has become an industry buzz word in the recent times. In SOA the software components are essentially software services. SOA provides methods for software system development and integration where systems group functionality around different business processes and package these functionalities as a set of inter-operable services. SOA also describes an IT infrastructure which allows different applications to exchange data with one another as they participate in different business processes. SOA aims at a loose coupling of services with operating systems, programming languages and other technologies which underlie applications. SOA separates functions into distinct units, or services, which developers make accessible over a network in order that users can combine and reuse them in the production of business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. It is a new method of developing and deploying complex business software applications.

Though the technique was discussed in the early 60s, its use was limited for the first 30 years as the software was relatively less complex. With the industries using computer technology widely, the software in recent times became increasingly complex and the need to integrate disparate systems grew drastically. SOA is becoming one of the preferred modes of software development and deployment in this scenario.

Web services provide one way of implementing SOA. A web service is a software program that can take the required input in an accepted standard and provide the desired output in the same. So, the input and output formats of the software service can be understood by any other software or human anywhere in the world even though they do not understand how the input is processed to get the output. Web services make functional building blocks accessible over standard Internet protocols independent of platforms and programming languages. These services can be new applications or just wrapped around existing legacy systems to make them network enabled. Web services use Extensible Mark up Language (XML) messages that follow Simple Object Access Protocol (SOAP) standard. Every web service is described by Web Service Description Language (WSDL) that defines exact formats of inputs and outputs in XML format.

For a business, what this means is that they can change their processes freely as demanded by the market. The software needed to support the new process can be created by simply restructuring the existing services. So, SOA solves one of the biggest challenges faced by the business community; namely, aligning the existing software systems with changing business requirements. In reality, this problem is so severe that businesses stall process improvements because of the cost of software up-gradation. When an organization decides to go through the path of SOA, the first challenge it faces is to define the services correctly. Service must perform one or more tasks of the organization and interact with other services through the standard protocol.

We can see how the service definition can affect the performance of the SOA with a simple example. Theoretically, there are two limits to service definition. At the finest granularity, each task in the organization may be built as one service. However, this results in a huge number of services even for a small organization. In any real scenario, multiple versions of the same service will be needed for various processes and complexity of managing is an exponential function of the number of services. At the coarsest granularity, the entire task universe can be defined as one

service and this results in zero flexibility in defining new processes defeating the entire purpose of SOA. So, the optimum distribution is somewhere in between.

Currently, software architects and business users work together to define the optimum service matrix for the organization or a single application. Quite a bit of this work is done intuitively. However, this approach has obvious limitations like lack of enough number of expert architects in an organization, differences in design decisions based on subjective perspectives, incorrect definitions for extremely complex systems, etc.

2. Problem formulation and methodology

The present work is an effort to develop an approach based on formal principles to define the optimum service granularity. In other words, our aim is to develop a set of mathematical tools that help the IT department of an organization identify which tasks should be combined to form which services. While we recognize that first principle approach can never replace the expert's intuition in real world complex problems, it definitely can provide a framework for the experts to design robust systems more accurately.

Our proposed long term approach will have the following steps:

(i) Develop a language/notation for representing tasks and processes for SOA analysis: Tasks and processes happen at the physical level and the services are designed at the software development layer. We need a common notation to describe and equate them.
(ii) Define Macro-clusters: A typical organization has thousands of tasks. For the final service definition, many parameters (redundancy, cost, reliability) need to be optimized. The inputs needed for computing these parameters include inter-task interaction (energy of combining, cost of combining, etc.). In most cases, it is not feasible to compile such inputs for all the tasks. So, as a second step, we identify the set of tasks that are most amenable to form a cluster purely from the ease of connectivity. We term these as macro-clusters.
(iii) We then compile the inter-task inputs between only those tasks within a macro-cluster.
(iv) Optimize the quality of services to finalize service granularity: Literature identified the parameters that define the performance of services. We apply these to the macro-clusters and provide an optimization framework to define the final service structure from these macro task clusters.

These can become inputs to the software architect who can then figure out an architecture in which these services can be deployed for that organization.

For the purpose of SOA, any organization can be viewed as a collection of tasks and processes. Tasks are the smallest quantum of work performed by either an individual or a software within the organization to achieve part of the objectives of the organization. Process is a collection of one or more tasks that define one complete activity. So, Leave management of employees is a process and applying for a leave is a task in that process. So, we can visualize the tasks as nodes on a graph and processes as lines(edges) connecting those nodes. By definition, a single node can be a part of multiple processes (see figure 1). Using this definition, a set of tasks qualify themselves as a service if all the nodes represented by those tasks are common to a large number of processes. Another candidate of a service is if a set of nodes are exclusive to a single process.

The current literature does not contain much in terms of a mathematical model for SOA from the fundamental process-task viewpoint. The bulk of previous studies are focused on issues relating to the quality of service (QoS) which pertains to the step 4 from our list above. Yu *et al* (2007) have studied the end-to-end QoS issues of composite services by utilizing a QoS broker

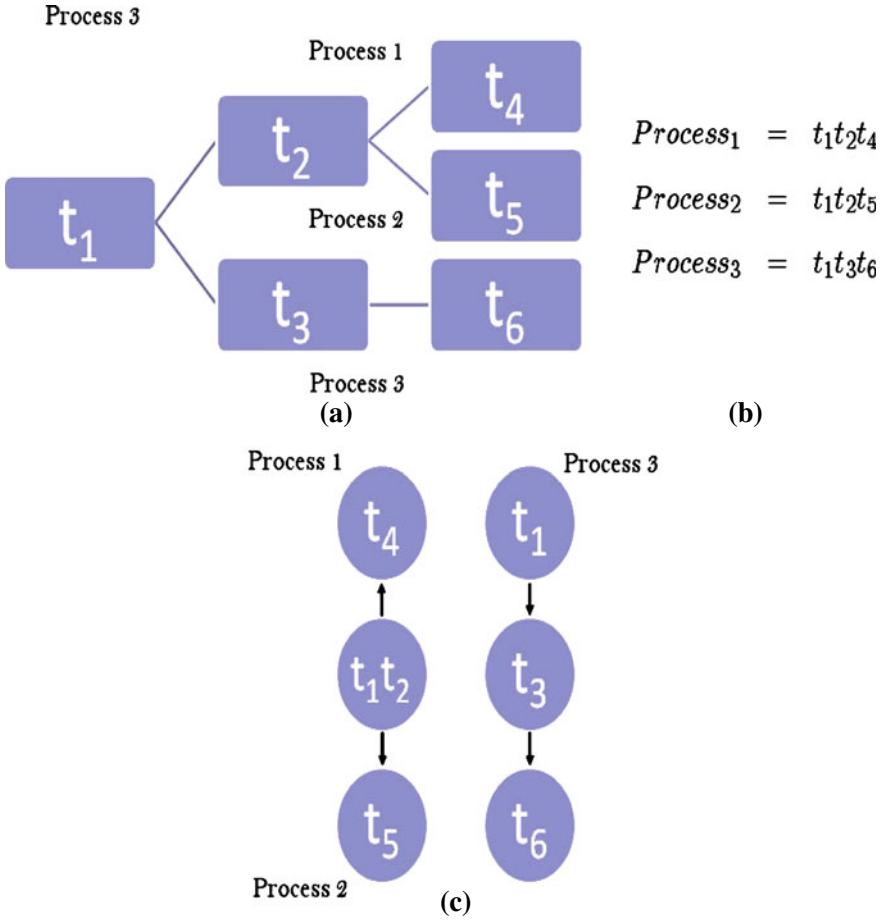

Figure 1. A sample sketch of processes and tasks in a workflow. Figure (**a**) shows the workflow picture, (**b**) shows the process–task picture i.e., the decomposition of the workflow into processes and tasks and (**c**) shows the service composition picture.

that is responsible for selecting and coordinating the individual service component. They have proposed two solution approaches viz. combinatorial (Multiple Choice Knapsack Problem) and Graph approach (constrained shortest path problem) to the Service selection problem. Yanchuk *et al* (2008) have proposed a mathematical foundation for leveraging service oriented programming paradigm which effects the way people build software systems. In the paper (Yu *et al* 2007) QoS Capable Web Service Composition has been studied. Lang *et al* (2008) have analysed the impact of SOA paradigm on the structure of the value chain in the software industry. They have considered SOA Service Integrator(SeI) to select suppliers and decide which services to integrate which is modelled as a linear integer 0-1 programming problem.

In this paper, we discuss steps 1 and 2 of our approach to the big problem which are mentioned above. As is shown in the following pages even the task of mapping the problem at hand to a mathematical language and clustering it, even with known methods, can be a daunting task here, let alone the problem of finding an optimal architecture. To show that the first step of clustering, at various granularities, is effective using our approach, we deem it better to consider several

arbitrary examples instead of a few realistic ones since different realistic examples are very likely to show considerable differences in their structure. Additionally, the difficulty in finding realistic examples cannot be underestimated since organizations are not as easily willing to share details about their functioning as one might imagine. Our current abstract approach holds the additional example of allowing for understanding how variations in structure of an organization can affect its service compositions; such experiments cannot be easily conducted with realistic examples. We model an organization as consisting of a set of processes and a set of tasks within the processes to attain strategic goals. Macro clusters are identified using the Jaccard's distance, Multiple Correspondence Analysis and Minimum Spanning Tree techniques. A real time example is discussed as a case study. The different methods are compared and contrasted permitting us the choice of the most appropriate method to be used in the event architecture problem.

Following the literature, we state the definitions of some frequently appearing terminologies in this paper: a *service* is a collection of tasks which are related to a functional context established by the service (Erl 2007), the term *granularity* is used to communicate the level of (or absence of) detail associated with some aspect of software program design and finally the granularity of the service's functional scope, as determined by its functional context, is simply referred to as *service granularity* (Erl 2007). While the above definition of service sounds simple enough, it is to be noted that in applications, the assignment of a service is a very complex task and is intricately tied to the overall architecture itself (Kadam *et al* 2010a), so not all clusters are 'service-worthy'. Therefore the problem of SOA is extremely challenging and the importance of a careful initial approach to establishing a proper abstract methodology, mapping to the real problem, good clustering algorithms and reliability measures on our approach, cannot be underestimated.

Our first method is the Jaccard's distance method which is used to measure the association between tasks. A binary process-task matrix is built based on the availability of task (denoted 1) or absence of it (denoted 0). The second method employed is MCA (Greencare 1984; Greencare & Balasius 1994) which is a simple graphical method. We consider the occurrence or non-occurrence binary matrix and compute the Burt Matrix for further analysis which yields the desired clusters. The final analysis is based on the Minimum Spanning Tree (MST) (Zahn 1971) technique which is a graph-theory based method where we consider adjacent tasks and take the frequency of occurrence of the task in all the processes. We take the complement of the frequencies of every pair of tasks and apply MST. Upon deleting maximum weight edges from the MST we can come up with corresponding clusters.

3. Process-task representation

In the rest of the paper, we define a convention to describe a business workflow. As mentioned briefly earlier, our overall business structure consists of several *processes*, each of which can be broken into more fundamental units that we call *tasks* which may be repeated across different processes. This particular decomposition of the work flow (W) can be better understood if one thinks of the process (P_i) as being a linear path with no branches; every branch would simply be another process. Therefore,

$$W = \cup_{i=1}^{N} P_i.$$

This is not to say that the process is completely independent, they could intersect at one or multiple nodes or tasks. *It is the nature of the intersection that we are seeking to understand in this paper.* More specifically, we are interested in finding the cluster set $\mathcal{C} = \{c_{kl}^m = P_k \cap P_l : 1 < k, l < N \text{ and } c_{kl}^m \neq \phi, m \in \mathbf{N}\}$. The superscript m indicates that the intersection of two processes might result in more than one cluster of tasks. The dimension of each cluster, $\dim(c_{kl}^m)$

is referred to as the *granularity*. Each cluster is a collection of two or more tasks and is currently defined in such a way that the ordering is not important, that is for instance the cluster $t_i t_j$ is not distinct from the cluster $t_j t_i$. In practice, the identification of the tasks in any process is in itself a difficult problem and perhaps this is the most difficult task that a SOA analyst performs. Tasks are to be chosen in such a way that: (i) they can collectively represent all the processes, (ii) all tasks are completely independent and therefore $t_i \cap t_j = 0$ for any i, j not the same. With this formal definition in place, we are set to analyse our problem at hand. We will proceed in the rest of the paper with the assumption that for any example that we might consider, such a decompositions of tasks exist.

4. Clustering

4.1 *Example 1*

In this section, we illustrate our argument with an abstract, randomly chosen example having ten different processes and fifteen different tasks.

$$\begin{aligned}
P_1 &= t_2 t_4 t_5 t_6 t_9 t_{13} t_{14} t_{15} \\
P_2 &= t_1 t_4 t_5 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{14} t_{15} \\
P_3 &= t_1 t_4 t_6 t_7 t_{10} t_{14} t_{15} \\
P_4 &= t_2 t_3 t_5 t_9 t_{12} t_{13} t_{14} t_{15} \\
P_5 &= t_2 t_3 t_4 t_6 t_7 t_9 t_{10} t_{11} t_{12} \\
P_6 &= t_1 t_5 t_8 t_{10} t_{12} t_{13} \\
P_7 &= t_6 t_7 t_8 t_{10} t_{11} t_{12} t_{14} \\
P_8 &= t_6 t_8 t_9 \\
P_9 &= t_1 t_3 t_4 t_5 t_7 t_9 t_{13} \\
P_{10} &= t_3 t_7 t_8 t_{11} t_{12} t_{13} t_{15}.
\end{aligned}$$

We can now discuss the results of our cluster analysis using the methods mentioned above.

4.1a *Jaccard's distance*: Standard cluster analysis method suggests the following algorithm: (i) Identify the pair of points that have the minimum distance among all. This pair will form the first cluster. (ii) Calculate the modified distance matrix using metric distance

$$(a, b + c) = \min\{distance(a, b), distance(a, c)\}.$$

(iii) Identify the points, which have the minimum distance to make a cluster and finally (4) repeat steps 2–3 until all the points are clustered. There are some similarity measures for binary variables viz. Simple-matching, Double-matching, Rogers–Tanimoto, Russell–Rao, Jaccard's, etc. (Timm 2002). Amongst these measure, we find Jaccard's the most appropriate as it calculates the 1–1 matches between the binary variables.

The Jaccard's distance approach (Hartigan 1975; Chalak *et al* 2006; Zahn 1971) which is based on Boolean algebra to find the distance between the tasks i.e., the columns of the matrix given in table 1. The Jaccard's distance is defined by

$$\text{Jaccard's distance} = 1 - S_{ij},$$

Table 1. Process–Task coefficient binary matrix pertaining to Example 1.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
P_1	0	1	0	1	1	1	0	0	1	0	0	0	1	1	1
P_2	1	0	0	1	1	0	1	1	1	1	1	1	0	1	1
P_3	1	0	0	1	0	1	1	0	0	1	0	0	0	1	1
P_4	0	1	1	0	1	0	0	0	1	0	0	1	1	1	1
P_5	0	1	1	1	0	1	1	0	1	1	1	1	0	0	0
P_6	1	0	0	0	1	0	0	1	0	1	0	1	1	0	0
P_7	0	0	0	0	0	1	1	1	0	1	1	1	0	1	0
P_8	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
P_9	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0
P_{10}	0	0	1	0	0	0	1	1	0	0	1	1	1	0	1

where S_{ij} is the Jaccard's coefficient of similarity and it can be calculated as

$$S_{ij} = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}. \tag{1}$$

This is a hierarchical method where every task will be clustered with remaining tasks at some stage. We define a cut-off distance to validate the clusters.

For the binary data matrix in table 1 we compute a symmetric distance matrix using Jaccard's distance norm (equation (1)) for all the columns (tasks). The distance matrix in table 2 is given as input for cluster analysis. We use the MATLAB program (MathWorks Inc.) to perform our cluster analysis and the results are provided below. We define α to be a threshold value such that if the distance between any points is greater than or equal to α, then it would not be ideal to combine those tasks. See figure 2 for a Dendrogram representing clusters of granularity 2 and 3 based on the Jaccard distance approach.

Table 2. The Jaccard's distance matrix based on equation (1) for Example 1.

0	1	0.857	0.5	0.5	0.875	0.571	0.71	0.75	0.5	0.857	0.75	0.714	0.71	0.714
1	0	0.6	0.667	0.667	0.667	0.875	1	0.5	0.86	0.833	0.714	0.667	0.67	0.667
0.85714	0.6	0	0.714	0.714	0.875	0.571	0.88	0.57	0.88	0.667	0.571	0.5	0.88	0.714
0.5	0.6667	0.714	0	0.571	0.571	0.429	0.89	0.43	0.57	0.714	0.778	0.75	0.57	0.571
0.5	0.6667	0.714	0.571	0	0.889	0.778	0.75	0.43	0.75	0.875	0.625	0.333	0.57	0.571
0.875	0.6667	0.875	0.571	0.889	0	0.625	0.75	0.63	0.57	0.714	0.778	0.889	0.57	0.75
0.57143	0.875	0.571	0.429	0.778	0.625	0	0.63	0.67	0.43	0.333	0.5	0.778	0.63	0.625
0.71429	1	0.875	0.889	0.75	0.75	0.625	0	0.78	0.57	0.5	0.429	0.75	0.75	0.75
0.75	0.5	0.571	0.429	0.429	0.625	0.667	0.78	0	0.78	0.75	0.667	0.625	0.63	0.625
0.5	0.8571	0.875	0.571	0.75	0.571	0.429	0.57	0.78	0	0.5	0.429	0.889	0.57	0.75
0.85714	0.8333	0.667	0.714	0.875	0.714	0.333	0.5	0.75	0.5	0	0.333	0.875	0.71	0.714
0.75	0.7143	0.571	0.778	0.625	0.778	0.5	0.43	0.67	0.43	0.333	0	0.625	0.63	0.625
0.71429	0.6667	0.5	0.75	0.333	0.889	0.778	0.75	0.63	0.89	0.875	0.625	0	0.75	0.571
0.71429	0.6667	0.875	0.571	0.571	0.571	0.625	0.75	0.63	0.57	0.714	0.625	0.75	0	0.333
0.71429	0.6667	0.714	0.571	0.571	0.75	0.625	0.75	0.63	0.75	0.714	0.625	0.571	0.33	0

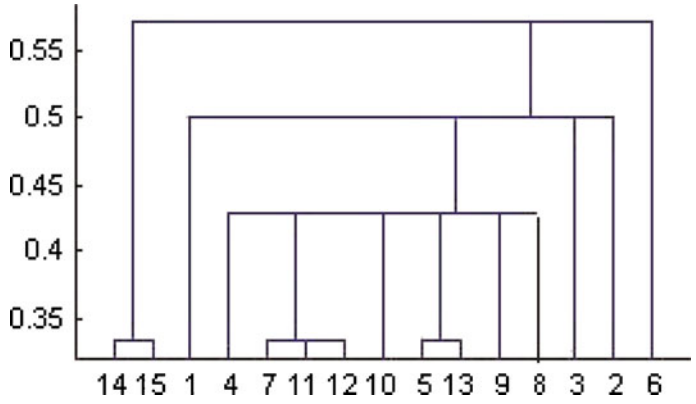

Figure 2. Dendrogram representing clusters based on the Jaccard distance pertaining to Example 1.

The clusters for (i) granularity 2 with threshold value of 0.8 (threshold $= 1 - \frac{granularity}{number of processes}$) are:

$\{t_1, t_4\}, \{t_1, t_5\}, \{t_1, t_7\}, \{t_1, t_8\}, \{t_1, t_9\}, \{t_1, t_{10}\}, \{t_1, t_{12}\}, \{t_1, t_{13}\}, \{t_1, t_{14}\}, \{t_1, t_{15}\}, \{t_2, t_3\},$
$\{t_2, t_4\}, \{t_2, t_5\}, \{t_2, t_6\}, \{t_2, t_9\}, \{t_2, t_{12}\}, \{t_2, t_{13}\}, \{t_2, t_{14}\}, \{t_2, t_{15}\}, \{t_3, t_4\}, \{t_3, t_5\}, \{t_3, t_6\},$
$\{t_3, t_7\}, \{t_3, t_9\}, \{t_3, t_{11}\}, \{t_3, t_{12}\}, \{t_3, t_{13}\}, \{t_3, t_{15}\}, \{t_4, t_5\}, \{t_4, t_6\}, \{t_4, t_7\}, \{t_4, t_9\}, \{t_4, t_{10}\},$
$\{t_4, t_{11}\}, \{t_4, t_{12}\}, \{t_4, t_{13}\}, \{t_4, t_{14}\}, \{t_4, t_{15}\}, \{t_5, t_7\}, \{t_5, t_8\}, \{t_5, t_9\}, \{t_5, t_{10}\}, \{t_5, t_{12}\}, \{t_5, t_{13}\},$
$\{t_5, t_{14}\}, \{t_5, t_{15}\}, \{t_6, t_7\}, \{t_6, t_8\}, \{t_6, t_9\}, \{t_6, t_{10}\}, \{t_6, t_{11}\}, \{t_6, t_{12}\}, \{t_6, t_{14}\}, \{t_6, t_{15}\}, \{t_7, t_8\},$
$\{t_7, t_9\}, \{t_7, t_{10}\}, \{t_7, t_{11}\}, \{t_7, t_{12}\}, \{t_7, t_{13}\}, \{t_7, t_{14}\}, \{t_7, t_{15}\}, \{t_8, t_9\}, \{t_8, t_{10}\}, \{t_8, t_{11}\}, \{t_8, t_{12}\},$
$\{t_8, t_{13}\}, \{t_8, t_{14}\}, \{t_8, t_{15}\}, \{t_9, t_{10}\}, \{t_9, t_{11}\}, \{t_9, t_{12}\}, \{t_9, t_{13}\}, \{t_9, t_{14}\}, \{t_9, t_{15}\}, \{t_{10}, t_{11}\},$
$\{t_{10}, t_{12}\},$
$\{t_{10}, t_{14}\}, \{t_{10}, t_{15}\}, \{t_{11}, t_{12}\}, \{t_{11}, t_{14}\}, \{t_{11}, t_{15}\}, \{t_{12}, t_{13}\}, \{t_{12}, t_{14}\}, \{t_{12}, t_{15}\},$
$\{t_{13}, t_{15}\}, \{t_{14}, t_{15}\},$

and (ii) granularity 3 with threshold value of 0.7 are:

$\{t_1, t_4, t_5\}, \{t_1, t_4, t_7\}, \{t_1, t_4, t_9\}, \{t_1, t_4, t_{13}\}, \{t_1, t_4, t_{15}\}, \{t_1, t_5, t_7\}, \{t_1, t_5, t_8\}, \{t_1, t_5, t_9\},$
$\{t_1, t_5, t_{10}\}, \{t_1, t_5, t_{12}\}, \{t_1, t_8, t_{10}\}, \{t_1, t_8, t_{11}\}, \{t_1, t_8, t_{12}\}, \{t_1, t_8, t_{14}\}, \{t_1, t_{10}, t_{12}\},$
$\{t_1, t_{10}, t_{14}\}, \{t_1, t_{10}, t_{15}\}, \{t_2, t_3, t_9\}, \{t_2, t_3, t_{12}\}, \{t_2, t_5, t_9\}, \{t_2, t_5, t_{13}\}, \{t_2, t_5, t_{14}\}, \{t_2, t_5, t_{15}\}$
$\{t_2, t_6, t_9\}, \{t_2, t_9, t_{13}\}, \{t_2, t_9, t_{14}\}, \{t_2, t_9, t_{15}\}, \{t_3, t_7, t_{11}\}, \{t_3, t_7, t_{12}\}, \{t_3, t_7, t_{13}\}, \{t_3, t_9, t_{13}\}$
$\{t_4, t_5, t_7\}, \{t_4, t_6, t_7\}, \{t_4, t_{14}, t_{15}\}, \{t_5, t_9, t_{13}\}, \{t_6, t_7, t_{10}\}, \{t_7, t_8, t_{11}, \{t_7, t_{11}, t_{12}\}, \{t_8, t_{10}, t_{12}\}$
$\{t_8, t_{10}, t_{14}\}, \{t_8, t_{11}, t_{12}\}, \{t_8, t_{12}, t_{14}\}, \{t_8, t_{12}, t_{15}\}\{t_9, t_{10}, t_{11}\}, \{t_9, t_{14}, t_{15}\}, \{t_{10}, t_{11}, t_{12}\},$
$\{t_{11}, t_{12}, t_{14}\}$
$\{t_{12}, t_{13}, t_{15}\}, \{t_{13}, t_{14}, t_{15}\}.$

4.1b *Multiple correspondence analysis*: Correspondence analysis is a statistical method for visualizing the associations between the levels of a two-way contingency table. It is a geometric technique which displays the rows and columns as points in low dimensional space, such that the position of the row and column points is consistent with their associations in the table. If we have multiple categorical variables then it is called Multiple Correspondence Analysis (MCA).

Table 3. The Burt matrix for Example 1.

4	0	1	3	3	1	3	2	2	3	1	2	2	2	2
0	3	2	2	2	2	1	0	3	1	1	2	2	2	2
1	2	4	2	2	1	3	1	3	1	2	3	3	1	2
3	2	2	5	3	3	4	1	4	3	2	2	2	3	3
3	2	2	3	5	1	2	2	4	2	1	3	4	3	3
1	2	1	3	1	5	3	2	3	3	2	2	1	3	2
3	1	3	4	2	3	6	3	3	4	4	4	2	3	3
2	0	1	1	2	2	3	5	2	3	3	4	2	2	2
2	3	3	4	4	3	3	2	6	2	2	3	3	3	3
3	1	1	3	2	3	4	3	2	5	3	4	1	3	2
1	1	2	2	1	2	4	3	2	3	4	4	1	2	2
2	2	3	2	3	2	4	4	3	4	4	6	3	3	3
2	2	3	2	4	1	2	2	3	1	1	3	5	2	3
2	2	1	3	3	3	3	2	3	3	2	3	2	5	4
2	2	2	3	3	2	3	2	3	2	2	3	3	4	5

Greencare (1984) suggested that the analysis of a data matrix is similar to its Burt matrix, $B = M^T M$, where M is the Coefficient matrix given in table 1 where the matrix B is always real and symmetric. Since a real and symmetric matrix always has a singular value decomposition or spectral decomposition, this approach will work for any problem. The Burt matrix corresponding to Example 1 is given in table 3. The single valued decomposition for B is computed and then from the resulting matrices, we take the eigenvectors corresponding to the most significant eigenvalues i.e., those which contribute a sufficiently high percentage of information. In the Example 1, the first two eigenvectors are found to contain 66.09 % of the total information are the only ones considered here.

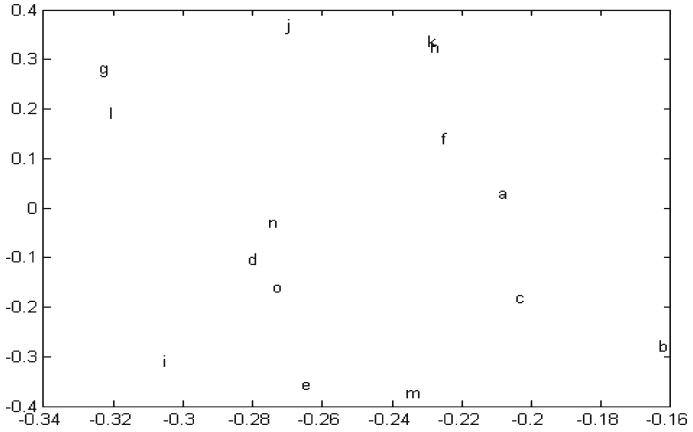

Figure 3. Eigen-vector plot for Example 1 based on MCA. The tasks are represented in the plot by alphabets, 'a' representing t_1, 'b' representing t_2 and so on. Clusters are identified by overlapping characters or those very close to each other.

We consider a scatter plot of the two eigen-vectors as shown in figure 3 and the clusters can be identified by inspection from the graph. The MCA method yields the following clusters: (i) at granularity 2 we get

$\{t_1, t_4\}, \{t_1, t_5\}, \{t_1, t_7\}, \{t_1, t_8\}, \{t_1, t_9\}, \{t_1, t_{10}\}, \{t_1, t_{12}\}, \{t_1, t_{13}\}, \{t_1, t_{14}\}, \{t_1, t_{15}\}, \{t_2, t_3\},$
$\{t_2, t_4\}, \{t_2, t_5\}, \{t_2, t_6\}, \{t_2, t_9\}, \{t_2, t_{12}\}, \{t_2, t_{13}\}, \{t_2, t_{14}\}, \{t_2, t_{15}\}, \{t_3, t_4\}, \{t_3, t_5\}, \{t_3, t_9\},$
$\{t_3, t_{13}\}, \{t_3, t_{15}\}, \{t_4, t_5\}, \{t_4, t_9\}, \{t_4, t_{10}\}, \{t_4, t_{13}\}, \{t_4, t_{15}\}, \{t_5, t_7\}, \{t_5, t_9\}, \{t_5, t_{12}\}, \{t_5, t_{13}\},$
$\{t_5, t_{14}\}, \{t_5, t_{15}\}, \{t_6, t_7\}, \{t_6, t_8\}, \{t_6, t_9\}, \{t_6, t_{10}\}, \{t_6, t_{11}\}, \{t_6, t_{12}\}, \{t_7, t_8\}, \{t_7, t_9\}, \{t_7, t_{10}\},$
$\{t_7, t_{11}\}, \{t_7, t_{12}\}, \{t_7, t_{13}\}, \{t_8, t_{10}\}, \{t_8, t_{11}\}, \{t_8, t_{12}\}, \{t_8, t_{14}\}, \{t_8, t_{15}\}, \{t_9, t_{12}\},$
$\{t_9, t_{14}\}, \{t_9, t_{15}\}, \{t_{10}, t_{11}\}, \{t_{10}, t_{12}\}, \{t_{11}, t_{12}\}, \{t_{13}, t_{15}\}, \{t_{14}, t_{15}\}$

and (ii) at granularity 3 we get

$\{t_1, t_3, t_4\}, \{t_1, t_3, t_{15}\}, \{t_1, t_3, t_{14}\}, \{t_1, t_4, t_{15}\}, \{t_1, t_4, t_{14}\}, \{t_1, t_{14}, t_{15}\}, \{t_1, t_2, t_5\}, \{t_1, t_2, t_{13}\}$
$\{t_1, t_2, t_9\}, \{t_1, t_2, t_3\}, \{t_1, t_2, t_4\}, \{t_1, t_2, t_{15}\}, \{t_1, t_2, t_{14}\}, \{t_1, t_5, t_{13}\}, \{t_1, t_5, t_9\}, \{t_1, t_5, t_3\}$
$\{t_1, t_5, t_4\}, \{t_1, t_5, t_{15}\}, \{t_1, t_5, t_{14}\}, \{t_1, t_{13}, t_9\}, \{t_1, t_3, t_{13}\}, \{t_1, t_4, t_{13}\}, \{t_1, t_{13}, t_{15}\}, \{t_1, t_{13}, t_{14}\}$
$\{t_1, t_9, t_3\}, \{t_1, t_9, t_4\}, \{t_1, t_9, t_{15}\}, \{t_1, t_9, t_{14}\}, \{t_2, t_5, t_{13}\}, \{t_2, t_5, t_9\}, \{t_2, t_5, t_3\}, \{t_2, t_5, t_4\}$
$\{t_2, t_5, t_{15}\}, \{t_2, t_5, t_{14}\}, \{t_2, t_{13}, t_3\}, \{t_2, t_{13}, t_4\}, \{t_2, t_{13}, t_{15}\}, \{t_2, t_{13}, t_{14}\}, \{t_2, t_9, t_3\}, \{t_2, t_9, t_4\}$
$\{t_2, t_9, t_{15}\}, \{t_2, t_9, t_{14}\}, \{t_3, t_4, t_{15}\}, \{t_3, t_4, t_{14}\}, \{t_4, t_{15}, t_{14}\}, \{t_5, t_{13}, t_9\}, \{t_8, t_{11}, t_{10}\}$
$\{t_8, t_{11}, t_7\}, \{t_8, t_{11}, t_{12}\}, \{t_8, t_{11}, t_6\}, \{t_8, t_{10}, t_7\}, \{t_8, t_{10}, t_{12}\}, \{t_8, t_{10}, t_6\}.$

4.1c *Minimum spanning tree*: A spanning tree of a graph G(V, E), of V vertices, is a set of $|V|$-1 edges(E) that connect all vertices of the graph. A spanning tree of a graph is a subgraph that contains all the vertices and is a tree (a graph in which any two vertices are connected by exactly one path). A graph may have many spanning trees (Wu & Chao 2004).

Given a connected, undirected graph, a spanning tree of that graph is a subgraph which is a tree and connects all the vertices together. We can also assign a weight to each edge, which is a number representing how unfavourable/favourable it is and use it to find the weights of the spanning trees by computing the sum of the weights of the edges. A MST or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of MSTs for its connected components (Narsingh 2005).

For the Example 1 we have made a combined graph for all the processes (figure 4). We have used GrIn (GRIN,2010). In table 1, t_4 and t_5 occurred together in P_1, P_2 and P_9, so weight of the edge = total number of processes − number of times adjacent tasks occur together $= 10 - 3 = 7$.

The clusters can be identified by the MST in the graph by the bold lines. The edges of the MST are $\{t_1, t_5\}, \{t_5, t_9\}, \{t_9, t_{13}\},$
$\{t_9, t_{12}\}, \{t_{11}, t_{12}\}, \{t_{10}, t_{12}\}, \{t_7, t_{10}\}, \{t_{12}, t_{14}\}, \{t_{14}, t_{15}\}, \{t_8, t_{11}\}, \{t_6, t_9\}, \{t_1, t_4\}, \{t_3, t_7\}, \{t_2, t_4\}.$

From the obtained MST we delete edges with highest weights or second highest weights and so on till we get the clusters we are satisfied with. By deleting the highest weight edges we get the subgraphs which are the required clusters. Thus, on deleting high weight edges we automatically get different granularity clusters. Different Spanning Trees will give different clusters of different granularity, not necessarily minimum. One may ask for composition of services of different granularities which can be obtained from these Spanning Trees (Grygorash *et al* 2006).

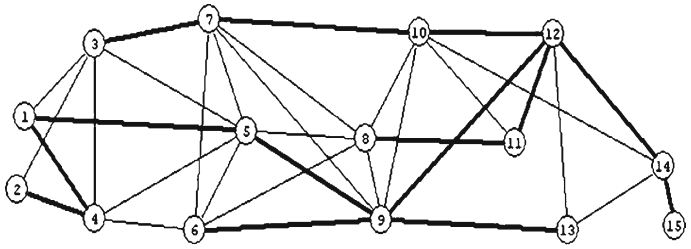

Figure 4. MST graph for Example 1. Bold edges denote the MST to consider for clustering.

The MST method yields the following clusters: (i) at granularity 2 we get

$\{t_1, t_4\}, \{t_1, t_5\}, \{t_2, t_4\}, \{t_3, t_7\}, \{t_5, t_9\}, \{t_6, t_9\}, \{t_7, t_{10}\}, \{t_8, t_{11}\}, \{t_9, t_{12}\}, \{t_9, t_{13}\}$
$\{t_{10}, t_{12}\}, \{t_{14}, t_{15}\}$

and (ii) at granularity 3 we get

$\{t_1, t_4, t_5\}, \{t_1, t_5, t_9\}, \{t_1, t_2, t_4\}, \{t_3, t_7, t_{10}\}, \{t_5, t_6, t_9\}, \{t_5, t_9, t_{12}\}, \{t_5, t_9, t_{13}\}, \{t_6, t_9, t_{12}\}$
$\{t_6, t_9, t_{13}\}, \{t_7, t_{10}, t_{12}\}, \{t_8, t_{11}, t_{12}\}, \{t_9, t_{12}, t_{13}\}, \{t_{10}, t_{11}, t_{12}\}, \{t_{10}, t_{12}, t_9\}, \{t_{10}, t_{12}, t_{14}\}$
$\{t_{11}, t_{12}, t_{14}\}, \{t_{12}, t_{14}, t_{15}\}.$

4.2 *Example 2*

Our previous example was based on randomly generated set of processes and tasks. It needs to be seen if the techniques employed above give realistic answers in a real life scenario. For this reason, in this section, we consider a real time software with a set of processes and tasks as defined below. The software addresses the needs of a user to search and download required mathematical or statistical functions as well as allowing the user to request addition of customized functions (figures 5 and 6).

1. **Process 1:** Registration
   Tasks: Registration (t_1), Login (t_2), Password validation (t_3), Logout (t_{21}).

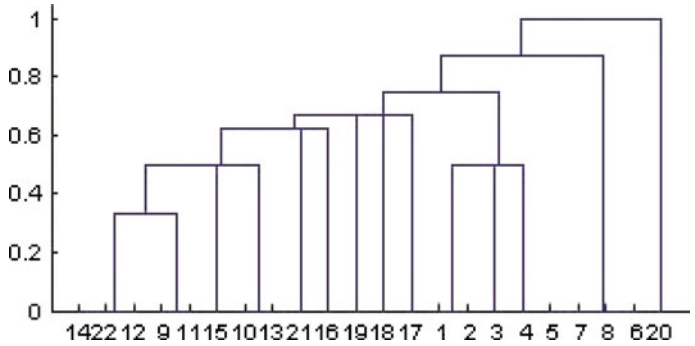

Figure 5. Dendrogram showing clusters pertaining to Example 2.

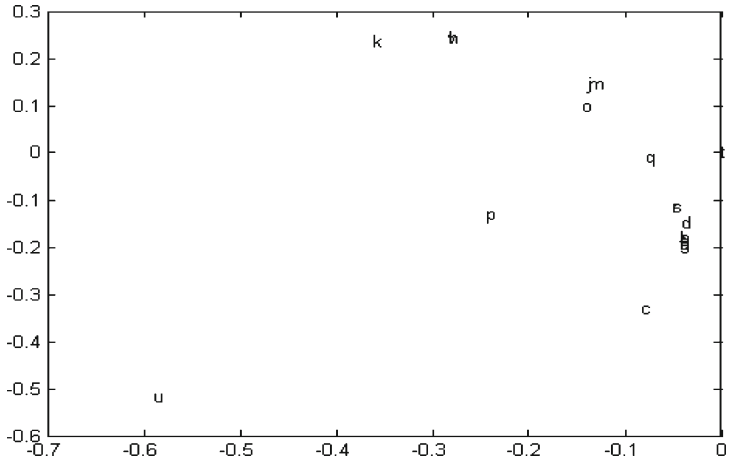

Figure 6. Eigen-vector plot based on MCA for Example 2. As done for Example 1, the alphabet characters are to be identified with tasks.

2. **Process 2:** Change Password
   Tasks: Enter new password (t_4), Password validation (t_3), Logout (t_{21}).

3. **Process 3:** The model developer is adding new models
   Tasks: Display the requests of users (t_5), Upload the dll (t_6), Upload the documentation (t_7), Upload the bench marking of the dll (t_8), Logout (t_{21}).

4. **Process 4:** Search for existing models/functions - user
   Tasks: Create/select project (t_9), Save project (t_{10}), Search (t_{11}), Download existing function (t_{12}), Save search (t_{13}), Received the notification (t_{22}), Provide feedback after downloading (t_{14}), Logout (t_{21}).

5. **Process 5:** Request Admin - user
   Tasks: Create project (t_9), Search (t_{11}), Place request to Admin (t_{15}), Received the notification (t_{22}), Sees the display of the updated request (t_{16}), Download function (t_{12}), Received the notification (t_{22}), Provide feedback after downloading (t_{14}), Logout (t_{21}).

6. **Process 6:** User inputs to Admin for approval - user
   Tasks: Create project (t_9), Search (t_{11}), Input domain (t_{17}), Logout (t_{21}).

7. **Process 7:** Coordinating development and usage - Admin
   Tasks: See the display of requests (t_{16}), Approvals (t_{18}), Logout (t_{21}).

8. **Process 8:** Making an expert system
   Tasks: See the display of requests (t_{16}), Assign weights (t_{19}), Logout (t_{21}).

9. **Process 9:** Invite Friend - user
   Tasks: Invite a friend (t_{20}), Logout (t_{21}).

4.2a *Jaccards distance*: The results of the Jaccard's method: (i) at granularity 2 with threshold of 0.78 are;

$\{t_9, t_{11}\}, \{t_9, t_{12}\}, \{t_9, t_{14}\}, \{t_9, t_{22}\}, \{t_{11}, t_{12}\}, \{t_{11}, t_{14}\}, \{t_{11}, t_{22}\}, \{t_{12}, t_{14}\}, \{t_{12}, t_{22}\}, \{t_{14}, t_{22}\}$

(ii) and at granularity 3 with threshold of 0.67 are;

$\{t_1, t_2, t_3\}, \{t_1, t_2, t_4\}, \{t_2, t_3, t_4\}, \{t_5, t_7, t_8\}, \{t_5, t_6, t_7\}, \{t_5, t_6, t_8\}, \{t_7, t_8, t_6\}, \{t_9, t_{11}, t_{14}\}$
$\{t_9, t_{11}, t_{22}\}, \{t_9, t_{11}, t_{12}\}, \{t_9, t_{12}, t_{14}\}, \{t_9, t_{12}, t_{22}\}, \{t_9, t_{14}, t_{22}\}, \{t_{10}, t_{13}, t_{15}\}, \{t_{11}, t_{12}, t_{14}\}$
$\{t_{11}, t_{12}, t_{22}\}, \{t_{12}, t_{14}, t_{22}\}, \{t_{12}, t_9, t_{14}\}, \{t_{12}, t_9, t_{22}\}.$

4.2b *MCA*: We identify the following clusters from the scatter plot of the eigen-vectors (see table 4):

(i) at granularity 2,

$\{t_9, t_{11}\}, \{t_9, t_{12}\}, \{t_9, t_{14}\}, \{t_9, t_{22}\}, \{t_{11}, t_{12}\}, \{t_{11}, t_{14}\}, \{t_{11}, t_{22}\}, \{t_{12}, t_{14}\}, \{t_{12}, t_{22}\}, \{t_{14}, t_{22}\}$

(ii) and at granularity 3

$\{t_1, t_2, t_5\}, \{t_1, t_2, t_7\}, \{t_1, t_2, t_8\}, \{t_1, t_2, t_6\}, \{t_5, t_6, t_7\}, \{t_5, t_6, t_8\}, \{t_5, t_7, t_8\}, \{t_6, t_7, t_8\}$
$\{t_9, t_{11}, t_{12}\}, \{t_9, t_{11}, t_{14}\}, \{t_9, t_{11}, t_{22}\}, \{t_9, t_{12}, t_{14}\}, \{t_9, t_{12}, t_{22}\}, \{t_9, t_{14}, t_{22}\}, \{t_{10}, t_{15}, t_{13}\}$
$\{t_{11}, t_{12}, t_{14}\}, \{t_{11}, t_{12}, t_{22}\}, \{t_{11}, t_{14}, t_{22}\}, \{t_{12}, t_{14}, t_{22}\}.$

4.2c *MST*: The edges of MST are $\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_{21}\}, \{t_{21}, t_{22}\}, \{t_{21}, t_{20}\}, \{t_{21}, t_{16}\}, \{t_{21}, t_{14}\}, \{t_{14}, t_{13}\}, \{t_{21}, t_{19}\}, \{t_{21}, t_{18}\}, \{t_{21}, t_{17}\}, \{t_{16}, t_{15}\}, \{t_{14}, t_{12}\}, \{t_{12}, t_{11}\}, \{t_{11}, t_9\}, \{t_{11}, t_{10}\}, \{t_{21}, t_8\}, \{t_8, t_7\}, \{t_7, t_6\}, \{t_6, t_5\}, \{t_3, t_4\}$. Using this approach in addition to the clusters found using the above two methods, we also have the sub graphs: $\{t_5, t_6, t_7, t_8\}$, $\{t_1, t_2, t_3, t_4\}$, $\{t_9, t_{11}, t_{12}\}$, $\{t_{20}, t_{21}, t_{22}\}$, $\{t_{13}, t_{14}\}$ (figure 7).

The MST method yields the following clusters: (i) at granularity 2 we get

$\{t_9, t_{11}\}, \{t_{11}, t_{12}\}, \{t_{12}, t_{14}\}$

and (ii) at granularity 3 we get

$\{t_1, t_2, t_3\}, \{t_2, t_3, t_4\}, \{t_5, t_6, t_7\}, \{t_6, t_7, t_8\}, \{t_9, t_{11}, t_{10}\}, \{t_9, t_{11}, t_{12}\}, \{t_{11}, t_{12}, t_{14}\}$
$\{t_{12}, t_{14}, t_{13}\}.$

Table 4. Eigen-vectors corresponding to the software case study problem.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
x	−0.043	−0.043	−0.0835	−0.0404	−0.0432	−0.0432	−0.0432	−0.0432	−0.3621	−0.1384	−0.3621
y	−0.1815	−0.1815	−0.331	−0.1495	−0.1934	−0.1934	−0.1934	−0.1934	0.23538	0.14627	0.23538
	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}
x	−0.2831	−0.1384	−0.2831	−0.1448	−0.2451	−0.079	−0.0502	−0.0502	0	−0.5891	−0.2831
y	0.24364	0.14627	0.24364	0.09737	−0.1313	−0.0083	−0.1143	−0.1143	0	−0.5177	0.24364

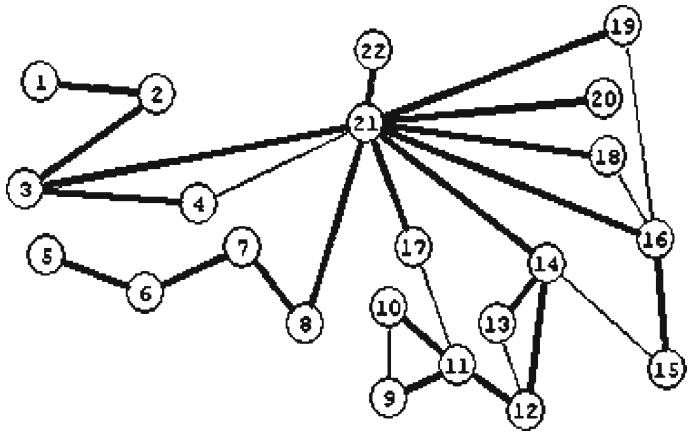

Figure 7. MST graph for Example 2. Bold edges denote the MST to consider for clustering.

The table 5 shows a comparison of the clusters found using the three different methods at different granularities. The results are for granularity 2 and 3 for examples 1 and 2. We can obtain higher order granularities also. Since the number of processes and tasks in the examples considered are fairly small. We also compare the results of our computations to the actual clusters in these examples. *Note that by actual clusters we mean two or more tasks in a process (corresponding to granularity) that recur at any distance from each other in multiple processes*.

While we have presented the details of just two examples above, we have performed similar calculations for twenty seven more randomly generated examples with six tasks and five processes, using all three methods. We wish to ascertain the validity of the three methods being implemented and this also allows for a comparative study. We define the *Probability of Success* as the percentage of clusters that are currently identified by a method for a particular example. In figure 8 this value is then plotted against the frequency of occurences of this success value. In the case of the Jaccard's method, we find upon appropriately choosing the threshold value we are able to identify the correct clusters twenty six out of twenty seven times. The MCA method also yields a high success rate majority of the times. In the case of the MST method, one hundred percent success is achieved only once but its distribution curve seems to indicate that it achieves

Table 5. Comparison of the service compositions obtained by the different methods used with the actual clusters found in the data for examples 1 and 2. The table shows the number of actual clusters and the resulting probability of success (P) obtained using each of the methods.

Granularity	# Actual clusters	Jaccard's distance	MCA	MST
		Example 1		
2	87	87 ($P = 1$)	58 ($P = 0.67$)	14 ($P = 0.16$)
3	49	49($P = 1$)	14($P = 0.29$)	6 ($P = 0.12$)
		Example 2		
2	10	10($P = 1$)	10($P = 1$)	6 ($P = 0.6$)
3	10	9 ($P = 0.9$)	7 ($P = 0.7$)	2($P = 0.2$)

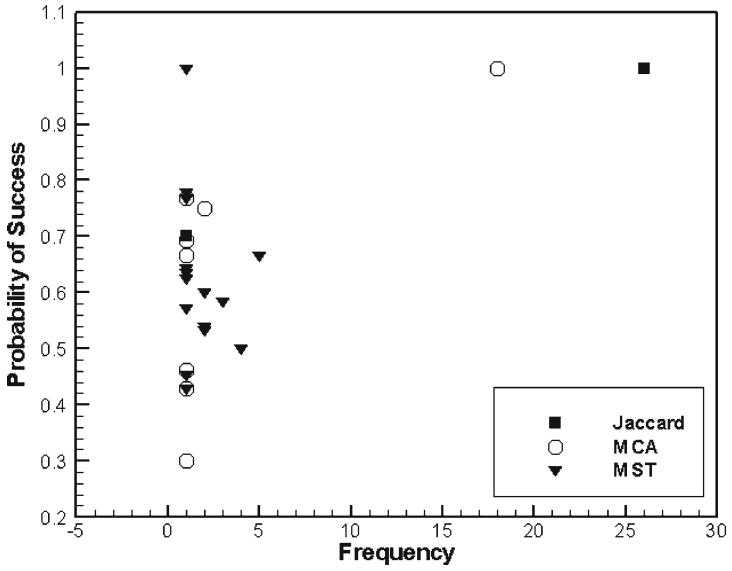

Figure 8. This figure shows the probability of success versus the frequency for 27 randomly chosen examples using each of the three methods for the case of granularity 2 clusters.

over fifty to sixty percent success on an average. Before declaring Jaccard's method the winner, however, one must remember that the results listed by the MST method are only based on the minimum path. If one were to choose the subsequent minimum paths, then one can identify more clusters. It is important to remember that the ultimate job at hand is not merely to identify all the existing clusters but those that are most repetitive which all the methods are able to do successfully.

We also observe that our methods provide us with several other clusters which do not exist in these groupings in the real scenario but may be frequently found to appear sufficiently 'close' to each other. Such clusters can also be significant in constructing services. We are working on more sophisticated real-time examples and results to which will be published in our later work. These clusters can give clues about possibly redesigning the work flow so to take into account the groupings suggested by our initial approach.

5. Leverage

Our examples and conclusions from the previous sections are based on fixed work flow patterns. In real applications however, work flow can evolve and change. Therefore it is important to consider the impact of such a structural change on the service compositions. To this end, we introduce the concept of *leverage* which quantifies the change in services based on any changes in tasks in a process. We define the term *leverage* as the measure of change in service composition when a task is added or removed from a process. It is denoted by $Q_{t,p}$. We relate $Q_{t,p}$ to the measure of deviation between the original process and changed process which can be defined as $Q_{t,p} = \text{norm}(d)$, where d is the deviation from the original distance (based on Jaccard's distance norm) to the distance after altering a task in a process. High $Q_{t,p}$ values tell us that a change of a particular task will have high impact on the service composition. This value gives an idea about

Table 6. Leverage matrix

1.0026	0.57421	0.63487	1.0026	0.55503	1.0026	0.57421	0.50277	0.73371	0.86603
0.59043	0.4259	0.35785	0.50083	0.34561	0.59043	0.4259	0.31447	0.40791	0.49554
0.65107	0.42753	0.39826	0.69582	0.43044	0.65107	0.42753	0.34681	0.69582	0.58973
1.0252	0.59137	0.68435	0.96825	0.56519	1.0252	0.59137	0.50908	0.80156	1.0252
0.54416	0.36742	0.29861	0.43684	0.36931	0.54416	0.36742	0.29533	0.37565	0.43076
0.64118	0.46786	0.38766	0.60162	0.39791	0.64118	0.46786	0.36931	0.52015	0.59722

the importance of a task in particular process and gives the critical tasks in a work flow. In order to elucidate this concept better, we consider the following Example 3 with different processes and tasks:

$$\begin{aligned} P_1 &= t_1t_2t_5t_6t_7t_8t_{10} \\ P_2 &= t_2t_7t_8 \\ P_3 &= t_3t_5t_9 \\ P_4 &= t_1t_2t_4t_5t_6t_7t_9 \\ P_5 &= t_5t_8 \\ P_6 &= t_2t_3t_7t_8. \end{aligned}$$

If we add or remove a task from any of the processes we get a new distance vector, corresponding to a row in the distance matrix. The leverage is the norm of vector which we get from subtracting the new distance vector with the original vector. The leverage matrix for the process–task coefficient matrix, in the above example, is given in table 6 with clusters $\{t_2, t_7, t_8\}, \{t_1, t_6\}$.

In table 6, the [4, 1] element is the highest i.e., 1.0252. So it has large impact on service composition. The [4, 1] element in the process–task coefficient matrix is 1. If it is changed to 0 i.e., in P_4 we ignore t_1 then the changed clusters are $\{t_2, t_7, t_8\}, \{t_1, t_{10}\}$. Therefore, the composition is drastically changed (see figure 9).

Consider the minimum element in table 6, namely [5, 3] with value 0.29861. Since P_5 does not contain t_3, so even if we were to include t_3 in P_5 the clusters remain $\{t_2, t_7, t_8\}, \{t_1, t_6\}$. Therefore, we observe that there is no change in the cluster if we change the non-critical/low-leverage but the cluster does change for critical/high-leverage tasks.

This aspect of our work is closely related to the concept of dynamic clustering (Arnaldo *et al* 1998; Charikar *et al* 1997; Can 1993; Chaudhri 1994) which has been successfully implemented in problems of machine learning, pattern recognition and related problems. The dynamic clustering method has the great advantage of seeking clusters while allowing for variations in the network without have to entirely rerun the clustering program; the only restriction is that the variations are incremental. In SOA, in general, one cannot necessarily restrict variations of the network to incremental changes alone and overall structural changes including internal variations and scaling would significantly change the nature of the services formed. In this section of this paper, we simply introduce the concept of 'leverage' and have shown a simple example of this idea. We have elaborated and explored further this idea in a follow up paper to this work (Kadam *et al* 2010b). In future, we would like to understand and adapt the dynamic clustering algorithms to our work to explore the advantage of seeking services through these methods and compare with our present results.

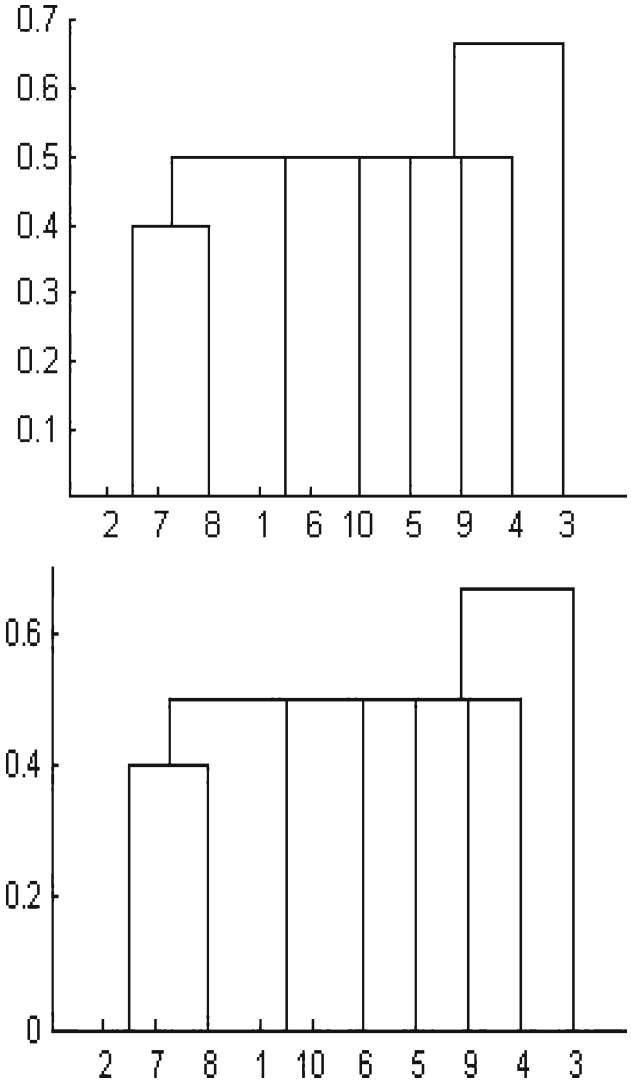

Figure 9. Dendrograms showing clusters pertaining to Example 3. The first graph shows the clusters for the original example while the second graph shows the resulting clusters upon removing the task t_1 from P_4.

6. Comments on the web service architecture

The ultimate aim of this work, as we have stated in the introduction is the architecture based on the service compositions that we find using the methods above. This task of reconstructing the work flow design by using different possible combinations of the services, is a very complex task and based on several factors which define the quality of service (QoS) of a web service. The granularity of the compositions is a very important issue from the application point of view. It is not immediately obvious if a coarse or a fine granularity of composition is preferable in any particular application. Mathematically speaking, this is an optimization problem where the necessity of the service composition and the total architecture is dependent upon several variables such as

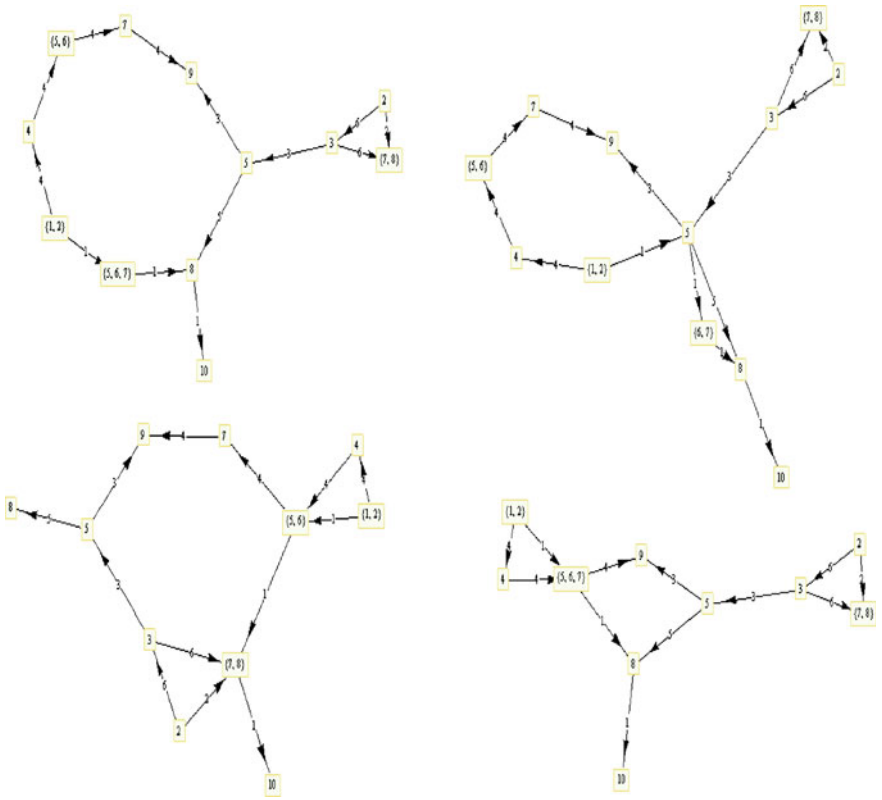

Figure 10. The figure shows several possible ways of architecting a software using the service compositions discovered for example 3. The ultimate goal is to find the configuration from among such choices which is *optimal* for the business. The tasks or services are indicated as nodes and the process number is labelled on the edge.

the cost, time, reliability, availability of the services, granularity, etc. (Yu *et al* 2007; Yanchuk *et al* 2008). The complexity of the problem can be visually elucidated by means of the figure 10. The figure is based on the results of Example 3 and shows some possible architecture scenarios at different granularities. It is to be noted that though the example discussed here is seemingly simple the realistic scenarios are extremely complex with an extremely large number of tasks and process. In such a case there are several difficulties that one faces among which includes finding service compositions; a brute force search is just not efficient in such a case. Additionally for such large cases, the number of possible combinations of services and independent tasks can be very large as well.

7. Conclusions

This paper discusses quantitative techniques based on network science for finding service compositions in a work flow using cluster analysis and graph theoretic methods such as Jaccard distance norm, Multiple Correspondence Analysis and Minimum Spanning Tree. We provide

two examples, first the randomly generated case and second the realistic software case study, both of which yield similar results at different granularities. We also studied twenty seven more random examples, the results of which are summarized in the figure 8. On an average we find that the Jaccard's method yields an average of about ninety nine percent of the actual clusters correctly. The MCA and MST methods yield about eighty seven percent and sixty percent respectively of the clusters correctly, based on our random examples. The success rate can be further improved by appropriate choice of threshold values for the Jaccards and MCA methods and by choosing different hierarchies of minimum paths for MST. Our first and foremost contribution in this paper lies in identifying and modelling the problem of SOA as a complex network and in the use of quantitative techniques from network science to address the issue of service compositions.

The concept of leverage is introduced as a measure of deviation in service compositions with respect to any changes in the work flow. Our quantitative methods of macro cluster identification are more likely to yield correct results than the intuitive methods followed currently. Further, computational time is reduced considerably. This is a particularly significant contribution to the study from the point of view of the application we are considering. While granularity based service composition is an important fundamental step in SOA, the concept of leverage allows us the freedom of making modifications to a work flow and obtaining a measure of the changes in the resulting service composition. This is helpful in two respects: (i) it allows us to compare service composition possibilities in two completely different business set-ups and (ii) also allows a given business to measure the change in services that it must undergo as a result of its own evolution in time.

The methods discussed here are purely metric based and made under the assumption that order of tasks is prescribed i.e., the computations are performed for a given work flow order. Also, the structure of the work flow has been taken to be linear in this paper. This means that the sequential multiplicity of a task in any process can be at most equal to 1. Furthermore, as is pointed out in (Yu *et al* 2007), conditional flow and loop structures within the work flow are equivalent to the linear flow; they can be considered as additional processes (see figure 1, for instance).

As discussed in section 6, the problem at hand can be a daunting combinatorial task. Therefore, we employ metric based methods which are computationally fast and an efficient way of finding clusters but may be prone to some errors. One of the problems in cluster analysis is the order of complexity. For large number of tasks there might be computational problems with the metric-based methods viz. Jaccard's. Moreover, if there are more than one pair of tasks which have same distance it takes any pair randomly in the first iteration and later adds the remaining pair to it. The drawbacks of MCA are that it is not a deterministic approach and secondly, there is loss of some information since we take only the first two eigen vectors based on a threshold parameter. Also, there is a chance of ambiguity for higher order processes since this is a graphical method. For MST we do not have to consider any threshold value or even discard any information. In the examples, the three methods give somewhat different clusters (see table 5). This is due to above mentioned reasons wherein some subjectivity or information loss is inherent in the solution procedures of some methods. It also appears to have to do with the dimension of the problem.

As described in the introduction, the architecture of the services is the ultimate task. This rather complex problem comprises of (i) defining the quality of services, (ii) optimizing the service utilization and finally (iii) quantitatively approaching service orchestration and service choreography. This paper manages to approach these early stages of this problem from a very fundamental point of view which is lacking in the previous literature. Though several interesting attempts have been made at finding optimal service compositions based on appropriately defined objective functions, our approach from a discrete point of view is perhaps the most

realistic model. We hope to discuss the optimization issue in combination with the currently analysed macro cluster analysis in our next paper. So other issues of immediate interest to us include (i) a detailed study of the concept of leverage and its links to dynamic clustering; (ii) how the probability of success changes with the threshold parameter, granularity and scaling of the problem.

The authors are grateful to Krishna Vedula for the helpful discussions and contributions. The author AV also wishes to thank Preetam Ghosh and Peter Mucha for their feedback. We are also grateful to the reviewers for their valuable suggestions to improve the paper.

References

Albert R and Barabasi AL 2002 Statistical mechanics of complex networks, *Rev. Modern Phys.* 74: 47

Arnaldo J, Abrantes J and Marques S 1998 A method for dynamic clustering of data. *Proceedings of the British Machine Vision Conference*, Lisbon, Portugal

Can F 1993 Incremental clustering for dynamic information processing. *ACM Transactions on Information Processing Systems* 11: 143–164

Chalak L, Chehade A, Elbitar A, Cosson P, Zanetto A and Dirlewanger E 2006 Morphological and molecular characterization of peach accessions cultivated in Lebanon. *Lebanese Science Journal* 7(2)

Charikar M, Chekuri C, Feder T and Motwani R 1997 Incremental clustering and dynamic information retrieval. *Proceedings of the twenty-ninth annual ACM symposium on theory of computing*. ACM New York, USA

Chaudhri B B 1994 Dynamic clustering for time incremental data. *Pattern Recognition Letters* 13: 27–34

Erl T 2007 *SOA: Principles of service design*. Boston, USA: Prentice Hall

GRaph INterface by Prof Vitaly Pechenkin. [Online]. Available: http://www.geocities.com/pechv_ru/, January 2010

Greenacre M J 1984 *Theory and applications of correspondence analysis*. London: Academic Press, 364 p

Greenacre M J and Balasius J 1994 *Correspondence analysis in the social sciences*. London: Academic Press, 370 p

Grygorash O, Zhou Y and Jorgensen Z 2006 Minimum spanning tree based clustering algorithms. *Proceedings of the 18th IEEE international conference on tools with artificial intelligence* (ICTAI06).

Hartigan J A 1975 *Clustering algorithms*. New York: Wiley

Kadam S, Kolluru D, Vaidya A and Viswanadhuni V 2010a Optimal clusters and architectures in complex networks. *Proceedings of the International Conference on Advances in Recent Technologies in Communication & Computing*, Kottayam, India, paper 51

Kadam S, Kolluru D, Viswandhuni A and Vaidya A 2010b Service worthy clusters in an evolving network, (submitted for publication)

Lang J C, Widjaja T, Buxman P, Domschke W and Hess T 2008 Optimizing the supplier selection and service portfolio of a SOA service integrator. *Proceedings of the 41st Hawaii International Conference on System Sciences*

Narsingh D 2005 *Graph theory with application to engineering and computer science*. Eastern Economic Edition, Prentice Hall India

Newman M E J 2003 The structure and function of complex networks, *SIAM Review* 45(2): 167

Pastor-Satorras R and Vespignani A 2001 Epidemic spreading in scale-free networks, *Phys. Rev. Lett.* 86: 3200

Porter M A, Friend A J, Mucha P J, Newman M E J 2006 Community structure in the US House of Representatives. Chaos 16: 041106

Porter M A, Mucha P J, Newman M E J and Friend A J 2007 Community structure in the United States House of Representatives, *Physica A* 386(1): 414–438

Timm N H 2002 Applied Multivariate Analysis. Springer Verlag

Wu B Y, Chao K 2004 *Spanning trees and optimization problems*. Boca Raton, FL: CRC Press

Yanchuk A, Ivanyukovich A and Marchese M 2008 *Towards a mathematical foundation for service-oriented applications design*. LNCS 3826: 545551. Berlin, Heidelberg: Springer-Verlag

Yu T, Zhang Y and Lin K May 2007 Efficient Algorithms for Web services Selection with End-to-End QoS Constraints. University of California, Irvine *ACM Transactions on the Web* 1(1), article 6

Zahn C T 1971 Graph-Theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers* C-20(1): 68–86

Zarei M and Samani K A 2009 Eigenvectors of network complement reveal community structure more accurately, *Physica A* 388: 1721–1730