A new Framework for Service ranking including CPM/PERT methods

Festim Halili, Mirushe AMETI, Merita KASA HALILI, Ilia NINKA

Abstract

In this paper we have proposed a new comprehensive framework for web service discovery and ranking. There are activities to be taken regarding the process of our proposed framework for ranking quality web services. Using CPM method the emphasis will lay on the trade-off between the cost of the framework to be implemented and its overall completion time. In other hand, by applying PERT techniques we will specify the completion of the framework activities in the shortest possible time. The deployment of similar or same web services into dedicated repositories (or registries) for the end-users to discover and select them has significantly increased lately, which has led to a new problem of choosing a suitable web service for the requester based on their needs and expectations. In view of this challenge, a QoS-based method for web service ranking is proposed in this paper. To achieve desired selection of a quality service, it is necessary to produce a framework that enables evaluation of a web service quality based on factors in terms of attributes, in addition to their level of importance based on the opinion of skilled engineers of Service Oriented Computing. Based on the proposed factors, the service providers can improve qualities of their web services, whilst the requester will have a proper list of ranking.

Keywords: Web services, QoS, matchmaking, service discovery, ranking, selection, quality factors, SOC, CPM/PERT

1. Introduction

The traditional approaches to software development where interactions are based on the exchange of the products with specific clientship are known as object-oriented programming, but they are not designed to face the challenges of open environments. SOC (Service Oriented Computing) [1] is a new paradigm and provides a way to create a new architecture that is determinant toward autonomy, versatility and heterogeneity. SOC uses services to sustain the development of low-cost, loosely-coupled, evolvable and massively distributed applications. Services are autonomous and self-describing entity and they can be published and discovered by the service requestor. The deployment of web services into traditional registries such as UDDI [4, 5] is massively executed by service providers. According to Zheng [6] by 2010 there have been existing around 28,500 public web services on the internet, and it is clearly that the number has increased by now.

Web services are versatile, meaning that they can be used in different styles like: RPC, REST and SOA [7, 8, 27] and the benefits of using them by business or non-business companies is explained in [9,10]. The architecture of web services relies on different standards or protocols, the one for service description based on functional properties is WSDL [11], but the lack of this standard is that it doesn’t evaluate or describe non-functional properties of web services, by means of QoS attributes, as it can be seen in the figure 1, where the Abstract part describes the sent and received messages and the operation which associates a message exchange pattern with one or more messages, whilst the Concrete part specifies the transport and wire format details for one or more interfaces and also the port (an endpoint) associates a network address with a binding.
To define the quality of web services, it is a difficult target; because the quality may depend from task-related factors which would affect the end users requirements.

A true concern to the service selection is the preliminary way to rank the matching services based on some criteria: a) semantic-based measure [13] or non-functional properties (QoS or eminence) [14]. There are proposed technical and methodical aspects for understanding the engineering foundations of SOC that concern the way applications can be developed to provide solutions to their business, without the regard of the programming languages the services are coded. SCA (Service Component Architecture) is proposed by the Open Service Oriented Architecture collaboration which addresses the aim of creating the model for service components [28]. However, it points out a low-level design [2] in terms of assembly model and binding mechanism. Another modeling language proposed by SENSORIA is SRML (The SENSORIA reference modeling language) [3, 12] which conveys both the static and dynamic aspect, where the static aspect includes the design-time description of complex services in terms of composition (orchestration) of simpler services which are indicated over state transition systems and temporal logic, whilst the dynamic aspects are formalized over a specified mathematical model. In this paper we will aim to extend the abstract model of service discovery and binding proposed in SENSORIA by including specific QoS factors in the part of Configuration Management of the proposed architecture, which will assist in the ranking process of web services. According to [2] the ontology unit in their proposed framework is an area that is still lacking standards, even though there has been perceptibly progress in the development of Semantic Web Techniques.

The current proposed selection systems concerning the requester (user) design have few issues left aside. Firstly, current systems are presuming that users have the abilities of formulating queries that can reflect their QoS requirements, because users might have no knowledge about what the realistic QoS values indicate. It is very important if the selection systems can provide to users the ability to choose the right QoS values for their requirements.

So, our aim in this paper will be to provide an advanced system with QoS factors as ranking criteria, where users could go through all available services in the given registries (UDDI) to gain particular ideas on their QoS value ranges.
2. Related Work

In order to discover Web services, it is necessary to obtain a collection of available Web services. The QoS has been previously exploited to support the service selection. In the beginning stage of Web services research, UDDI (Universal Discovery Description and Integration) was proposed as a repository or registry for publishing and invoking services, but it has not prevailed in the area of publicly available Web services [15] and by time passing other systems were proposed.

A prerequisite to selection is the process of ranking the matching services based on some criteria: a) semantic-based measure or b) non-functional properties (QoS or eminence), and some ideas in different papers are proposed based on these methods.

A. Semantic-based measure

In [2] it is proposed a unification process of discovery, ranking and selection based on the formal operational semantics. In addition, for modeling this process, there are used unifying wires for a given business configuration system. The semantics is based on a graph-based representation and configuration of the GC (global computers) specified by business activities. The three steps: discovery, ranking and selection are based in compliance with required business and interaction protocols and a tentative of optimization of QoS constraints, but there are not specified exact QoS factors to determine the ranking of available web services.

In [16], it is proposed to add context to knowledge of service provider and requester to service description and incorporate a planning method with intention to achieve the understanding for the inner ontology concepts between parties in service binding.

Segev and Toch in [17] indicate a two-step context-based semantic approach to the problem of binding and ranking Web services for possible composition of services (orchestration or choreography [9]), and it is also given an empirical analysis and comparison of the different methods for classification (WSDL context, WSDL TF/IDF and Description context).

In addition, the semantic-based measure advantage is that many of the tasks involved in using Web services can be (semi) automated, including discovery, selection, composition, mediation, monitoring and ranking. However, many systems have been developed without considering the ability of the Web for integrating services and sharing resources, and thus the industrial rise of Semantic Web Services technologies has been slower than predicted.

B. Non-functional properties

The process of selecting the suitable Web service to a client requirement is an important task as many Web services are able to meet functional requests of user, however, non-functional attributes is an onerous matter also. Various non-functional properties based on QoS models have been proposed in the literature in which QoS parameters are evaluated by a service broker or matchmaker. One such model is evaluated in [18], where the QoS consultant acts as a broker or agent between client and service provider. Once, a search is performed for query, the mediator offers to the requester a list of candidate services that are matched with the given request.

In [5] it is proposed the real-time assurance of service responsiveness incorporated in UDDI registries, but it is not evaluated the process of ranking services according to specific requirements of the service searcher or client, that would ease the process of selection, which is the intention of this paper.

In [19, 20] are proposed QoS based architectures for ranking and selection of web services, by describing the capabilities of a Web service to fulfill the requirements of consumers in specific domain.

Jaeger and Ladner in [21] proposed improving method for QoS of Web Service compositions based on redundant services. They discussed how already identified candidates which a selection process originally has divided out, can lead to improvement of composition with particular QoS categories. They have proposed four QoS categories: execution time, cost, reputation and availability, but they didn’t evaluate other categories, because service requesters often cross the lines of marginalized categories.
An interesting approach to non-functional properties is proposed by Ahmadi and Binder [22], by providing a flexible matchmaker for service discovery, selection and ranking, including both, functional and non-functional properties into consideration. In the proposed framework, the matchmaker provides an expressive language for the clients to define service requests, by indicating the involved registries or service repositories, non-functional parameters and an utility function for ranking Web services. Here, the UDDI are separated from the third party repositories and the matchmaker plays a crucial role enabling the service requestors to search inside them, while supporting different service description languages and emerging languages. However, perception of the structural properties of frameworks often assists in gaining better insights and develops better algorithms. Based on this, in our paper, relied on large-scale services discovery, we propose an uninterrupted method to rank web services.

3. Proposed Service Ranking and Selection Framework

A. Problem Complexity

The traditional service discovery model is known for having difficulties in the precise classification and ranking of the list of similar services published by different providers, because it is limited to only three roles: service consumer, service provider and UDDI registry as shown in the figure 2 below:

![Figure 2. Traditional discovery of services](image)

As we can see from the above picture the procedure of discovery and selection of services is simple and easy to understand, but when a list of services meeting consumer’s functional requirements have been discovered, it is difficult for the service consumer to chose which one should be invoked among all these services with similar descriptions and capabilities.

A. Proposed Framework

Our proposed framework deals with a set of non-functional properties indicated by QoS and it flanks performance attributes. The Service Consumer is always concerned about the performance of the service to be invoked; therefore we incorporate QoS in our framework to rate functionally similar web services. Our idea is to facilitate service providers to publish information with QoS, in order to do that; we need to model service descriptions. Furthermore, it is necessary to provide a method for consumers to submit service requirements according to their needs. A powerful web service modeling language has been given in [3] called SRML (Sensoria Reference Modeling Language). Nevertheless, it does not support QoS registry and attributes, therefore we propose a service reference modeling language including QoS named SRML-Q. We extend the configuration management of the overall ‘engineering’ architecture and processes proposed in [2,3] by including several components to support service
classification, ranking, quality update and binding or selection. The main framework for web service selection and ranking with QoS attributes is shown below in the figure 3.

![Proposed QoS approach for ranking and selection framework](image)

Figure 3. Proposed QoS approach for ranking and selection framework

We have defined a web service in the WS-Registry with six elements, one added to the definition in [23]:

$$WS = \{WS-Name, UI-S, WS-Desc, Qf, FSet, In\}$$

According to the above tuple we denote:

- **WS-Name** as the name of the specific web service,
- **UI-S** as the unique identifier of the web service, with intention to discern them from each other.
- **WS-Desc** service role description
- **Qf** is the published QoS factor description that is specified as $Q_f = Q_S \cup Q_D$
- **FSet** is the web function set pointed as $FSet = \{fset_1, fset_2, \ldots, fset_n\}$, where each $fset_i (1 \leq i \leq n)$ can be triggered for a certain operation task.
- **In** is the invocation factor, denoting the number of times a web service is requested and binded to a specific consumer.

Similarly we define the service request, just removing the UI-S, which is not necessary in this case. This system assists the Service Consumer to search web services based on the input/output operations which incorporate with the QoS requirements. The similarity batch assists in classifying similar web services and prepare them to send to the ranker, who proceeds in ranking web services based on the QoS preferences set by the client and highest ranked service will be provided. There are occasions when many similar web services are readily available to meet consumer request. In case the list of the provided services by the QoS database is long, then the service selection method is activated that take inputs as matches services as per client preferences, afterwards depending on the approach of the constraint’s and client’s requests, it runs the proper web service selection algorithm and provides the results to the client.
The client has the right to rate the consumed service by providing information regarding the working experience of the service, then the UpdateOperator refreshes quality criteria value in the QoS database regarding the collected feedback information in quality rating database.

4. Categorizing Quality Factors

To measure quality of web services, we would need to define and categorize the quality factors. In [24], the quality factors are formulated based on three objectives: Usability, Conceptual Reliability and Representation Reliability. In the appendix of the same paper, there are given tables with identified service quality factors and quality sub-factors and the number is enormous. For our framework, we propose the quality factors that are often indicated as most important to the service consumers:

**Execution Time** - The execution time defines the time needed to execute the service. In this paper, we presume that the values of individual services result in the overall execution time of the composition. For our calculation we consider the worst/maximal execution time as relevant.

**Cost** - The cost represents the amount of resources needed to use a service. For the calculation of the composition we presume that the resources are spent once a service is invoked.

**Reliability** – Ratio of error messages to total messages.

**Reputation** - The concept of reputation represents a ranking given by users of the service, like the auction platform ebay allows clients to rank the behavior of other clients [11]. Since the reputation of an individual service can be considered as the average ranking of individual users, we regard the average of the individual values as the aggregated reputation of a composition.

**Availability** - The availability denotes the probability that the invocation of the service performs successfully and delivers a result within the promised QoS of other categories.

5. CPM/PERT for the Framework of Ranking Services

A. Problem Complexity

According to Manikandan [29] a project is a series of activities directed to the accomplishment of a desired objective. CPM (Critical Path Method) is a method where activities are shown as a network of precedence relationships using activity-on-node network construction. It has deterministic activity times. In other hand, in PERT activities are shown as a network of precedence relationships using activity-on-arrow network construction. It has probabilistic activity times. In table 1 we have listed the activities and descriptions necessary to complete the project of creating a model serving as a framework for ranking services. It indicates the principles of applying CPM/PERT methods to enable the trade-off between the cost of the project and its overall completion time and also for completing this project in the shortest possible time. There are several stages which must be used to find CPM. The first stage during finding CPM/PERT is to define the activities that must be performed to achieve the specified goal and to display duration of activities (days, weeks, months,..) and immediate predecessors of activities (which activity depends from another). In the table below are shown activities and their duration.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
<th>Immediate predecessors</th>
<th>Duration (weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Planning for modeling the service ranker</td>
<td>/</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>Configuration management</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>Coding web services</td>
<td>A,B</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>Creation of QoS Database</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>Binding and Classifying services</td>
<td>A,D</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>Installing and testing other hardware device</td>
<td>A,B</td>
<td>3</td>
</tr>
</tbody>
</table>
If we sum the provided number of weeks in the above table we can easily notice that the total time required to complete activities is 18 weeks. However, in some cases we can see that two or more activities can be conducted simultaneously.

**Earliest start & earliest finish time** - We are interested in the longest path through the network, i.e., the critical path. Starting at the network’s origin (node 1) and using a starting time of 0, we compute an earliest start (ES) and earliest finish (EF) time for each activity in the network [25]. The expression EF = ES + t can be used to find the earliest finish time for a given activity. For example, for activity A, ES = 0 and t = 1; thus the earliest finish time for activity A is EF = 0 + 1 = 1.

**B. Constructing the network of activity dependencies**

To construct a network of CPM for a specific project we must first have the activities and their duration and also dependence of activities. Since we already have the table we can begin with the construction of the network of CPM, respectively construction of network with EF (earliest start) and ES (earliest finish) time.

In CPM activities are shown as a network of precedence relationships using activity on node network construction, where we can use two methodologies: Single estimate of activity time and Deterministic activity times. From the following project network that we have constructed, it is noticeable that we have used nodes (representing an event) and arrows (representing activity).

![Project Network with ES and EF time](image)

**Figure 4. Project Network with ES and EF time**

**Earliest start time rule:**

The earliest start time for an activity leaving a particular node is equal to the largest of the earliest finish times for all activities entering the node.

**Latest start & latest finish time** - To find the critical path we need a backward pass calculation. Starting at the completion point (node 10) and using a latest finish time (LF) of 12 for activity J, we trace back through the network computing a latest start (LS) and latest finish time for each activity. The expression LS = LF – t can be used to calculate latest start time for each activity. For example, for activity J, LF = 12 and t = 2, thus the latest start time for activity I is LS = 12 – 2 = 10 [26].

Now, we can construct a complete CPM network with ES, EF, LS, LF as seen in the figure below:

---

1 SRML-Q is a SENSORIA Reference Modeling Language including QoS (Quality of Service).
Slack (Free Time) - Once the event times have been specified then the activity slack can be determined. Activity slack is defined as the difference between the late start (LS) and early start (ES) times for an activity. Numerically, \((LS - ES) = \text{activity slack}\). Also, the difference between the late finish (LF) and the early finish (EF) times also indicates the amount of slack for an activity. Numerically, \((LF - EF) = \text{activity slack}\).

During the planning phase of a project, planners often develop budgets for each task and sub-task of the project. When these budgets are aggregated, we arrive at the budget for the project. Besides managing time, project managers must monitor budget expenditures throughout the project. As the project progresses through time, the graphical depiction of the network must change to reflect the progress being made on the project. Project managers usually hold regular meetings to determine the status of a project’s implementation with regular updating of the PERT network required (or updating of whatever project management tool is being used).

Afterwards, when we construct the CPM network, including ES, EF, LS, LF and slack, we can also build a table where we can represent the measures of EF, ES, LS, LF and slack for each activity, as we have shown in the table below:

**Table 2. Detailed Table for each activity including ES, LS, EF, LF, slack**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

The importance of Float or Slack and Critical Path is multidimensional. Slack enables the amount of allowance of each activity, whilst the Critical Path is a sequence of activities from start of the network to the finish with zero slack. In addition, critical activities are activities on the critical path. The minimum time to complete the project is identified by the Critical path. By implementing this method, we will be able to not waste the resources on non-critical activities, because it will not shorten the project time.
As we see in figure 5 we have few paths, but why we have chosen the path outlined in red!? If we analyze all possible paths, we will see that this path is the longest path (based on activity duration). In this network possible paths are:
A-B-C-Dummy-I-J = 7 weeks
A-B-F-H-I-J = 12 weeks
A-B-Dummy-I-J = 7 weeks
A-D-E-G-H-I-J = 11 weeks
A-D-E-Dummy-I-J = 7 weeks
For this reason we chose for critical path (longest path), the path A-B-F-H-I-J=12 weeks.

C. PERT for dealing with uncertainty

So far, times can be estimated with relative certainty, confidence. For many situations this is not possible, e.g Research, development, new frameworks and projects etc. For PERT we could use 3 time estimates:
- \( m \) = most likely time estimate, mode.
- \( a \) = optimistic time estimate,
- \( b \) = pessimistic time estimate, an

Expected Value (TE) or \( \mu = (a + 4m + b) / 6 \)

Variance \( \sigma^2 = (b - a / 6)^2 \)

Std Deviation \( \delta = \sqrt{\sigma^2} \)

When we know the formulas for these times, we could construct the table below which should help us to construct later the PERT network for finding critical path.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Immediate predecessor</th>
<th>Opt. time (a)</th>
<th>Most likely time (m)</th>
<th>Pess. time (b)</th>
<th>Expected Value ( \mu )</th>
<th>Variance ( \sigma^2 )</th>
<th>S. Dev ( \delta )</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>/</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.11</td>
<td>0.33</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>1.78</td>
<td>1.33</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>A,B</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.11</td>
<td>0.33</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.11</td>
<td>0.33</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>A,D</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>1.00</td>
<td>1.00</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>A,B</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>1.00</td>
<td>1.00</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>D,E</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>B,F,G</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>1.00</td>
<td>1.00</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>C,E,F,H</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.11</td>
<td>0.33</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>H,I</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>1.78</td>
<td>1.33</td>
<td>J</td>
</tr>
</tbody>
</table>

While we construct PERT method we apply the same technique as CPM, with a slight difference that in this case, for the purpose of finding critical path we don't have to rely on activity duration, rather than in this case a significant role plays the factor of the Expected Value \( \mu \) (Mi from greek), because when we find a critical path we will find it based on \( \mu \). So, the network of PERT will have the following appearance:
After we construct the PERT network, it is easily possible to fill another table with information where we could find the LS (latest start), ES (earliest start), and a slack for each activity similar as we did with CPM network.

<table>
<thead>
<tr>
<th>Activity</th>
<th>LS (Latest Start)</th>
<th>ES (Earliest Start)</th>
<th>Slack</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>H</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>I</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>J</td>
<td>19</td>
<td>19</td>
<td>0</td>
<td>Yes</td>
</tr>
</tbody>
</table>

As we can notice in the figure 6, there are several paths, but why we have chosen the path outlined in red?! If we analyze all possible paths, we will see that this path is the longest path (based on activity duration). In this network possible paths are:
A-B-C-Dummy-I-J = 14
A-B-F-H-I-J = 20
A-B-Dummy-I-J = 13
A-D-E-G-H-I-J = 22
A-D-E-Dummy-I-J = 14
For this reason we chose for critical path (longest path)

D. PERT for dealing with uncertainty

Firstly, we will graphically present the paths in the CPM network and with their associated times to complete the project:
From the above data represented in graph, we can notice that the violet dot represents the highest value, indicating that the path A-B-F-H-I-J in the constructed network of CPM specifies the necessary time to complete the activities in the proposed framework (as seen in figure 3) for the web service selection and ranking process, which is 12 weeks.

In regard to the figure 6 indicating the PERT network and the critical path, we will graphically present data for the constructed network of activities:

From the above data represented in graph, we can notice that the blue symbol represents the highest value, indicating that the path A-D-E-G-H-I-J in the constructed network of PERT specifies the necessary time to complete the activities in the proposed framework (as seen in figure 3) for the web service selection and ranking process, which is 22 weeks.

6. Conclusion and Future Work

The composition of web services is faced with the large number of growing Web services into registries, there have been given a lot of efforts to deal with this problem, but still needs more research to be expressed. The process of ranking discovered Web services on the discovery service side, makes the composer able of regulating time and quality of the generated QoS based web services. In our proposed framework, we extend the traditional way of searching and selecting web services by adding QoS database incorporated with quality factors, and also proposed a SRML-Q service reference.
modeling language to meet the constraints between the service consumer and service provider, which would help a lot in ranking web services.

CPM/PERT enable us to balance the trade-off between the cost of the framework construction for web service discovery and ranking and the overall completion time.

For the future work, we are going to propose several algorithms for service selection with QoS factors, by describing the differences between them, and also provide an algorithm to calculate and rank each service’s QoS value. Furthermore, we will provide a simulation experiment of a SET of services QoS information.

7. References