Trustworthiness Monitoring and Prediction of Composite Services

Hisain Elshaafi, Jimmy McGibney and Dmitri Botvich
Telecommunications Software & Systems Group
Waterford Institute of Technology, Waterford, Ireland
Email: helshaafi, jmcgibney, dbotvich@tssg.org

Abstract—This paper presents an approach to monitoring and predicting the trustworthiness of services that are assembled from component services. In service compositions the number of component services that need to be aggregated may be large and dynamically changing. Additionally, the component services may vary in their importance to the value of the composite service and in their trustworthiness and resource capacity. Service compositions require the capability to dynamically adapt to changes that may occur at runtime. Those changes can occur in supply and demand, in the environment or in the component services’ properties and behaviour. Service composers need to be able to respond swiftly to changed trustworthiness requirements and capabilities of service compositions, where those changes may not be easily predictable. With the availability of alternatives providing the same functionality as those already integrated in a composition, service composers can take advantage of this by replacing degrading or unsatisfactory components.

I. INTRODUCTION

Service Oriented Architecture (SOA) and Services Computing are increasingly popular, with increased attention from industry. A key concept is that services can be dynamically or statically composed to create new services. Services are described, published, discovered, and assembled, providing distributed business processes. The paradigm enables flexible use of resources for optimising operations within and across organisations. Service composition assembles multiple basic services into a single value-added combined service. The resulting service may be used directly by a service consumer or recursively incorporated in further compositions. Dynamic service composition is performed automatically at runtime as compared to static development-time composition.

For a composite service (CS) to be trustworthy for utilisation by its target consumers, and for services to be able to trust and rely on other possibly unknown services, service composition techniques must be able to identify which component services are trustworthy. The composition techniques also must be able to maintain the most trustworthy and cost efficient composite service. Maintaining trustworthiness helps consumer confidence and provides a safe environment for businesses to dynamically interact and carry out transactions. Therefore, addressing trust is essential for the success and adoption of the services paradigm.

We define trust as a relationship between two or more entities that indicates the contextual expectations from an entity towards another in relation to reliance in accomplishing a certain action at a certain quality. Trustworthiness of an entity is the level of trust that the trusting entity or its agent has in that entity.

Trustworthy and secure dynamic service compositions require run-time monitoring and adaptation of services. The adaptations are needed due to changing operational, business threat environments or due to changes in service qualities and behaviour. Among the challenges to addressing those problems is that the changes may not be easily predictable. Since down-time is costly, a composite service must be able to operate even during an attack or increased demand, taking risks and adaptation costs into account. This paper focuses on those problems with the aim of establishing and maintaining the trustworthiness of dynamic composite services.

Component services in a composition may vary in their importance to the composite service as a whole. For example, in a travel service a user may not appreciate all component services to the same extent such as car rental, health insurance, flight booking, etc. Therefore, it is more useful to see the composite service as a unit that is composed of unequal subunits in terms of their contribution to the quality and trustworthiness of the service. The components may differ also in the probability of their execution in the CS based on their resource capacity or on other criteria. This paper proposes the mechanisms for monitoring and predicting the trustworthiness of composite services taking into accounts the variables discussed above.

The paper is structured as follows: Section II discusses the function and architecture of the trustworthiness module. Section III explains the mechanisms for the aggregation and calculation of the trustworthiness of a composite service based on the service’s composition plan. It also includes calculation of the service costs. The prediction of service trustworthiness based on its monitoring is examined in section IV. Section V describes experiments using simulations of the trustworthiness monitoring and dynamic composition and adaptation. Related work is described in section VI and conclusions and future work are discussed in section VII.

II. TRUSTWORTHINESS MODULE

The Aniketos project [1] includes a trustworthiness module (TWM) which is responsible for runtime monitoring of composite service trustworthiness based on a set of mechanisms and metrics to ensure contract compliance. Aniketos is an EU research project that addresses trustworthy and secure
service compositions with run-time monitoring and adaptation of services. Monitoring is the process of checking that service contracts are fulfilled over time, particularly if changes can occur to operational or business environments or to internal service quality and behaviour. Monitoring is also used to detect vulnerabilities and discover attacks on a service, e.g. by making use of intrusion detection systems or dynamic testing tools available in the environment.

A service composer is a service provider that is responsible for constructing service compositions and offering them to consumers. A service composer is notified of important changes in the trustworthiness of the composite service as a result of one of its components. A component service that is below the satisfactory trustworthiness level can be replaced with another component service offering the same functionality but with better trustworthiness. The monetary cost of a composite service as a result of its adaptation is also determined. The consideration of costs ensures that a balance is maintained between both trustworthiness and cost efficiency of the service. The main focus of this work is on the trustworthiness monitoring and prediction for maintaining trustworthy composite services.

Figure 1 shows the architecture of the TWM. The trust events refer to the notifications received by the module from event Processing, QoS monitoring, user ratings and other components. Those events include QoS metrics and alerts that indicate violations or adherence to the service contracts, threats or changes in the environment. In addition to the direct experience through those events, TWM can exchange recommendations with other online modules in relation to service trustworthiness. Incoming events are evaluated by a rules engine to generate service ratings. The rules calculate the rating for the event and add other attributes including the event timestamp and type. Trust ratings are then stored by the module and can be used for calculating the overall trustworthiness level of each service. Context configurations allow customisation of the trust context by adjusting the weighting of types of trust events e.g. security and performance events. Policy configurations allow setting the trustworthiness thresholds and algorithmic constants such as the rating decay rate. The trust engine is responsible for the aggregation of trustworthiness of a composite service from that of its components and providing a prediction of the trustworthiness level of a service. The TWM is implemented in Java as dynamic OSGi service platform [2] sub-modules and uses Drools [3] for implementing the rating rules. This architecture allows the substitution of the sub-modules dynamically as in the case where alternative algorithms are required or configurations for the policy and context need to be changed.

III. TRUSTWORTHINESS OF A SERVICE COMPOSITION

We model the trustworthiness level $T_{cs}$ of a composite service in general as a function $g$ of the trustworthiness of its components:

$$ T_{cs} = g(\{T_1, T_2, ..., T_m\}) $$

However, the calculation of the trustworthiness level depends on the trustworthiness properties and the structure of the business process. The selection of component services statically (during design time) or dynamically is based on the predicted trustworthiness level of the composite service. The trustworthiness properties include a set of properties that are used to determine the overall trustworthiness level. In this paper we consider reputation, reliability, and security properties. Reputation is the information available about a service from user ratings that can be used to determine its trustworthiness. Reliability refers to the percentage of successful execution of a service within a time limit. Security properties may include a number of properties such as confidentiality, non-repudiation, authentication, and encryption. A security property $s_v$ for a service $v$ is a boolean $s_v \in \{0, 1\}$ with 1 representing the fulfilment of the property and 0 for its non-fulfilment. For other trustworthiness properties the value may be scalar as in reputation $p$ and reliability $r$ where $0 \leq p, r \leq 1$.

Selected services are executed in a business process. The process is viewed externally as the composite service. The calculation of the trustworthiness of the CS depends on the way the abstract service is constructed. It also depends on the probability of execution and the importance of the component services in the composition as in the travel service example described in section I. The probability of execution of a component service may be based on the characteristics or interdependence of component services in the process or limited supply of the component service. For example in an emergency composite service a fire or ambulance service may be required in a certain percentage of executions. An example of limited capacity is where a certain car rental service is most trustworthy but has limited supply. In that case more demand requires additional supply from other possibly less trustworthy car rental service providers.

A. Service Composition Constructs

Component services may be invoked in a business process in one or more path constructs such as the following basic and commonly supported constructs (illustrated in Table I):

- **Sequence**: Services are invoked one after another.
- **Parallel with Synchronisation (AND split/AND join)**: Two or more services are invoked in parallel and their outcome
is synchronised. All services must be executed successfully for the next task (service) to be executed.

- **Loop or Iteration**: A service is invoked in a loop until a condition is met. We assume that the number of iterations or its average is known at the time of composition.
- **Exclusive Choice (XOR Split/XOR join)**: A service is invoked instead of others if a condition is met. We assume that the likelihood of each alternative service to be invoked is known at the time of composition.
- **Discriminator (AND split/OR join)**: Two or more services are executed in parallel but no synchronisation of the outcome of their execution.
- **Multi-choice with Multi-merge (OR split/OR join)**: Multiple services may be executed in parallel. Subsequent services can be executed when at least one of the multiple services is completed.
- **Unordered Sequence**: Multiple services are executed sequentially but arbitrarily.

We use \( \theta \) to denote a service construct in a composition. The above constructs and several other possible patterns are supported by modelling languages and products to varying degrees. The constructs are investigated in other works such as Workflow Patterns Initiative [4].

### B. Unequal Weighting of Components

Each component \( v \) in a composition has a weight \( w_v \) based on its importance in the composition \( w_v \in \{w_1, ..., w_l\} \) where \( w_v \geq 0 \) and \( l \) is the number of component services excluding alternatives in exclusive choice constructs (as will be justified shortly). If the weights are normalised for a particular composition as:

\[
\frac{w_v}{\sum_{i=1}^{l} w_i}
\]

then we can represent the weight proportion of \( v \) (\( \beta_v \)) as follows:

\[
\beta_v = \frac{w_v}{\sum_{i=1}^{l} w_i} \cdot l \tag{2}
\]

We consider components in an exclusive choice construct as a single unit in terms of their weight proportion \( \beta_0 \) and its calculation, where \( \theta \) is the service construct. For example, consider the case where a requirement may be satisfied by only one of two services \( \{v_1, v_2\} \) and the trustworthiness of \( v_1 \) is more than that of \( v_2 \) but its capacity is limited to a certain quantity. When \( v_1 \) becomes fully in use, \( v_2 \) is invoked. Therefore, a common weighting value is used. For a composition with \( m \) components and \( x \) exclusive choice constructs:

\[
l = m - (n_x - x)
\]

where \( n_x \) is total number of components in the \( x \) exclusive choice constructs.

The weighting of the components is used in the calculation of the CS reputation, such as in the case of a sequence construct:

\[
p_\theta = \prod_{i=1}^{n} p_i^{\beta_i} \tag{3}
\]

where \( \beta_i \) is the weight proportion of the \( i \)th component service in a sequence construct with \( n \) components, \( 0 \leq \beta_i \leq l \) and \( \sum_{i=1}^{l} \beta_i = l \).

### C. Aggregation of Trustworthiness and Cost

#### 1) Aggregation of Reputation, Reliability and Cost

Table I shows our functions for calculating the reputation \( p_\theta \), reliability \( r_\theta \) and cost \( c_\theta \) per service construct. These properties require the following approaches for their aggregation:

- **Sequence, Parallel and Unordered Sequence**: The reputation and reliability are calculated as a product of that of constituent services with the weighting considered in the case of reputation as in equation (3). In case of reliability a failure of a component means failure of subsequent dependent components. This is unlike some other types of constructs (e.g. discriminator) where subsequent components may be partially independent of the failure of the construct components and can be executed as long as a minimum set of components succeeds. The cost of those constructs is the sum of the cost of their components.

- **Loop**: The reputation, reliability and cost of a loop construct containing \( n \) iterations of a service \( v \) is the same as a sequence construct of \( n \) copies of \( v \) i.e. \( r_\theta = (r_1)^n \), etc.

- **Exclusive Choice**: Each service \( v \) among the alternative services in the exclusive choice construct has a probability \( \rho \) that it will be executed and \( \sum_{i=1}^{l} \rho_i = 1 \). The aggregation of trustworthiness and cost in the exclusive choice is the sum of that of each component service multiplied by its probability.

- **Discriminator**: Since a discriminator construct only fails if all constituent services fail, its reliability is as follows:

\[
r_\theta = 1 - \prod_{i=1}^{n} (1 - r_i) \tag{4}
\]

However since all component services are executed, the reputation takes all services into consideration as in equation (3) and similarly the cost in this case is the sum of the cost of all components.

- **Multichoice with Multimerge**: In this construct each non-exclusive service \( v \) is associated with a probability \( \rho \) that it will be executed. Note that in this case \( \sum_{i=1}^{l} \rho_i \geq 1 \) due to the non-exclusiveness. The calculation of reliability and cost in multichoice multimerge case is similar to that in the discriminator construct. The reputation considers both the probability of execution and weighting of the component services:

\[
p_\theta = \prod_{i=1}^{n} p_i^{\rho_i \cdot \beta_i} \tag{5}
\]

#### 2) Aggregation of Security Properties

For security properties, the level of security for a property in a composition follows the weakest link approach. Therefore, we calculate the score for a property \( \sigma \) for a CS with \( m \) components as:
The level of security for the composition based on $b$ security properties where each property $\sigma$ has a weighting $\gamma$ ($\gamma \geq 0$) is calculated as follows:

$$s_{cs} = 1 - e^{-\sum_{k=1}^{b} (\gamma_k \cdot \sigma_k)}$$

This means the bigger $\gamma$ value the more the property’s change affects the security level and consequently the trustworthiness.

A threshold for the minimum allowed security level e.g. $s_{cs} = 1$ is also set. The description of how $s_{cs}$ is aggregated with other trustworthiness properties is in section IV.

IV. TRUSTWORTHINESS PREDICTION OF A SERVICE

Here we propose an algorithm that is more efficient than those proposed for multiagent systems in REGRET [6] and FIRE [7] since there is no need to recursively run through all the ratings with each new rating received. In this algorithm, the reputation is determined using moving averages that are updated with every new rating. Older ratings reduce in value over time. The comparison with those algorithms is further discussed in the evaluation.

The reputation $p_v$ is determined by two values; the reputation score $u_v$ for service $v$, $0 \leq u_v \leq 1$ and the confidence $f_v$ in the score and $0 \leq f_v \leq 1$. Both of the two values (i.e. reputation and confidence scores) are important in indicating the status of a composite and component services. Reduction of the reputation score signifies receiving consistent bad ratings of the service while reduction in confidence indicates either low number of ratings received recently or significant fluctuations in the rating scores. Those fluctuations may for example indicate that a service is not scalable enough to meet demands during peak times. We calculate the reputation score $u_v$ as a dynamically weighted moving average of the service’s rating scores. When a new rating is received the reputation score is updated.

First we update the total weight of all received ratings. The weighting is based on the recency $w_t$ and the category of the ratings $w_g$. Recency weight indicates how recent are the ratings received for the service. The more recent the ratings the higher the weight since future ratings are more likely to be close to the latest ratings. Reputation ratings of a service can be classified into a set of categories or types with different weights depending on the way they are gathered. Examples of types of user ratings may include feedback on satisfaction, value, speed, etc. A service composer might not value those categories equally and hence the customisable category weighting.

Recency weight $w_t$ decays exponentially and $0 \leq w_t \leq 1$, as follows:

$$w_t = e^{-\lambda \Delta t}$$

where $\lambda$ is the decay constant, a customisable positive number that controls the rate of decay; and $\Delta t$ is the age of the rating i.e. the difference between the current time and the time when the rating took place.

The accumulated weight of the reputation score $w_v$ ($w_v > 0$) is updated as follows:

$$w_v = w_v \cdot w_t + w_a$$

where $w_a$ is weight of the new rating $a$ calculated as follows:

$$w_a = w_{t_a} \cdot w_{g_a} \cdot \Omega$$

where $w_{t_a}$ and $w_{g_a}$ are the recency weight and category weight for the rating $a$; and $\Omega$ is the credibility value of the rater which is used to protect from malicious raters. For a rating that is generated at the time of calculation i.e. $\Delta t = 0$ and $w_a = w_{g_a} \cdot \Omega$, the new accumulated weight:

$$w_v = w_v \cdot w_t + w_{g_a} \cdot \Omega$$

\[ \text{TABLE I} \]

<table>
<thead>
<tr>
<th>Construct</th>
<th>Reputation ($p_v$)</th>
<th>Reliability ($r_v$)</th>
<th>Cost ($c_v$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>$\prod_{i=1}^{n} P_i^{\beta_i}$</td>
<td>$\prod_{i=1}^{n} r_i$</td>
<td>$\sum_{i=1}^{n} c_i$</td>
</tr>
<tr>
<td>Parallel</td>
<td>$p_i^{n \cdot \beta_i}$</td>
<td>$r_i^{n \cdot \beta_i}$</td>
<td>$n \cdot c_i$</td>
</tr>
<tr>
<td>Unordered sequence</td>
<td>$(\sum_{i=1}^{n} (\rho_i \cdot p_i))^{\beta_i}$</td>
<td>$\sum_{i=1}^{n} (\rho_i \cdot r_i)$</td>
<td>$\sum_{i=1}^{n} (\rho_i \cdot c_i)$</td>
</tr>
<tr>
<td>Loop</td>
<td>$\prod_{i=1}^{n} P_i^{\beta_i}$</td>
<td>$1 - \prod_{i=1}^{n} (1 - r_i)$</td>
<td>$\sum_{i=1}^{n} c_i$</td>
</tr>
<tr>
<td>Exclusive choice</td>
<td>$\prod_{i=1}^{n} P_i^{\beta_i}$</td>
<td>$1 - \prod_{i=1}^{n} (1 - r_i)$</td>
<td>$\sum_{i=1}^{n} (\rho_i \cdot c_i)$</td>
</tr>
</tbody>
</table>
To facilitate the recalculation of the trustworthiness level when new ratings are received, the values of $w_v$ and $u_v$ are stored after each update. The following is the formula for updating $u_v$ after receiving a new rating.

$$u_v = \frac{(w_v - w_a) \cdot u_v + w_a \cdot a}{w_v}$$  \hspace{1cm} (10)

Since confidence reflects both the frequency of receiving new ratings and the stability of their values as described earlier, we calculate the confidence value of service $v$, $f_v$, as:

$$f_v = f_{\eta} \cdot f_{\delta}$$  \hspace{1cm} (11)

We call $f_{\eta}$ the rating quantity confidence indicating how frequent new ratings are received; and $f_{\delta}$ the rating quality confidence which indicates the stability of the ratings values. The more frequent and stable the ratings the more the confidence i.e. certainty in relation to the calculated reputation score. $f_{\eta}$ is calculated as follows:

$$f_{\eta} = 1 - e^{-\alpha \cdot w_v}$$  \hspace{1cm} (12)

where $\alpha$ is a constant parameter that can be used to adjust the slope of the relationship between the sum of the ratings’ weights and the quantity confidence. The higher the value of $\alpha$ the faster the full confidence (i.e. 1) is reached. It can be set to any positive value but for gradual increase in confidence it should typically be set to a value between 0 and 1. The confidence increases in proportion to the number of ratings and to the degree of their recency.

The quality confidence $f_{\delta}$ is calculated as follows:

$$f_{\delta} = 1 - d_v$$  \hspace{1cm} (13)

where $d_v$ is the deviation history of the ratings around the reputation score, calculated as in equation (14):

$$d_v = \frac{(w_v - w_a) \cdot d_v + w_a \cdot |u_v - a|}{w_v}$$  \hspace{1cm} (14)

To help update the reputation when new ratings are received, the value of $d_v$ is stored after each update. $|u_v - a|$ is the absolute difference between the overall reputation score and individual rating score. $f_{\delta}$ indicates the deviation of the ratings around the overall reputation score and ranges between 0 (highest deviations) and 1 (lowest deviations).

### A. Optimal Service Composition

For optimal selection of a component service for service compositions, we use the following formula:

$$\max \left( \tau \cdot T_{cs} + \frac{\varsigma}{C_{cs}} \right)$$  \hspace{1cm} (15)

where $C_{cs}$ is the cost of the CS and $T_{cs}$ is a representation of the trustworthiness calculated from security, reliability and reputation’s score and confidence as in equation (16) omitting the $cs$ subscript for all variables. $\tau$ and $\varsigma$ are constants used to normalise the values of trustworthiness and cost respectively and to customise their priority.

$$T = (u \cdot f) \cdot r \cdot s$$  \hspace{1cm} (16)

To optimise service selection allowing to choose among the best component services as per the computation techniques described in this paper, an optimisation solutions is needed. Since the trustworthiness levels and costs of component services have discrete values and because of the non-linearity of those attributes, linear programming and other solutions that require continuous variables and/or linearity are not suitable. Additionally, the number of services to select from may be large making heuristic methods a better option to provide fast and good results. Genetic algorithms (GA) are well-suited to these kinds of problems. A custom GA is required to suit the characteristics of the problem of service composition.
the parents as cell arrays, and returns the children that result from a two-point crossover by exchanging randomly selected sections in the parents' matrices. The custom mutation function randomly selects two elements in a row of a parent and swaps their values. Since all elements but one are set to 0, the mutation may have an effect only if the value of one of the affected elements equals 1. The number of generations can be fixed to a constant number or set to be proportionate to the number of service types and number of services. Figure 3 shows the improvement of the score of best composition over 50 generations for a simulation of services. Note that the problem is converted to a minimisation one. In the simulation there are 10 types of services organised in constructs as illustrated in Figure 4 with each type having between 5 and 10 concrete services. The cost and trustworthiness of the services are randomly assigned.

B. Comparison to Other Approaches

Figure 4 illustrates an example composite service used in the simulation that includes the composition constructs discussed earlier. Simulation services continue to receive new ratings over the duration of their runtime. The arrival time for service requests is based on Poisson distribution and the mean for the requests changes over time peaking towards the end. Ratings are created based on service executions results and their values vary between services, the time of the rating and whether there is an increased demand. The high demand is set to cause consistent low performance (resulting in mainly low reputation score) or fluctuations in performance (resulting mainly in low reputation confidence) in some of the services. A Gaussian random number generator is used to generate new ratings where the mean and the variance depend on the component service, its type (functionality), and the time of rating (e.g. high demand).

During the simulations each of the services receives a rating for each request. The ratings trigger the update and checking of the trustworthiness of the composite service. Figure 5 shows the evaluation of the trustworthiness of the services using our approach compared to that using other approaches including:

- averaging of the reputation of components as proposed in [15].
- taking the minimum reputation of the components as the reputation of the composite service based on the weakest link concept where the reputation of the composition is as good as that of its component with the lowest reputation.

In Figure 5 (A) only three out of ten component services significantly decrease in their trustworthiness during the peak time. Despite the low reputation of three component services the reputation using the averaging technique still high. The
The trustworthiness of a CS may be affected differently by changes in trustworthiness of its components if they are part of constructs requiring different approaches for the calculation. For example, as illustrated in Figure 8 (A) a moderate decline in the reliability of a component service executed in a loop results in a significant decline. In this experiment $v_1$ is executed three times in the CS. Also note that in this case reliability values are more significant than cost of a component service because of the exponential effect of changes in reliability. In the case of exclusive choice construct the trustworthiness of a CS is only partially affected by decline of the trustworthiness of one of the construct services depending on its probability of execution (see Figure 8 (B)). All three component services have equal probability ($33\frac{1}{3}\%$). In a discriminator construct \{v_1, v_2, v_3\} as in Figure 8 (C). This decline does not affect the CS as the calculation method suggests as long as other services in the construct maintain their reliability. The reliability of the CS here is less than that of component services because of other sequence component services in the CS. Worthy of notice in this construct is that reputation and cost of a component play more significant role than its reliability since the reputation is the product and the cost is the sum of the respective values of the component services.

VI. Related Work

In the past decade there has been large amount of activity in the area of computational trust and reputation, with applications in security, multi-agent systems, game theory, and spam filtering [5]–[9]. The terms trust and trust models are also used in Web service standards (e.g. WS-Trust [10]) but they are limited to the context of being able to trust the identity of the service [11]. However, establishing a service’s identity does not mean that the service itself is trustworthy. For instance, such an authenticated service could be temporarily unreliable or unavailable.

There is also work on trust and reputation specific to the services domain. In [12] the authors present an agent based trust model for service reputation that enables rating of individual services as well as providers. The model has shortfalls, such as the need for human intervention. A framework for reputation based service selection is proposed in [13]. Service consumers submit their ratings to a Reputation Manager Service which computes the service’s reputation based on those ratings. Singh [8] discusses the challenges of trustworthy service composition. He states that current approaches fail to adequately address the challenges for trust in service oriented computing. Techniques of aggregation of QoS were described by Hwang et al [15] together with reputation where averaging of reputation of the components was proposed as discussed in section V-B. No techniques were discussed on aggregating security properties. In [14] the authors introduce a framework for establishing trust in service oriented environments named...
RATEWeb. The framework operates by aggregating reputation ratings from consumers in a P2P fashion. It aims to support the use of trust in service selection and composition. However, unlike the work described in this paper, it does not consider the computation of trustworthiness in a composite service. It also does not provide mechanisms for responding to dynamic changes in the environment that for instance may affect a service’s trustworthiness.

VII. Conclusion

This paper has presented an approach to monitoring and predicting the trustworthiness of composite services. The dynamic plugin-based trustworthiness module continuously monitors the adherence of the services to their contracts and receives ratings, metrics and alerts relating to the reputation, QoS, security and other events. Service ratings are generated using a rules engine and stored in the module’s trust ratings’ store. The calculation of trustworthiness and cost depends on the construction of the composite service.

Ongoing and future work includes the development of novel optimisation mechanisms for dynamic service composition and adaptation to provide robust algorithms that take into account multiple objectives including trustworthiness, cost and pricing. Another area under development is the use of access control and resource management techniques to manage and maintain the trustworthiness of composite services.

Acknowledgement: The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos) [1].

REFERENCES

[1] Aniketos (Secure and Trustworthy Composite Services), http://www.aniketos.eu
[5] L. Xiong, L. Liu, PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities, IEEE Transactions on Knowledge and Data Engineering v.16 n.7 p.843-857, 2004
[12] E. Maximilien, M. Singh, Agent-based trust model involving multiple qualities, Proc. 4th Int. Conf. on AAMAS, 2005