MOVING REPUTATION TO THE CLOUD

C. Hillebrand* and M. Coetzee†

* *Academy of Computer Science and Software Engineering, University of Johannesburg, South Africa*
SAP P&I BIT Mobile Empowerment, Pretoria, South Africa
E-mail: channel.hillebrand@gmail.com
† *Academy of Computer Science and Software Engineering, University of Johannesburg, South Africa*
E-mail: marijkec@uj.ac.za

Abstract: Reputation is used to regulate relationships of trust in online communities. When deploying a reputation system, the requirements and constrains of the specific community needs to be accommodated in order to assist the community to reach their goals. This paper identifies a need for a framework for a configurable reputation system with the ability to accommodate the requirements of a variety of online communities. Such a reputation system can be defined as a service on the Cloud, to be composed with the application environment of the online community. Consequently, this paper introduces the concept of RaaS (Reputation-as-a-Service) and discusses a potential framework to support the creation of a RaaS. In order to define the framework, research is conducted into features of SaaS (Software-as-a-Service) architecture components, user requirements for trust and reputation, and features of current centralized online reputation frameworks that can be configured in order to support a reputation service on the Cloud.

Key words: Reputation-as-a-Service, trust, reputation, Software-as-a-Service, reputation framework, cloud service.

1. INTRODUCTION

Online shopping has grown significantly in the past years and it is predicted that such sales will increase annually by 10% for the next 4 years [1, 2]. People are influenced by product reviews to make purchasing decisions and therefore tend to buy from online stores with a good reputation [1]. As online shopping is characterized by insecurity, anonymity, lack of control and potential opportunism, online communities should take the necessary steps to ensure that participants are trustworthy.

For online trading communities such as eBay, a centralized online reputation system is used to compute and publish reputation scores for service providers, services, products or entities such as buyers and seller within a community. The reputation score reflects the collection of opinions or ratings that entities have about the objects. Ratings are provided to a reputation algorithm to compute reputation scores [3]. In order to be effective, reputation managers need to accommodate the specific needs of the communities where they are deployed.

Consider the example of Organization ABC, an online store for a start-up company that sells products to consumers over the mobile web. As trust and reputation is a major component to enable m-commerce, the online store of Organization ABC needs to deploy a reputation system to control trust relationships between consumers, suppliers and their portal. As there is no off-the-shelf reputation system to integrate into their application environment, and it is expensive to custom develop, the m-commerce web site may initially be implemented without it. Ideally, Organization ABC needs a reputation system that is simple to use, with easy to understand ratings between 0 and 5 to ensure the growth of the community. In another type of online community, where crime incidents are posted and recorded with mobile phones, a reputation system is needed to ensure that no malicious or false incidents are reported. The requirements for this reputation system may be very different to those of the online store of Organization ABC. This highlights that a configurable or customizable reputation system is needed that can support multiple online communities in a cost-effective and efficient manner.

Recently, a business model for software applications namely SaaS (Software-as-a-Service) has emerged which lowers the cost of development, customization, deployment and operation of applications [4]. As SaaS applications generally support the concept of software application configuration and customization, this research proposes to present a configurable reputation system as a SaaS solution. Here, a multi-tenant architecture is followed where organizations pay only for the features that they access, and are able to configure or customize the reputation system to suit their community's needs.

The contribution of this paper is to identify requirements and challenges in order to define a RaaS (Reputation-as-a-Service) framework. As trust and reputation systems can be very complex, the focus of this research is the definition of a RaaS framework that provides similar but configurable functionality currently supported by central online reputation systems.

In the next section, trust and reputation is defined for this research. Five general components of reputations systems are given which is referred to throughout the paper. The requirements for a RaaS component is identified by considering SaaS configuration aspects, user requirements for trust and reputation and finally requirements from

Based on "Towards reputation-as-a-service", by C. Hillebrand and M. Coetzee which appeared in the Proceedings of Information Security South African (ISSA) 2013, Johannesburg, 14 to 16 August 2013.

reputation frameworks. A RaaS framework is presented and the paper is concluded.

2. TRUST AND REPUTATION

Trust and reputation is present in a variety of online communities. Trust is the individual's perspective on a particular service or product and reputation is a group's perspective on a particular service or product [5]. As trust and reputation are concepts that are often used interchangeably, they are now defined for the purposes of this research.

2.1 Trust

Trust is challenging to define as it manifests itself in many different ways in varying contexts. Almost every aspect of daily life is supported by some form of trust. For example, in Figure 1, consumer X, the trustor, orders products from organization ABC, the trustee. For this research, the following definition of trust is adopted. The trust of consumer X in organization ABC is defined as the level of subjective probability that organization ABC will deliver high quality products on time [6].

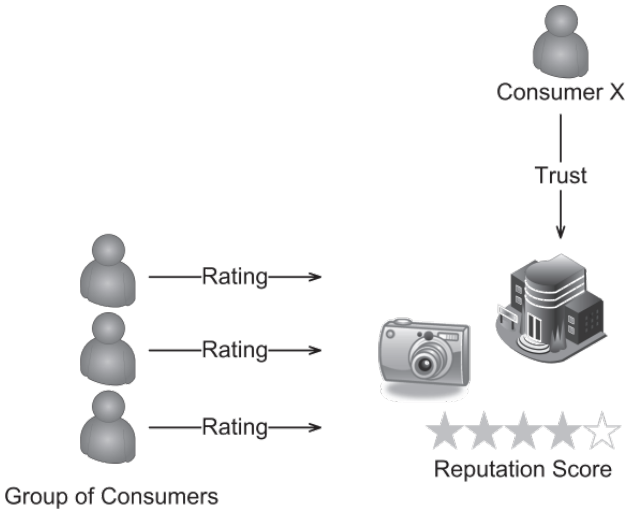

Figure 1: Trust and Reputation

The trust of consumer X in Organization ABC is affected by trust properties such as transitivity, subjectivity and the asymmetric nature of trust [7]. If Organization ABC has the reputation of delivering high quality products, consumers automatically assume that any product of Organization ABC is also of high quality due to the property of transitivity, suggesting that trust is transferable. But, as both consumer X and Z can have different levels of trust towards the same organization ABC, trust is subjective. The asymmetric property of trust is defined by the fact that consumer X needs to trust that organization ABC will deliver the necessary services, but organization ABC needs to trust consumer X to pay on time.

Closely related to trust, is reputation. In the next section the concept of reputation is addressed in order to identify elements that it consists of.

2.2 Reputation

Reputation can be considered as a collective measure of trustworthiness [8]. In order to better regulate relationships in online communities, opinions about interacting parties' past behavior is collected and aggregated in order to define a summary evaluation, or reputation. The predictive power of reputation supposes that past behavior of a participant is indicative of their future behavior.

In Figure 1, reputation is illustrated by a group of consumers' opinion on a specific product. The group of consumers in Figure 1 gives a product a good rating over time, ensuring that the product has a good reputation score [9]. In this paper the term "rater" is used to represent a participant or consumer who assigns ratings for others. Reputation is calculated by incorporating past experiences, direct experiences and recommendations using various algorithms and models for this purpose [10]. One party can thus trust another based on their "good" or "bad" reputation.

A reputation system is an application that facilitates the process of calculating and evaluating reputation for a specific community. The five main components found in reputation systems and models are [11]:

1. *Gathering behavioral information* where direct experiences, and experiences of acquaintances of consumers, recommendations from others, transaction history, pre-trusted entities and raters reliability are collected.

2. *Scoring and ranking* of entities are done next resulting in a reputation score, computed using averages, fuzzy logic, or Bayesian networks.

3. *Entity selection* is done next using the reputation score and other utility functions as specified.

4. *Transaction* is carried out with the selected entity.

5. *Reward and punishment* is finally given by assessing the transaction and giving a rating.

Most current reputation systems are built using these common components, but for a specific context and application domain, using proprietary vocabularies [12]. Each defines its own method to query, store, aggregate, infer, interpret and represent reputation information.

In order to be able to define a cloud-based reputation service that can be usable by different communities, the next section investigates the requirements that such a reputation service framework or RaaS framework should comply to.

3. REQUIREMENTS FOR A RAAS FRAMEWORK

In order to determine requirements to define a RaaS framework, this research now reports on the two main drivers for a RaaS framework namely general SaaS application requirements and current state-of-the-art reputation system requirements to create a comprehensive list of requirements for the RaaS framework.

3.1 Requirements for SaaS applications

SaaS applications are deployed on cloud infrastructures and exposed to applications or users to be consumed over the Internet. SaaS is pay-per use, meaning organizations only have to pay for the features they want to use in an application, making it a cost effective solution [13, 14]. The main motivation for organizations adapting SaaS applications is to reduce IT support costs, ensure business agility and outsource hardware and software maintenance. SaaS architecture, design attributes and application requirements are now investigated to identify requirements that a cloud reputation service needs to meet.

SaaS application architecture: A SaaS application such as a reputation service or RaaS is integrated with existing enterprise systems, as shown in Figure 2. The architecture is based on service-oriented architecture design principles.

To create a cohesive application that incorporates a RaaS, the orchestration of the flow of data from the service to the end user is crucial. For example, it is important to send data such as a valid rating for a specific product to the right retail system at the right time. Furthermore, SaaS-to-enterprise integration poses challenges such as semantic mediation, data quality, interface mediation and other logical operations when moving data between domains so that the data is useable when it reaches the target system. Depending on the business requirements and integration capabilities of the chosen SaaS product, the integration approach is generally not trivial. Even though SaaS applications are exposed via comprehensive APIs to ease the integration process, it may still be the case that a custom SaaS integration layer is needed.

To support application integration, SaaS application architecture generally consist out of 3 main layers namely the consumption, service and data layer [13]. Starting at the bottom of Figure 2 the service and data layers are found in the *SaaS component*. Here are reusable software components and their data are exposed as services preferably using REST APIs (Representational state transfer) [15]. Messages are formatted in in JSON (JavaScript Object Notation) [16] format style or in XML (Extensible Markup Language) [17] format.

The service layer is defined by multiple sub-layers such as the service wrapping, schedule and service technology application layers. Support is provided to administrators of tenants to customize services. To persist SaaS specific data, the data layer stores rating, reputation and other relevant information.

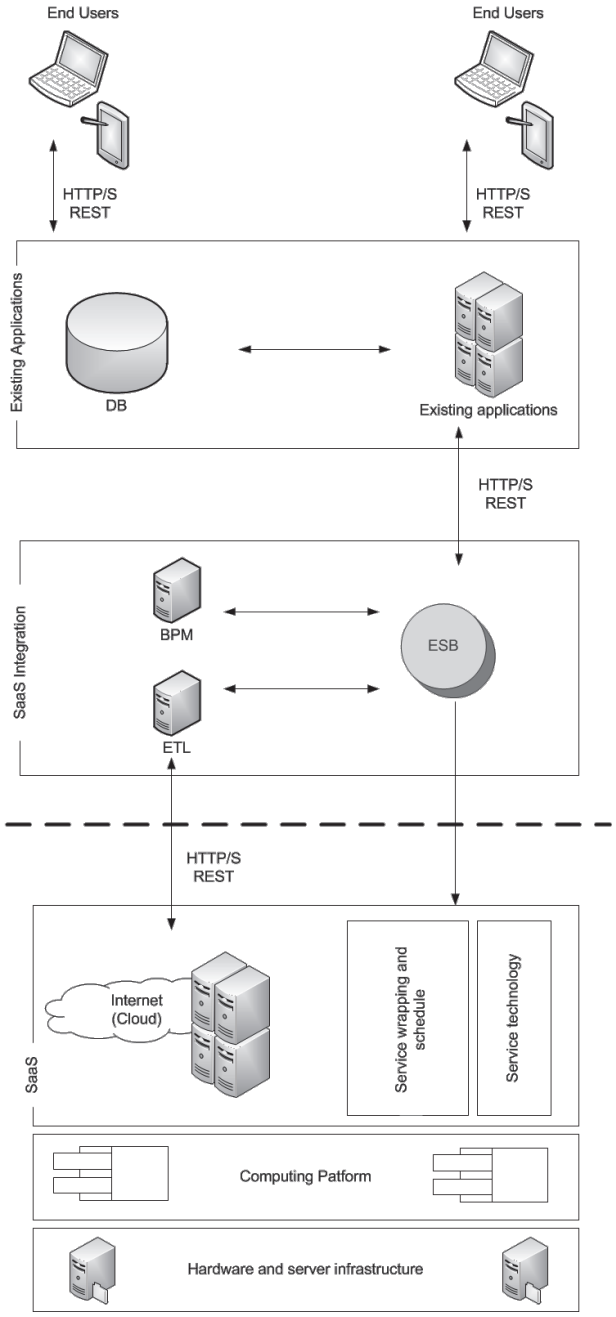

Figure 2: SaaS Architecture Layers

Towards the middle of the architecture, enterprise integration is supported by the *SaaS integration component* using integration products such as an ESB (Enterprise Service Bus) that provides orchestration capabilities. In this way, enterprise activities performed by existing enterprise applications are combined with activities of SaaS applications.

Finally, the consumption layer presents end users with an integrated view of information generated by these orchestrated applications.

To be successfully integrated into other enterprise applications, an important requirement for SaaS applications is to

ensure that integration with other applications is seamless. To ensure that many types of enterprises can integrate SaaS applications, design attributes, discussed next should be addressed.

SaaS design attributes: In order to design a RaaS component, several design attributes should be taken into consideration to ensure conformance to a typical SaaS application [18]:

- Multi-tenancy to ensure that a single instance of a RaaS can be used by multiple tenants and their clients with different needs and functionalities.
- Single version provides the capability that only one version of the RaaS is exposed to clients.
- Logical data separation accommodates the storage of different tenants' data such as configuration and rating data is in their own data domain.
- Application integration, as mentioned in the previous section, should be supported so that applications of tenants can integrate functions of the RaaS into their applications with ease.

To ensure the flexibility of SaaS applications, an application is configured in such a way to support design attributes and integration processes. Next, SaaS application requirements for RaaS are discussed to support configuration.

SaaS application requirements for RaaS: The focus of a RaaS model is to deliver software functions to many clients over the Web with a single instance of a software application running on a multi-tenancy platform [19]. However, every tenant that needs to use a RaaS supported with this model can be unique, requiring changes to the reputation system.

Tenants may have a different industry focus, their customers may behave differently, they may support diverse product offerings and have different regulations, organizational culture and operational strategy. These features require RaaS components to be tailored, by leveraging two major approaches namely configuration and customization [19].

Configuration does not involve source code change of the RaaS application and support differences through setting pre-defined parameters, or leveraging tools to change application functions within pre-defined scope, such as adding data fields, changing field names, modifying drop-down lists, adding buttons, and changing business rules. On the other hand, customization involves RaaS application source code changes to create functionality, leading to a more costly approach for both SaaS vendors and clients.

There are seven fundamental configuration and customization requirements that can be tailored, to make the RaaS component as flexible as possible [19] namely:

- Support for different organization structures require the ability to add, delete and changes roles.
- Support for different types of data can be made possible by adding custom fields and types, and deleting data not needed.
- Support for different processes requires tasks to be switched, added and reordered and their roles to be changed.
- Business rules can be modified by changing or setting rules and the rule triggers.
- Reputation computations can be made more generic by adding or changing actions or triggering actions at different points.
- The user interface can be changed with respect to the look and feel, the data presented and the addition of data.
- Reporting can be changed with respect to style, dataset used and query rules.

In summary, the RaaS should be developed to have standardized software features to serve as many clients as possible using a configuration approach. The RaaS developer needs a strategy to enable self-defined configuration by their tenants without changing the SaaS application source code for any individual tenant [19].

The RaaS environment needs to be thoroughly analyzed to determine the common configuration requirements. In conjunction, a sophisticated web based tool is needed to allow clients to configure the RaaS service themselves.

The next step is to investigate the second main driver to determine RaaS requirements, namely those stemming from trust and reputation systems.

3.2 Trust and reputation system requirements for RaaS

Centralized online reputation systems, which is the focus of this research, collects users' opinions on products, transactions and events as reputation information, to aggregate and publish it. Many trust and reputation models have been proposed, each targeting different contexts, with their own unique features. While most research focuses on addressing the ever-increasing complexity, not much attention has been paid to the process of integrating reputation systems into applications. The next section has the aim of identifying a set of basic requirements to be addressed the RaaS by firstly investigating real-world user requirements, and then general trust and reputation system components.

User requirements for trust and reputation systems: Previous research [10] collected formal user requirements for trust and reputation systems from system developers. It was found that users required a clear, layered and pluggable architecture for representing the calculation process of the trust score. Categorized user requirements were found to be closely coupled with the previously discussed five components found in reputation systems [20].

User needs for each of the first three components were identified as follows:

- Information Gathering
  - The success of each interaction needs to be rated and quality parameters continuously monitored.
  - Simple and intuitive rating scales should be used.
  - The quality parameters of a service should be controlled and certified by a trusted party; ratings of such a party can be used as a starting point for trust computation.
  - Raters reliability must be controlled as they could provide dishonest ratings.
  - Initial rating should not influence or bias subsequent votes.
  - Similarity between recommenders' preferences should be considered.
  - Trust values decay and become invalid over time.
- Scoring and Ranking
  - There is a need for a single trust rating which is calculated by taking into account different service aspects and their weights.
  - The calculation should be an aggregation of all weighted aspects, similar to an "average".
- Entity Selection
  - When entities or services are selected, they should be sorted according to their trust rank and made comparable to each other.

By considering such a user-centered design approach, the proposed RaaS component can be created to fulfill the needs of users as far as possible. General reputation framework requirements are discussed next.

Reputation system framework requirements for RaaS: The focus of this section is to identify major characteristics of reputations systems to identify requirements for the RaaS component. In order to achieve this, an adapted framework is defined from the work of others [21–23]. There are eight elements which are discussed following the phases of reputation management components from information gathering, to scoring and ranking and entity selection. The elements, shown in Figure 3 include:

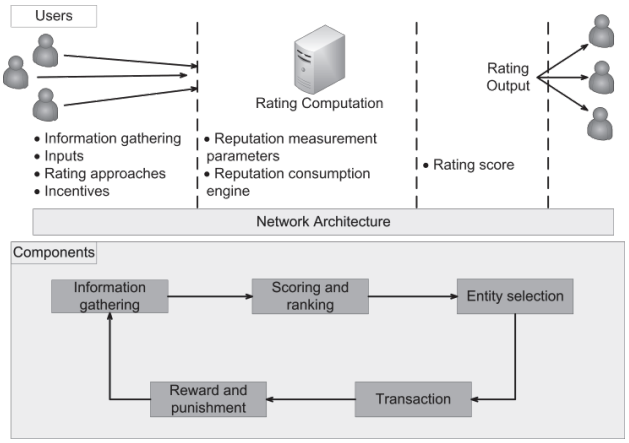

Figure 3: Reputation Manager Framework

1. Network architecture
2. Information gathering
3. Inputs
4. Rating approaches
5. Incentives
6. Reputation measurement parameters
7. Reputation computation engines
8. Rating score

Figure 3 shows how the eight elements at the top of the diagram fit in with the first three components at the bottom. Network architecture supports the framework across all components, whereas information gathering, inputs and rating approaches and incentives are found under the *Information Gathering* component. Reputation measurement parameters and reputation computation engines fall under the *Scoring and Ranking* component and rating score under the *Entity Selection* component.

Each of these elements are now described, starting with the network architecture.

a) Network architecture: The network architecture of a reputation system can either be a centralized, decentralized or hybrid architecture [8]. The network architecture determines how information is gathered and stored. For this research, the RaaS component needs to follow the cloud architecture, thereby limiting the scope of architecture choice. In a centralized architecture, all data is stored in a central repository with all reputations scores publicly available to participants. For distributed or hybrid reputation systems, there is no central point where ratings are submitted or feedback can be obtained. Instead, each participant is given the responsibility to collect ratings from others. For a RaaS component this is not a viable option as the cost and complexity level would be too high. A centralized architecture is simple and cost-efficient, and

conforms to the RaaS and user requirements identified previously. This choice directly influences the discussion of the next elements, as they need to comply with this requirement.

Next the elements related to the *Information Gathering* component are discussed. For this research, a transaction between two participants is the basis of a rating. Generally, a participant cannot rate another one without having had a transaction with him. After a transaction, participants usually have no direct incentive for providing rating about the other party. The information gathering phase should be carefully designed to address this issue.

b) Information gathering: The information gathering phase collects rating inputs over a period of time. There are a numbers of important aspects to consider such as [24]:

- the collection channel,
- the information sources,
- the number of raters, rating granularity and reputation of raters,
- collection costs.

Collection channels can be direct or indirect. Direct channels collect information from raters just after the transaction, by sending emails asking them to do a rating or by using a 3rd party for rating collection. Indirect channels collect information from other reputations systems, increasing complexity of information gathering.

Most online reputation systems consider reputation ratings from a global perspective. For example, eBay's feedback forum provides feedback profiles of sellers and buyers publicly to the all users. The shortcoming of such a global view is that these values lack personalization [5]. Information can be gathered from past experiences, direct experiences and recommendations [20]. The gathered information plays a major role to calculate a reputation score for a particular user or product. This information might come from several sources such as direct experiences with the targeting entity, neighbours of participants, acquaintances, the group the participant belongs to or organizations. In this regard it is important to consider the set of raters, their expertise and credibility [24].

A sufficient number of raters who rate transactions can help a reputation system to avoid personal bias whereas a restriction on the number of raters may influence level of detail between raters and objects being rated. A reputation system can be defined to have no restrictions on the number of raters leaving ratings, which means anyone can rate; or only registered participants can provide a rating; or only some registered raters can provide a rating after a transaction has finished such as eBay allows. It should not be allowed to rate a transaction or object more than once, for example, in eBay if buyers and sellers transact, the reputation system will only allow one rating per transaction to avoid the manipulation of the reputation score.

Directly related to the number of raters who rated objects is granularity [25], which indicates if the model is context-dependent or not. As raters may have a good reputation for their expertise in one domain, and a low reputation for another, granularity identifies how information sources associates to the reputation object. When a system allows any raters to do a rating, the granularity is usually very loose. If a reputation system requires information sources to have a good credibility to leave reviews this increases the cost for a rater to provide a rating which in turn reduces the number of invalid ratings.

The reputation of the rater should be considered by having other participants to give feedback on those ratings. Some reputation systems have a ranking mechanism for their users, called the Karma mechanism that records every action of a user and gives points to it [23].

Finally, the input collection costs should be considered. This is the cost that indicates how much time it takes to collect a single unit of reputation information, where collection channels can have an important effect on this cost [24].

Next, the type of information source is described.

c) Inputs: Different information formats can be chosen based on the way in which they will be used in a reputation system. Some reputation systems support arithmetic operations and other evidence where numeric quantification is more appropriate. It can also be possible to provide a mapping from qualitative to numeric labels. For example, ratings such as a score between 0 and 10 can easily be aggregated to an overall score, to give a comparable value between reputation objects. On the other hand, text reviews contain detailed information which can be very useful.

Generally, a rating can be expressed as either a quantitative or Boolean format [21]. A quantitative metric is a measurable input such as a value between 0 and 10 whereas a Boolean format is either 0 or a 1 to represent "like" or "dislike". As it is important that the reputation score is useful to the community where it will be used, the RaaS can be configured for this purpose.

In order to ensure the completeness of ratings collected, rating approaches are discussed next.

d) Rating approaches: A larger variety of rating information can give a better view of a reputation object as it provides a more complete picture. For example, travel reputation systems can allow participants to rate hotels for their value, rooms, location, cleanliness and service separately [24].

In single-criterion rating systems or binary rating systems, participants reveal their general opinion with regards to a

reputation object, resulting in reputation information that is not too reliable and accurate.

In systems where multiple-criteria can be used, better quality reputation scores can be defined. A set of criteria needs to be defined and a rating is provided for each. This can allow a participant to choose a partner based on specific criteria that matches his own. On the down side, many rating criteria may reduce the evaluators' motivation on leaving ratings. This can be overcome by making some criteria optional to rate.

Next the role of incentives in information gathering is discussed.

e) Incentives: Raters of a reputation system may have different motivations for providing ratings. Incentives are important as their absence drives only some of the users to voice their opinions and report feedback where those with a moderate outlook are unlikely to provide ratings [25]. This results in an unrepresentative sample of ratings and opinions. For example, reputation systems have incentives for raters such as sellers to behave honestly in order to be chosen by buyers as this can increase their profit through the increased amount of transactions. These incentives are necessary because fabricated ratings can promote specific sellers or to discredit others - e.g. authors can write fake reviews on Amazon in order to boost the sale of their own books. In order for RaaS to be implemented successfully, the motivations for providing a rating should be identified.

There are various types of motivations [23] such as altruistic motivation which is in favour of doing good to users being rated and can be classified as tit-for-tat, friendship and exploiting opinionated incentives. Commercial motivation, is used to generate revenue and is categorized as direct revenue incentives and branding incentives. Egocentric motivation is used for self-gratification and is categorized as fulfilment incentives and recognition incentives.

By explicitly rewarding participants for reporting feedback, rewards made by the reputation systems must cover the cost of reporting feedback to encourage more participants to report, giving a more representative set of ratings. In addition, rewards must be designed so that selfish participants are convinced to rate truthfully to advance themselves [25].

The next section now considers the next reputation component namely the *Scoring and Ranking* of ratings. Here, the reputation computation engine and rating approaches are discussed.

f) Reputation computation engines: One of the most critical features of a reputation system is the reputation computation aggregation algorithm. Such an algorithm integrates ratings into one score, and at the same time needs to ensure that bad raters are identified and removed to obtain accurate ratings. There are many complex aggregating algorithms that have been proposed such as fuzzy models and Bayesian systems.

Currently, most online reputation systems as eBay and Amazon choose to use simple algorithms [8], such as summation, average or percentage. Simple summation adds all of the ratings, regardless if it is positive, neutral or negative and the calculation is easily understood and adopted by users [8, 21]. Unfortunately, this feedback metric is flawed, for example, if a user has 10 positive feedback points out of 10 transactions and another has 20 positive and 10 negative feedback points out of 30 transactions, they would have the same reputation score [5].

Average rating is based on the same principle as simple summation, however average rating is perceived as more accurate. Ratings can also be calculated by means of weighted average ratings. This infers that each user has a credibility score that determines their weight ratio [5]. Many interesting aggregating algorithms have been proposed that can be classified into five categories [26]:

- By averaging ratings, simplicity in algorithm design is ensured and low cost in system execution.
- Weightings are introduced by weighting the ratings of acquaintances but those of strangers are averaged.
- Only ratings from witnesses are used, who have interacted with the entity being rated. In such a weighted majority algorithm only the ratings from witnesses are aggregated, and the weight of witnesses is decreased if it differs from self-own recognition.
- The weight of ratings is based on the similarity of the experience between the rater and the other participant to improve accuracy.
- Ratings can be aggregated and weights of raters can be updated through deriving the expectation of the Beta distribution.

In simulation [26] it was found that most complex algorithms will have better results. However, in several circumstances the simple algorithm can outperform the complicated algorithms. In particular, the first average algorithm is found to be more resistant to different type of bad raters [26].

To configure the reputation aggregation algorithm for a RaaS, one of these aggregation algorithms can be chosen as they may be able to accommodate a variety of communities and would be understood and adopted by users [5].

g) Reputation measurement parameters: There are crucial parameters which may increase the accuracy of the expected reputation score namely transitivity rate and time [21].

Transitivity rate represents the fact that recommendations from third-hand ratings with a transitivity degree of three may have the least influence on the trustworthiness measurement. Therefore, in a recommendation chain,

recommendations from known participants who already have had interaction with the requested party should have more weight as first-hand recommendations, than those who are known but have not had any previous interactions with the requested party or those who are unknown.

Time influences the effect of ratings on the computation as the most recent rating will have a higher weight ratio than ratings that are older. Thus ratings decay over time. The advantage is that users benefit from having a rating value that reflects how the most recent services performed. These parameters attempt to ensure that ratings are more accurate as weight ratios are an effective way to counteract "bad" raters.

Finally, the *Entity Selection* component is considered where the resultant reputations is now used.

h) Rating score: The reputation system finally reports its results to users in two different formats namely aggregated reputation scores and individual ratings and opinions [24]. Reputation scores are the result of the scoring and ranking component, whereas the individual ratings are collected through the information gathering component.

When reputation scores are presented, the time line it represents should be provided to assist users with decision-making. Reputation information is disseminated to end users via different access methods such as web sites, emails or RSS (Rich Site Summary) feeds. Certain information may be made publicly available, whereas others may require a subscription fee.

Next, the requirements for a RaaS framework are presented.

3.3 Requirements for a RaaS framework

Table 1 and Table 2 briefly provides the most relevant requirements for a RaaS framework. The requirements are given according to the two main drivers and the first three components of reputation systems. It has been indicated on the table where a high cost is associated with a set of requirements. In the last case, the list of requirements identifies a set of configurable features that should be present.

SOFTWARE-AS-AS-SERVICE REQUIREMENTS			
SAAS application architecture	• SOA layered architecture • Well-designed interfaces • Standards-based messaging • Support for integration and data semantics		
SAAS design attributes	• Multi-tenancy • Single version • Logical data separation		
	Information gathering	**Scoring and ranking**	**Entity selection**
SAAS application requirements using configuration	• Add, delete and changes roles • Custom fields and types, and deleting data not needed • User interface look and feel, data presented.	• Tasks switched, added and reordered and task roles to be changed • Business rules modified by changing or setting rules and the rule triggers. • Reputation calculation - adding or changing actions or triggering actions at different points. **(HIGH COST)**	• Reports changed with respect to style, dataset used and query rules.

Table 1: SaaS Requirements for RaaS

SaaS requirements: From the list of requirements in Table 1, the design of the RaaS component is driven by considering integration features, SaaS design considerations and configuration features. SaaS application architecture and design attributes are requirements that should all be applied when designing and implementing the RaaS and are not configurable in nature.

Configurable features: SaaS application feature requirements shown in the last section in Table 1 provide an indication of the possible configurable options that should be present to administrators of tenants. Important configuration options are those allowing the configuration of input data such as roles, data types such as rating and reputation scores, rules that apply the strictness by which types of raters are allowed to rate or incentives, actions by which algorithms are applied, and reporting of ratings and reputation scores. This list only provides an indication of the types of configurable features that should be present. These features can be fleshed out in more detail after Table 2 is reviewed.

Trust and reputation system requirements: Table 2 summarises the requirements elicited from users, and those from investigating trust and reputation system components. Between these two sets, some overlap can be noted. Both sets highlight that the manner in which ratings are done - preferably after each and every transaction - is important to apply. Furthermore, that the results produced by the reputation algorithm can be understood with ease, that the reputation of the rater needs to be verified, the decay of trust and the sorting of reputation scores needs to be implemented.

Configurable features: Trust and reputation system

TRUST AND REPUTATION SYSTEM REQUIREMENTS			
	Information gathering	**Scoring and ranking**	**Entity selection**
User trust and reputation requirements	• Each interaction needs to be rated • Simple and intuitive rating scales • Initial ratings should be treated differently • The quality parameters of a service should be controlled and certified by a trusted party • Raters reliability must be controlled • Similarity between recommenders is important. • Trust values decay and become invalid over time.	• A single trust rating calculated by taking into account different service aspects and their weights. • Computation should be an aggregated of all weighted aspects, similar to an "average". **(HIGH COST)**	• Services should be sorted according to their trust rank • Providers should be made comparable to each other.
Reputation system framework requirements	• Direct or indirect collection channels • Restrictions on who can rate • Context-dependant ratings • Collection costs • Simple rating format • Single/multiple criteria • Incentives **(HIGH COST)**	• Complexity of aggregation algorithm • Weighting of recommendations • Decay of trust **(HIGH COST)**	• Reputation score / individual ratings • Time reported

Table 2: Trust and Reputation Requirements for RaaS

requirements provide more detail as to the configurable options that should be present to administrators of tenants. Configurable features are now presented according to the three reputation system components.

For the *Information Gathering* component, important configurable features are the type of collection channel, the type of rating information such as from direct experience, the context of the rating such as cost and/or quality of a product, the reputation of the rater and its maintenance, the format of the score and the incentives provided to raters. A high cost factor is the use of collection channels used to source ratings and feedback.

The most complex set of configurable features are those of the *Scoring and Ranking* component, where reputation computations are performed. There are many possible configurable features such as the order of tasks executed, choice of algorithms, and rules and weightings of criteria. The *Scoring and Ranking* component is not trivial to apply and is very complex in nature.

The result of the reputation computation is used in the *Entity Selection* component, where aspects such as the reputation scores and individual ratings can be provided to end users.

Next a RaaS framework is proposed in light of the identified requirements.

4. RAAS FRAMEWORK

A RaaS framework is defined using the requirements defined in this paper. First the architecture is given by considering only RaaS functional components and not business components such as billing and metering. RaaS framework configuration is described followed by the RaaS-to-enterprise integration process.

4.1 RaaS architecture

The RaaS component is integrated with the applications of tenants. As organizations may make use of more than one service eg PayPal, the RaaS is can be more accurately described as a SaaS Mashup service.

Figure 4 gives the architecture of the RaaS component integration, based on how SaaS integration is implemented. At the bottom of Figure 4, the RaaS component is defined over a basic cloud infrastructure found in data centres where hardware and software are used to define virtual machines that are provided to tenants to run their applications on.

The RaaS component exposes REST APIs to tenants to support various features such as input collection channels, users, roles, incentives, weightings, rating measures, calculations, rater reputations, rating criteria, entity selection and reputation reports. For each tenant, the context of interaction needs to be maintained, uniquely supported by their set of configurable features. A configuration tool allows administrators of tenants to configure features.

RaaS API calls and results are composed with those of the existing applications of the tenant using integration tools. End users of a tenant such as Organization ABC do not directly communicate with the RaaS, but interacts with the RaaS component via web interfaces. Reputation data is sent into the system programmatically, using a REST API.

From the list of configurable features given previously, it is

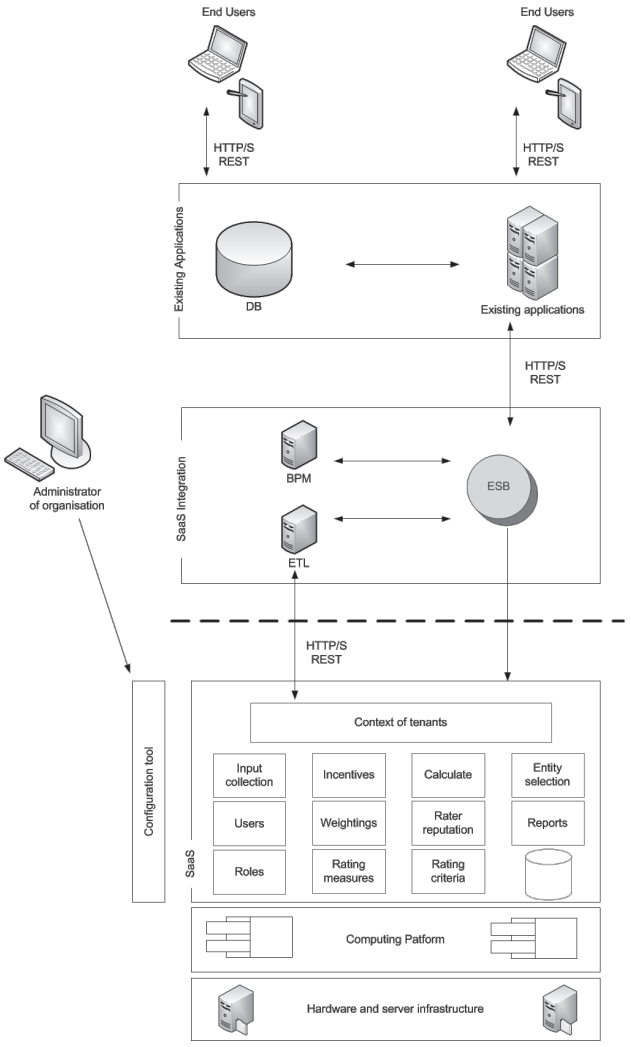

Figure 4: RaaS Architecture

clear that RaaS configuration is not straightforward. This aspect is described next in more detail.

4.2 RaaS configuration

The RaaS architecture supports a configuration tool to allow administrators of tenants to uniquely define features to suit their community needs. The RaaS configuration tool should be designed to be easy and intuitive for the administrators, but at the same time be able to satisfy the needs of tenant requirements. Without this feature, it would be impossible to use the single instance of the software for different tenant applications.

The RaaS component supports many functions such as input collection channels, incentives, users and roles, ratings and reputation calculations. The configuration of these functions is no trivial matter, and much intelligence is required to ensure that options are set that would ensure a reputation score that is a true reflection of the behaviour of quality of the entity being rated. Careful consideration should be given to aspects such as the type of data that is exposed, the type of rating format is required by specific algorithms, whether weights can be set or not, which groups of raters may be granted the ability to rate, which objects can be rated and the number of criteria to be used.

Considering the above mentioned complexities in configuring reputation computation this research now proposes a two level reputation configuration approach, one for novice users, and one for knowledgeable users who understand the implications of their choices.

For novice users, there may be a few options available, based on the risk the organization may experience, to select from such as:

- Low - reputation computation with basic summation of values.
- Medium - reputation computation that encourages strangers by initially bootstrapping their trust to a level to ensure participation.
- Strict - reputation computation that is strict with "bad" behavior as the risk is high.

An advanced configuration panel may be made available to knowledgeable users to select a variety of options.

In both cases, the RaaS should make available a simulation feature that will illustrate to the administrator what the effect of the choice will be, in order to avoid any misunderstandings.

Next, a RaaS application integration example is described.

4.3 RaaS application integration example

The utilization of the RaaS is now discussed according to a set of sample requirements. This discussion now returns to the case of Organization ABC. This organization is an online store for a start-up company that sells products to consumers over the mobile web. It requires a reputation system that is simple and easy to understand, easy to integrate into their applications, favour good behaviour, encourage users to rate, decay older ratings and allow any product to be rated that is sold to customers.

The administrator of Organization ABC configures the RaaS via a configuration menu page after he/she is authenticated by the system. The configuration menu provides the administrator with the capability to view past settings of the RaaS, configure new settings and view reports on the activity of the users within the RaaS.

When an administrator configures a new RaaS, he/she choose the "Low" configuration option based on the organizational requirements that sets initial values for a number features. A few features are further configured as follows:

- Input data such as those relating to the types of entities to be rated, user profiles and rating values (out

of 5) are set. Initial values are provided by the "Low" configuration features that can be adjusted.

- Input collection channel is set to prompt the user directly after a transaction to save on collection costs.
- Incentives are set to be of altruistic nature to encourage users to rate. Rewards are given every time a rater rates a transaction. The rater is given a public reputation score to encourage others to rate.
- Reputation calculation is set to a simple average rating calculation so that users can easily understand and interpret it.
- Rating criteria is set to single, indicating that a product purchased is rated by the buyer.
- Trust decay is set so that previous ratings decay by 50% when older than 6 months and by 100% when older than 18 months.

After configuration features are set, the RaaS is programmatically integrated with the orchestrated calls and responses of the existing applications of Organization ABC.

Figure 5 illustrates a typical web interface displaying rating data. Buyer X views the product search page of Organization ABC on the web browser or app via the mobile device. He has searches for a product and sees its ratings displayed as star ratings out of 5. The page is programmatically created, by sending a REST request such as "https://www.example.com/sendReputation/11", where 11 is the product code that is sent to the RaaS, and displaying the result.

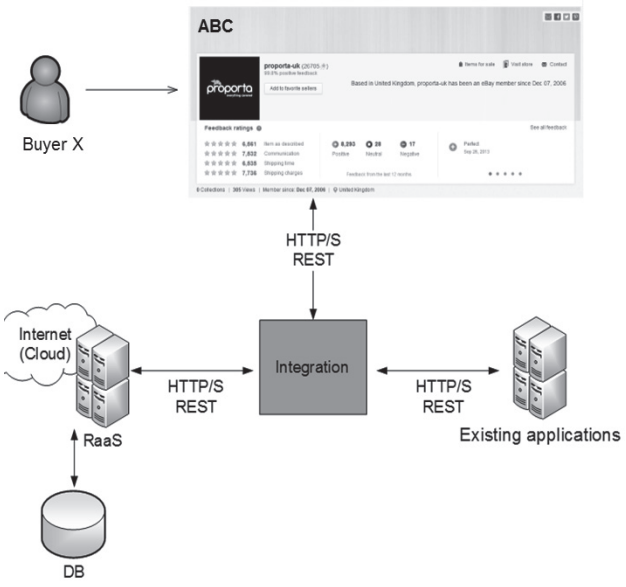

Figure 5: RaaS Communication

In the background, for all transactions processed, ratings are sent to the RaaS to enable the calculation of a reputation score of meaning. Reputation scores are stored with their product profiles. Rater reputation is maintained in the same manner to support the altruistic incentives configuration.

The RaaS thus gives organization a competitive edge as they do not experience a delay in time to market for their new online store, and at the same time they have a sophisticated reputation system to support their online store.

4.4 RaaS framework challenges

There are many challenges that stem from the creation of a RaaS framework:

- Organizations may be challenged by the fact that their data is housed externally. Even though SaaS providers have better security than many of their clients, the transfer of control over corporate data is a difficult decision.
- Configuring reputation computation and behavior is complex. As workflows allow automation of processes involving human and machine-based activities it may be important to apply it in this context.
- Each tenant has specific needs with respect to their data requirements. To address this, a template for storing data can be provided that meets most requirements, with options to add fields to tables.
- As tenants of the RaaS component have a large variety of users, and the responsibility for creating individual accounts for end users, and granting access to resources lies with the tenant, a well-developed access control component should be provided.
- The management of raters and other identities is complex. In most cases, user accounts are managed and stored independently by each tenant and authentication occurs within the organizational boundary. This means that the identity of the user, with any relevant credentials is sent to the RaaS to allow identification and access control. Different types of identities and credentials need to be managed.
- A key factor for SaaS is availability. For mission-critical applications, network availability is a dangerous point of failure.

5. CONCLUSION

The main aim of this paper was to identify requirements for a RaaS framework. Here, this has been done only at a theoretical level. Although a comprehensive set of requirements have been identified, more research and analysis is still needed. To the best of our knowledge, this is the first attempt at identifying requirements for such a framework.

The example scenario indicates that it is plausible to create a configurable reputation system to accommodate a variety of online communities. This research focussed on elements that exist in reputation systems and how these elements can be configured. Further, RaaS framework requirements were identified by considering SaaS application requirements, user requirements and reputation framework requirements. RaaS exposes services by utilizing the proposed architecture.

Future work will address the definition of more detailed configuration features and how they affect reputation scores in different scenarios. A RaaS prototype is to be defined for which simulations are to be performed to experiment with the different configurable components to identify which elements are more appropriate to configure.

6. ACKNOWLEDGEMENT

The support of SAP P&I BIT Mobile Empowerment and the National Research Foundation (NRF) under Grant number 81412 and 81201 towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the authors and not necessarily to be attributed to the companies mentioned in this acknowledgement.

REFERENCES

[1] Biz Community, "Mastercard worldwide reveals online shopping survey," 2012. [Online]. Available: http://www.bizcommunity.com/Article/196/168/73927.html

[2] S. Mulpuru, C. Johnson, and D. Roberge, "US online retail forecast, 2012 to 2017," 2013. [Online]. Available: http://www.forrester.com/US+Online+Retail+Forecast+2012+To+2017/fulltext/-/E-RES93281?objectid=RES93281

[3] Y. Wang and J. Vassileva, "Toward trust and reputation based web service selection: A survey," *International Transactions on Systems Science and Applications (ITSSA) Journal*, vol. 3, no. 2, pp. 1–38, 2007.

[4] F. Braithwaite and M. Woodman, "Success dimensions in selecting cloud software services," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2011.

[5] H. Li, "A configurable online reputation aggregation system," Ph.D. dissertation, University of Ottawa, 2007.

[6] D. Gambetta, *Trust Making*. Oxford, 1990, ch. Can We Trust Trust?

[7] T. D. Huynh, "A personalized framework for trust assessment," in *SAC' 09*, 2009.

[8] A. Josang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[9] D. Fahrenholtz and W. Lamersdorf, "Transactional security for a distributed reputation management system," in *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web)*, 2002.

[10] M. Vinkovits, "Towards requirements for trust management," in *Tenth Annual International Conference on Privacy, Security and Trust*, 2012.

[11] F. G. Marmol and G. M. Perez, "Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems," *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 185–196, 2010.

[12] R. Alnemr, M. Schnjakin, and C. Meinel, "Towards context-aware service-oriented semantic reputation framework," in *International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11*, 2011.

[13] H. Liao, "Design of SaaS-based software architecture," in *International Conference on New Trends in Information and Service Science*, 2009.

[14] X. Jiang, Y. Zhang, and S. Liu, "A well-designed SaaS application platform based on model-driven approach," in *Ninth International Conference on Grid and Cloud Computing*, 2010.

[15] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[16] json.org, "Introducing JSON." [Online]. Available: http://www.json.org/

[17] World Wide Web Consortium, "Extensible markup language (XML)." [Online]. Available: http://www.w3.org/XML/

[18] J. Espadas, D. Concha, and A. Molina, "Application development over Software-as-a-Service platforms," in *The Third International Conference on Software Engineering Advances*, 2008.

[19] W. Sun, X. Zhang, C. J. Guo, P. Sun, and H. Su, "Software as a service: Configuration and customization perspectives," in *IEEE Congress on Services Part II*, 2008.

[20] F. Marmol and M. Perez, "TRMSim-WSN, trust and reputation models simulator for wireless sensor networks," *IEEE*, 2009.

[21] Z. Noorian and M. Ulieru, "The state of the art in trust and reputation systems: A framework for comparison," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 5, no. 2, pp. 97–117, 2010.

[22] G. Swamynathan, "Reputation management in decentralized networks." [Online]. Available: http://www.powershow.com/view/11e08-ndu1z/reputation_management_in_decentralized_networks_powerpoint_ppt_presentation

[23] F. R. Farmer and B. Glass, *Web Reputation Systems*. O'Reilly, 2010.

[24] L. Liu and M. Munro, "Systematic analysis of centralized online reputation system," *Decision support systems*, vol. 52, no. 2, pp. 438–449, 2012.

[25] R. Jurca, "Truthful reputation mechanisms for online systems," Ph.D. dissertation, 2007.

[26] Z. Liang and W. Shi, *Collaborative Computing: Networking, Applications and Worksharing*, 2005, ch. Performance Evaluation of Rating Aggregation Algorithms in Reputation Systems.

NOTES