Personalized Composition of Trustful Reputation Systems

Johannes Sänger(✉), Christian Richthammer, André Kremser, and Günther Pernul

Department of Information Systems, University of Regensburg, Regensburg, Germany
{Johannes.sanger,Christian.richthammer,Andre.kremser, Gunther.pernul}@wiwi.uni-regensburg.de

Abstract. The vast amount of computation techniques for reputation systems proposed in the past has resulted in a need for a global online trust repository with reusable components. In order to increase the practical usability of such a repository, we propose a software framework that supports the user in selecting appropriate components and automatically combines them to a fully functional computation engine. On the one hand, this lets developers experiment with different concepts and move away from one single static computation engine. On the other hand, our software framework also enables an explorative trust evaluation through user interaction. In this way, we notably increase the transparency of reputation systems. To demonstrate the practical applicability of our proposal, we present realistic use cases and describe how it would be employed in these scenarios.

Keywords: Trust management · Reputation systems · Reusability · Component repository

1 Introduction

New environments such as eCommerce platforms and content communities offer manifold opportunities but also pose many challenges. Unlike in traditional settings, electronic interaction partners are usually strangers whose trustworthiness is unknown. To cope with this, trust and reputation models have emerged in recent decades. They allow to rate a set of objects (e.g. actors, products) and calculate a reputation value based on the feedback given. Since Resnick et al.'s paper [1] on the use of reputation systems to facilitate trust in Internet transactions, a vast number of reputation systems has been proposed. One major problem in this context is that most of them are completely designed from scratch and that established ideas are rarely considered [2]. To address this, Sänger and Pernul [3] decompose common reputation systems into single functional building blocks and describe and implement them in a publicly available repository.

One important factor that can be observed in connection with the repository is the subjective nature of trust, meaning that the trust value regarding one

P. Samarati (Ed.): DBSec 2015, LNCS 9149, pp. 207–214, 2015.
DOI: 10.1007/978-3-319-20810-7_13

entity might be different from the view of various end users. Many trust models try to cover this by involving individualized information on the current end user in the computation process. However, while the output value is based on distinct input data, the computation methods applied are alike for every user in such systems. In this work, we choose a different path. We take existing concepts and let the user make up his "new" personalized trust model. Thereto, we develop a software framework that allows to combine the reusable components provided in the aforementioned repository [3]. The benefits of such an approach are manifold. From the end user's point of view, we allow the dynamic adaption of a reputation system to a specific situation, enable an explorative trust evaluation through user interaction, and notably increase the transparency of reputation systems as the user himself composes the computation methods. Considering the increasing number and sophistication of attacks on reputation systems [4], these benefits constitute important aspects to raise situational awareness and thus to foster trust in reputation systems. We argue that being able to actively add and remove particularly designed computation components greatly facilitates the discovery of manipulations such as multiple referrals between colluding entities.

The remainder of the paper is based on the design science research paradigm including the guidelines for conducting design science research by Hevner et al. [5]. In particular, we follow the design science research methodology introduced by Peffers et al. [6]. In Sect. 2, we delineate the scientific background and related work important with respect to this work. Thereby, we outline our research goals. After that, we introduce our concept for flexible and dynamic trust model composition. We expose the conceptual design in Sect. 3 and describe how it is implemented in Sect. 4. Subsequently, we demonstrate the proper functioning of our software framework and discuss the benefits of our approach in Sect. 5. Finally, we sum up our contribution and conclude in Sect. 6.

2 Background and Related Work

In this section, we discuss previous work that forms the basis for this paper. The information presented directly leads us to our research goals.

2.1 Reusability for Trust and Reputation Systems

The ongoing interest in trust and reputation systems has resulted in a large number of different proposals. The computation methods applied to come up with an ultimate trust value making a statement about the trustworthiness of a particular entity range from simple arithmetic via statistical approaches through to graph-based models. Moreover, they involve multiple factors such as context information, propagation, and personal preferences.

Since most of the reputation systems proposed in literature use computation methods that are entirely built from scratch [2], well-established approaches are rarely considered and promising concepts get lost in the shuffle. In order to foster reusability of the particular components of reputation systems, Sänger and Pernul [3] propose a hierarchical component taxonomy of computation engines.

The taxonomy forms the basis for setting up a repository containing design knowledge both on a conceptual and an implementation level[1]. On the conceptual level, the components are described in design pattern-like artifacts. On implementation level, all components are provided in form of reusable web services. This repository, in turn, is supposed to serve as a natural framework for the design of new reputation systems.

2.2 Research Gap

Compared to [3], we want to go one step further. The selection and interpretation of adequate components for new reputation systems in a specific application area requires time, effort, and to some extent knowledge of the area. Therefore, we argue that there is a need for an application that supports this development process by enabling the software-supported selection and composition of components. On the one hand, this further supports the objectives of fostering the reuse of existing ideas, encouraging researchers and platform operators to focus on the design of single components, and allowing them to experiment with different concepts. On the other hand, the software-supported selection and composition of components also gives platform operators the ability to dynamically combine particular functional blocks and move away from one single static computation engine. Furthermore, we now take the end users of reputation systems as another group of stakeholders into consideration. With the help of an application for the dynamic composition of reputation-based trust models, we want to enable an explorative trust evaluation through user interaction. Since end users themselves can compose the computation methods, such an application notably increases the transparency of reputation systems. Moreover, end users are able to dynamically adapt a computation engine to the specific situation and to their own needs.

3 Conceptual Design

In this section, we describe the conceptual design for a software framework, where single components of a reputation system (implemented as web services) can be initially selected by the user, are gradually composed to a fully functional reputation system and where the selection can finally be flexibly and dynamically refined. At each step, our system supports the user by filtering further possibilities based on composition rules, input data and previous user decisions.

Figure 1 depicts the overall process of our component-based reputation system with user interaction. The phases "selection" and "composition" are iteratively run through so that a user can successively adapt and refine the selection of components. In this way, he can experiment with different compositions, exploratively evaluate reputation and approach to the most suitable reputation system. To develop the conceptual design of our framework, we carry out the four major steps described in the following.

[1] Available online: http://trust.bayforsec.de/.

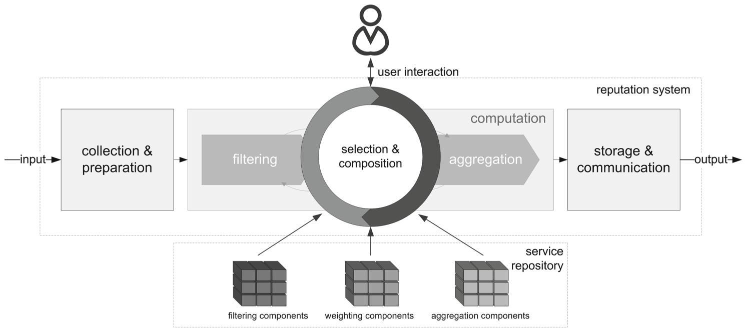

Fig. 1. Component-based reputation system with user interaction

3.1 Standard Input Format

As the provision of suitable input data is crucial for the proper functioning of a component (web service) and as the single components can be dynamically exchanged, a standard input format needs to be defined. Based on this standardized structure, we adapt each component to implement their interface accordingly. Consequently, the framework can only be used if input data is provided according to this standard.

3.2 Matching Input Data and Components

Despite the input being provided in a standardized format, it is not ensured that it contains all data necessary for the proper functioning of each component. Time-based filtering, for instance, is only applicable if referrals contain a timestamp. Therefore, the next step that needs to be considered is the analysis of necessary input data for each component. Thus, based on the requirements of each service in the service repository, we firstly set up a global list of data necessary for the proper functioning of all components. Secondly, we match every computation component to the standard input data on the list to reveal the type of input data required by each service. Applying these information, our software framework can ensure that the user can only select services compatible to the input data provided.

3.3 Interactive Component Selection

The interactive component selection enables the user to choose between the different components via a graphical user interface (GUI). Based on the selections made, a workflow description is generated. Using this workflow description, it is no longer necessary to call each component (web service) individually. Instead, the input data and the workflow description serve as input for the automated component composition service that delivers the final reputation value as output.

3.4 Automated Component Composition Service

After the selection has been made, the framework needs to automatically compose the selected components to a fully functional reputation system. Thereto, composition rules need to be defined. Filtering services, for instance, can only be executed in sequence whereas weighting services can be processed in parallel. To guarantee the correct execution of these rules, a workflow handler and a workflow engine need to be implemented. The workflow handler validates the syntax and semantics of the submitted workflow description. It then creates an execution list that is handed over to the workflow engines. The workflow engine gradually processes the execution list. It calls each computation component with its settings that are specified in the task. Having processed the complete execution list, the reputation value result is transmitted back to the workflow handler which returns the result to the user.

4 Implementation

To demonstrate the feasibility of our conceptual design, we implemented the dynamic composition framework in a software prototype. In order to allow the dynamic addition of computation components without altering the web service, we set up a plug-in orientated framework. In this way, the computation components can be developed independently and are made available to the framework by configuration. Thereby, they can be used either with an internal API or via REST-API. The interactive component selection is implemented in a web-based application using the current web standards HTML5, JavaScript and CSS3. These enable the user to select computation components in order to generate a workflow description in JSON format via the browser. The automated component composition service is implemented in PHP. Here, it is only necessary to provide the workflow description along with the referrals as input data. The automated component composition service processes the input and delivers a reputation value as output.

5 Evaluation

To rigorously demonstrate the proper functioning and goodness of our solution, we carry out a descriptive scenario-based evaluation in which we demonstrate how the framework could be used to build a reputation engine in practice. Here, we present a realistic scenario from the viewpoint of two potential user groups – system developers and end users. According to Hevner et al. [5], the evaluation procedure employed by us is a standard approach for innovative artifacts like ours.

5.1 Scenario Analysis, Part 1: System Developer

The fictitious web developer Arthur Dent runs an eBay-like electronic marketplace platform. Similar to eBay's reputation system, his current system calculates the seller reputation value based on the sum of positive (+ 1) and negative (- 1) ratings. Furthermore, it provides the share of positive ratings during the last 12 months.

As his current reputation system is quite vulnerable against a variety of attacks, he decides to integrate a new computation engine in order to enhance the security of his platform. Analyzing his environment, he comes up with the following list of requirements:

1. The reputation value should directly reflect changes in seller behavior.
2. Ratings pushing one's reputation by means of fake transactions between two friends should not be considered.
3. The reputation value should reflect the seller's behavior for a specific product group to deter dishonest sellers from building high reputation in one product group while cheating on other products.

As Arthur does not really know how to implement such an engine, he makes use of our dynamic composition framework. To decide which components best fit his requirements, he firstly reads the functionality description of each component in the knowledge repository[2]. Based on this, he experiments with different compositions and finally chooses a combination of the four components "weighting:TimeDiscountingAbsolute" (requirement 1), "filter:MultipleReferrals" (requirement 2), "weighting:ContextDiscountingAbsolutCongruence" (requirement 3) and "aggregation:SharePositive". Having selected the components on the user interface, a workflow description is generated by the framework as shown on Listing 1.

Listing 1. Workflow generated for the four selected components

```
‘‘MultipleReferrals’’->
(
‘‘TimeDiscountingAbsolute’’:[{‘‘time_limit’’:365}] |
‘‘CongruenceAbsolute’’:[{‘‘context-text’’:
        [‘‘product_category_1’’,‘‘product_category_2’’]
        }]) ->
‘‘SharePositive’’
```

Using this workflow description along with the referrals, Arthur only needs to call the automated component composition service, and will receive a reputation value that is calculated using the specified components.

This first part of the scenario demonstrates that a system developer can easily arrange a new computation engine that perfectly fits his requirements without any need to implement the logic. In this way, he is encouraged to reuse existing ideas and build on findings made by others. Besides the systems developers, the second user group that can profit from our framework are the end users.

5.2 Scenario Analysis, Part 2: End User

Having started to integrate the new static reputation engine in the platform, Arthur has an idea. Why not let the user compose his "own" reputation engine?

[2] http://trust.bayforsec.de/ngot/index.php.

To accomplish this, he pre-defines the available input data and integrates the resulting component selection form in his platform.

To evaluate if the dynamic reputation engine actually fulfills his requirements and to see how seller reputation can be exploratively analyzed from a user's point of view, he creates a fictional profile of a malicious seller on his platform who carried out 20 transactions (one per day, starting on 2015-01-01). For the first 10 transactions, he behaved honestly to build a high reputation. He thereby sold trading cards. Subsequently, he started to cheat on transactions involving mobile phones for 5 times resulting in negative ratings. To recover his reputation, he finally asked a friend to rate him positively after 5 fake transactions attributed to mobile phones as well.

Having set up the seller profile, Arthur gradually composes the reputation engine designed in the previous part. Figure 2 depicts the resulting reputation values. The four lines on the chart reflect the different composition phases.

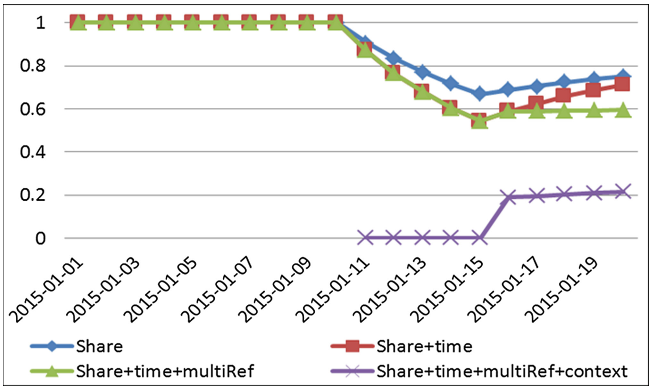

Fig. 2. Reputation values calculated over time

The first engine, which only consists of the aggregation component, provides a value of 1 (100 Nevertheless, the multiple positive ratings provided by the same buyer are still equally considered. Adding the MultipleReferrals filter this problem is sufficiently addressed. Note that the user can only become aware of the fact that the seller has been rated unfairly high by one buyer, if he actively filters these multiple referrals. Whereas initially presenting the lower reputation value may indeed reflect the real seller reputation, but still hide information about the multiple unfair ratings. Finally, Arthur extends the engine by the context-based weighting component. The corresponding values on the chart depict the reputation values calculated for the context-description "phone" and "mobile". As there is no evidence for this context at the beginning, the first value is provided for the 2015-01-11. The reputation values involving context stay considerably low, since the seller did not perform well in this context for all referrals considered.

6 Discussion and Conclusion

Overall, this scenario elucidates that our dynamic composition framework has an obvious utility from a practical point of view for system developers and end users. Developers, on the one hand, can experiment with different combinations, easily build reputation computation engines and even let the end user decide which combination to apply without having to implement various systems. End users, on the other hand, can add and remove single components, exploratively analyze seller reputation, create a personalized computation engine and in this way strengthen their situational awareness. All in all, we reduce efforts, allow to increase the robustness of reputation systems by extending their capabilities, and enhance the transparency of reputation systems as well as the situational awareness through involving the user in the decision process.

Acknowledgement. The research leading to these results was supported by the "Bavarian State Ministry of Education, Science and the Arts" as part of the FORSEC research association.

References

1. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Commun. ACM **43**(12), 45–48 (2000)
2. Tavakolifard, M., Almeroth, K.C.: A taxonomy to express open challenges in trust and reputation systems. J. Commun. **7**(7), 538–551 (2012)
3. Sänger, J., Pernul, G.: Reusability for trust and reputation systems. In: Zhou, J., Gal-Oz, N., Zhang, J., Gudes, E. (eds.) Trust Management VIII. IFIP AICT, vol. 430, pp. 28–43. Springer, Heidelberg (2014)
4. Hoffman, K., Zage, D., Nita-Rotaru, C.: A survey of attack and defense techniques for reputation systems. ACM Comput. Surv. **42**(1), 1–31 (2009)
5. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
6. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manage. Inf. Syst. **24**(3), 45–77 (2007)