Universität
Augsburg
University

Reference Architectures for Trustworthy Energy Management and Desktop Grid Computing Applications

Gerrit Anders[1], Lukas Klejnowski[2], Jan-Philipp Steghöfer[1], Florian Siefert[1], Wolfgang Reif[1]

[1] Institut für Software & Systems Engineering
Universität Augsburg
[2] Institut für Systems Engineering – SRA
Leibniz Universität Hannover

Report 2011-11 March 2011

Institut für Informatik
D-86135 Augsburg

Reference Architectures for Trustworthy Energy Management and Desktop Grid Computing Applications

Gerrit Anders, Jan-Philipp Steghöfer, Florian Siefert, Wolfgang Reif
Institute for Software & Systems Engineering
Augsburg University, Germany
E-Mail: {anders, steghoefer, siefert, reif}@informatik.uni-augsburg.de

Lukas Klejnowski
Institut für Systems Engineering – SRA
Leibniz Universität Hannover, Germany
E-Mail: klejnowski@sra.uni-hannover.de

Abstract

This report presents two reference architectures that can be used as architectural blueprints for applications of two different system classes. The first system class comprises applications in the field of energy management and the second one contains applications in the domain of desktop grid computing. Because applications in the scope of energy management are safety-critical and desktop grid computing applications have to cope with a variety of self-interested participants, applications of these domains have in common that they can increase their robustness and efficiency by considering the trustworthiness of participants. Therefore, the reference architectures given here are based on a middleware that provides functionality for the utilization of trust: experiences with participants can be stored and evaluated so that trust can be derived and incorporated into the applications.

1 Introduction

Reference architectures are a tool to facilitate the construction of applications of a specific class by guiding the development of an appropriate architecture. In this report, we present two reference architectures, each modeling the abstract concepts of a specific system class as well as their interdependencies. The first one serves as a template for architectures for trustworthy applications in the field of energy management, whereas the second one defines the basic concepts of the system class of desktop grid computing applications.

A characteristic of applications of these system classes is that they are open, heterogeneous systems that have either to ensure safety or to provide high performance in spite of untrustworthy, self-interested participants. Our approach to deal with this situation is to make use of the knowledge of the trustworthiness of participants in order to increase the systems' robustness and efficiency. Trust, as we understand it, is a multi-faceted concept consisting of the facets *functional correctness*, *safety*, *security*, *reliability*, *credibility*, and *usability* [8]. For example, we define *reliability* as the quality of a system with respect to its availability under disturbances or partial failure and *credibility* as the belief that a participant interacts in a desirable manner. Important properties of trust are that it is of temporary nature and that it allows to measure a participant's confidence in its interaction partners by evaluating experiences made with these participants in previous interactions, and thus enables cooperation in systems with various participants.

Both architectures have in common that they are composed of three layers: the lowest layer shows relevant concepts of a target platform in the form of a middleware that runs services, handles communication, and provides an infrastructure for the utilization of trust mechanisms. The middle layer presents system-class-specific concepts and their interdependencies founded on the concepts defined by the target platform. Together, these two layers form the reference architecture as a platform-specific model that can host multiple applications simultaneously. On top, the application layer specifies the architecture of one or more applications based on a given reference architecture.

In Sect. 2, we introduce the middleware that is used as the lowest layer in both reference architectures. Subsequently, we propose the reference architecture for trustworthy energy management systems in Sect. 3. The architecture for trustworthy desktop grid computing applications is presented in Sect. 4. Sect. 5 concludes this report.

2 A Trust-Enabling Middleware

Both presented reference architectures are based on the *Trust-Enabling Middleware* (TEM) [5]. It provides common concepts and functionality that are useful to both reference architectures and their respective applications. For this reason, the lowest layer of the reference architectures is denoted as *TEM layer* in the following.

The TEM is based on the message- and service-oriented middleware OCμ [9] that runs services on nodes that are physical devices with computational capabilities, such as personal computers, handhelds, or sensor nodes. These services have in common that they exhibit reactive behavior, i.e., they only act in re-

sponse to a situation. Based on that, the TEM extends OCμ services by the concept of *Periodic Services*, that allow to run software agents that feature anticipatory, proactive behavior and thus can take control of a situation. To communicate, both types of services can exchange so-called *Event Messages*; message transport is handled by OCμ.

Additionally, services can register with a so-called *Node Availability Service*, which monitors the availability of nodes by sending heartbeat messages and informs registered services in case a node goes off-line. This enables fast reaction in case of a malfunction.

The TEM further provides an infrastructure with basic functionality for determining the trustworthiness of a node or a service. This infrastructure is called *Trust Metric Infrastructure*. For the purpose of obtaining trust values, an application can store experiences in a database managed by the TEM. By using application-specific metrics that transform and interpret stored experiences, a service can evaluate its direct trust in or the reputation of a node or service. Moreover, the TEM evaluates the reliability of nodes and services with regard to the availability of nodes by analyzing communication between services [4].

Because of these functionalities, the TEM is an ideal basis for the construction of trustworthy multi-agent systems. In the following, we present two architectures based on this middleware. For more information on the TEM and the Trust Metric Infrastructure, we refer to [5].

3 An Architecture for Trustworthy Energy Management Applications

In this section, we present a reference architecture that defines basic concepts for a system class in the domain of energy production and consumption. More precisely, it serves as a template for the system class of applications that are based on power consumers, power producers, and a power grid to which producers and consumers are connected. We call this reference architecture the *Trusted Energy Grid*.

Applications in the field of energy production and consumption have several properties in common: firstly, they are usually large-scale systems since there are many participants. Secondly, they are safety- and mission-critical systems due to the fact that a failure within the system could injure people or cause harm to its physical infrastructure. Thirdly, they are heterogeneous, open systems since participating components are made by different manufacturers and their behavior is hard to predict. The Trusted Energy Grid, as a reference architecture for various kinds of applications, features and has to deal with these properties, too. For example, applications on top of the Trusted Energy Grid could coordinate and manage the electrical storage provided by the batteries of electric vehicles, as investigated in [2], or group controllable consumers in order to sell load reduction to electricity suppliers [6]. Furthermore, as shown in Sect. 3.3, another application's task might be to maintain safety by permanently balancing energy consumption and production in the face of a vastly increasing number of unpredictable and less controllable participants.

To be able to cope with uncertainty in power networks, the Trusted Energy Grid must provide tools to regard trust of participants and of information they

make available to the system. Since the TEM provides trust mechanisms and functionality out of the box, we present the Trusted Energy Grid as a platform-specific model based on this middleware.

In the following, we show the characteristic concepts of the system class described by the Trusted Energy Grid (see Sect. 3.1) as well as the concepts of the TEM that are useful in this context (see Sect. 3.2). Subsequently, we give an example of an application that utilizes the concepts introduced by this reference architecture (see Sect. 3.3).

3.1 Trusted Energy Grid: System-Class-Specific Concepts

As a reference architecture for energy management applications, the Trusted Energy Grid's main concepts are power consumers (`GenericPowerConsumer`), power plants (`GenericPowerPlant`), and the power grid (`PowerGrid`), see Figure 1. The power grid connects power plants and consumers. It is modeled by an entity called `TransmissionAgent` that is responsible for determining the power line frequency by comparing energy production and consumption, and further can be refined to take transmission-specific details of the grid into account (e.g., the energy lost in long distance transmission). A power consumer can be any kind of energy consuming entity such as an electrical device, a household, a large or industrial consumer, or an entire supply area. Each consumer has a load curve (`LoadCurve`) that shows a consumer's energy demand over time. Generic power plants feed energy into the power grid. They are divided into deterministic power plants (`DeterministicPowerPlant`) and stochastic power plants (`StochasticPowerPlant`). Deterministic power plants are power plants whose output can be determined and scheduled in advance as is the case with nuclear, hydro, or coal power plants. The output of a deterministic power plant is adjusted according to a schedule (`Schedule`) that states how much power should be generated at which point in time. In contrast, stochastic power plants are power plants whose output is rather unpredictable since it depends on consumer attitude or natural phenomena. Examples are domestic combined heat and power units or weather-dependent power plants (`WeatherDependentPowerPlant`), such as solar power plants and wind turbines.

Weather stations (`WeatherStation`) monitor the current weather conditions (`WeatherCondition`), such as solar radiation and wind speed, and create weather forecasts (`WeatherForecast`) based on these measurements. Since the output of weather-dependent producers depends on current weather conditions, weather forecasts can be used to predict their future output and, for example, to adjust schedules of deterministic power plants accordingly.

Power plants, consumers, the power grid, as well as weather stations and conditions are embedded in an environment (`Environment`) that is equipped with a coordinate system and, therefore, feature a geographical location.

Additionally, both power plants and consumers are able to participate in the energy market (`EnergyMarket`) where they can sell and buy energy at auctions. Moreover, the energy market is an abstract concept that serves as an interface to enable interactions between different applications running in parallel on the Trusted Energy Grid. For realistic scenarios, the `EnergyMarket` can be implemented based on the model of the European Energy Exchange[1].

[1] http://www.eex.com/

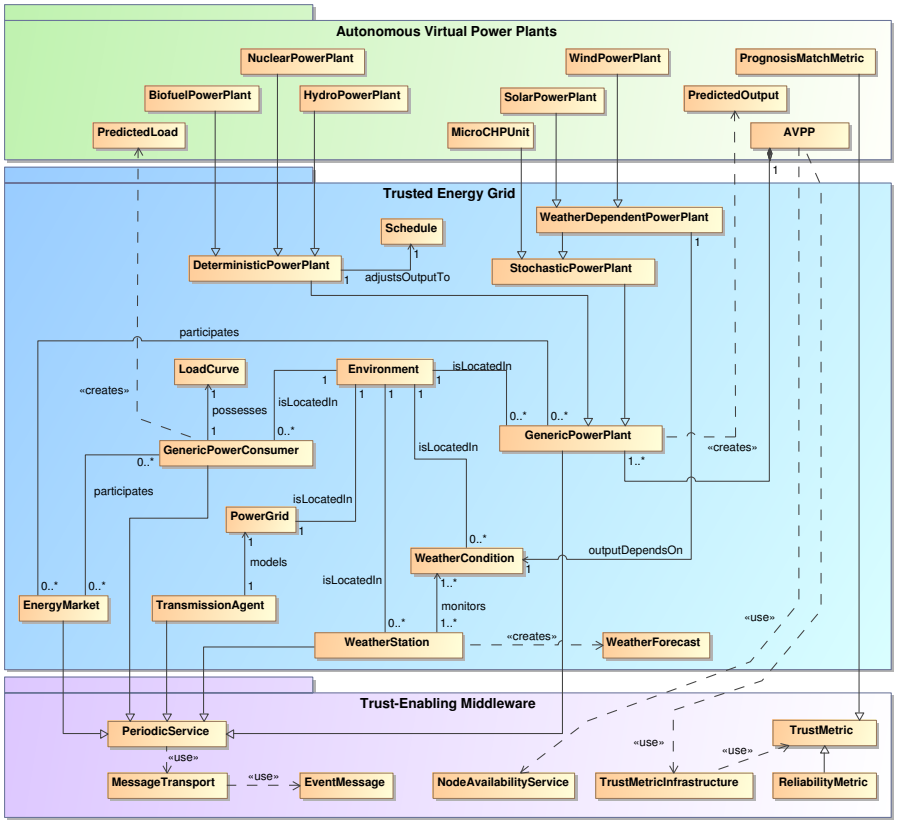

Figure 1: The Trusted Energy Grid Reference Architecture

3.2 Trusted Energy Grid: TEM Layer

In the Trusted Energy Grid, power plants, consumers, the transmission agent, weather stations, and the energy market exhibit reactive and proactive behavior. Furthermore, these concepts are heavily dependent on the capability to communicate information to other entities in the system. Consequently, each of these concepts is realized as a `PeriodicService` in the platform-specific model presented here, thus inheriting, among other things, the ability to show anticipatory, self-initiated behavior and to exchange information via messages (`EventMessage`, `MessageTransport`).

As presented in Sect. 2, the TEM estimates the reliability of nodes and services by monitoring the availability of nodes. For the Trusted Energy Grid, the knowledge about the reliability of power plants or consumers is very important because a safety-critical system should be able to take preventive measures to maintain safety and to ensure proper operation in spite of unexpectedly unavailable resources. In the Trusted Energy Grid, almost all `PeriodicServices` represent physical components whose availability should be coupled to the availability of their physical counterpart. For example, if a physical power plant goes off-line, the service that represents this power plant should also be unavailable so that its reliability decreases. However, since the TEM measures the availabil-

ity for each node and not for every single service, each `PeriodicService` that represents a physical component has to run on its own node. Hence, whenever a physical component fails, the corresponding node goes off-line.

Additionally, to store experiences from which trust can be derived, the Trusted Energy Grid makes use of the `TrustMetricInfrastructure` provided by the TEM. The transformation and interpretation of this information into a trust value is done with the help of a metric that adheres to the interface defined by an abstract trust metric (`TrustMetric`). For example, the `ReliabilityMetric` can be used to determine the reliability of services.

3.3 Trusted Energy Grid: An Exemplary Application Layer

In this section, we demonstrate an example application based on the Trusted Energy Grid that we call *Autonomous Virtual Power Plants* (AVPPs) [1].

In the power grid, one of the major challenges is to balance energy production and demand despite uncertainty introduced by a fluctuating energy demand and a steadily increasing number of uncontrollable power plants whose future output is difficult to predict and very volatile. That is because the output of such power plants depends on the availability of natural resources like wind or sunlight, or on consumer behavior as is the case with domestic combined heat and power (CHP) units. AVPPs are an approach to tackle this problematic situation. Their objective is to hold energy production and consumption in balance at all times in order to maintain safety and to guarantee proper operation of the power grid despite a tremendously increasing number of stochastic power plants.

One of the central ideas of this application is to partition the power plant landscape into several groups of power plants called, as the application itself, AVPPs. Each AVPP consist of various controllable power plants that require a schedule as well as uncontrollable ones. A major advantage of partitioning the power plant landscape by the use of AVPPs is that the complexity of creating schedules, which is an optimization problem with a large search space, is greatly reduced. More precisely, we define an AVPP (see `AVPP` in Figure 1) as a self-organizing, self-adapting, and self-optimizing ensemble of different power plants. Here, self-organization means that AVPPs autonomously find a suitable structure that supports the system's goal, i.e., balancing energy production and demand. If the structure is not suitable any more, e.g., because one or more AVPPs repeatedly cannot cope with the load situation, power plants restructure themselves into new AVPPs. Moreover, an AVPP self-adapts because it autonomously reacts to changes in energy production and consumption by dynamically adjusting schedules of controllable power plants in order to maintain the balance.

As mentioned above, AVPPs consist of different types of power plants, which extend the basic concepts provided by the reference architecture. This can be deterministic power plants, such as biofuel (`BiofuelPowerPlant`), hydro (`HydroPowerPlant`), or nuclear power plants (`NuclearPowerPlants`); but also weather-dependent producers, such as solar power plants (`SolarPowerPlant`), wind turbines or wind farms (`WindPowerPlant`), as well as other stochastic power plants like domestic CHP units (`MicroCHP`).

In order to enable proactive measures to ensure stable operation, power consumers predict their future load (`PredictedLoad`) based on information about

their former energy consumption. Moreover, power plants predict their future output (`PredictedOutput`), which depends, among other things, on the power plant's state and historic data. Predictions further depend on the weather forecast provided by weather stations in case of weather-dependent power plants, on consumer attitude in case of usage-dependent power plants, or on a deterministic power plant's schedule and performance characteristics.

To handle the uncertainty introduced by inaccurate predictions and unreliable power plants, AVPPs use additional information about the trustworthiness of power plants and consumers. The trustworthiness of a power plant is characterized by its reliability and credibility. Its reliability is measured by the TEM and thus states how often the power plant was off-line. The credibility of a power plant indicates the accuracy of its predicted power outputs. Furthermore, the trustworthiness of a consumer is its credibility in terms of the accuracy of its predicted load and is therefore defined analogously to the credibility of power plants. The information about predicted and actual load or power output is stored by making use of the TEM's `TrustMetricInfrastructure`. A prognosis match metric (`PrognosisMatchMetric`) can then be used to derive credibility from this information once actual values are available by comparing predicted values with actual ones.

The application makes use of the knowledge about the trustworthiness of power plants in several aspects. On the one hand, it is used in the course of AVPP formation. The objective of the AVPP formation is to form several AVPPs of similar quality because a single AVPP that cannot cope with a given situation could endanger the proper operation of the whole system. Therefore, the formation of AVPPs tries to increase robustness by grouping trustworthy and untrustworthy power plants together so that every AVPP exhibits almost the same mix with respect to the trustworthiness of power plants. On the other hand, an AVPP benefits from knowledge about the trustworthiness of power plants and consumers when creating schedules. That is because an AVPP creates schedules in such a way that it reduces dependence on untrustworthy, deterministic power plants by decreasing their scheduled output. This is reasonable since untrustworthy power plants can cause imbalances between energy production and consumption due to inaccurate predictions or malfunctions. Furthermore, an AVPP makes sure to hold sufficient reserve power to be able to cope with unexpected situations caused by untrustworthy power plants or consumers.

In addition, an AVPP registers with the TEM's `NodeAvailabilityService` in order to be notified when a power plant goes off-line. In such a case, the AVPP triggers the recalculation of schedules.

Having described the Trusted Energy Grid and an example based on this reference architecture, in the next section, we present a reference architecture for desktop grid computing applications.

4 An Architecture for Trustworthy Desktop Grid Computing Applications

This section introduces the *Trusted Computing Grid* reference architecture for the system class of desktop grid computing applications. These are applications

that are executed in an environment, called desktop grid system, that consists of a great number of computers that cooperatively process computationally intensive tasks produced by clients, i.e., instances of desktop grid applications. However, as we focus on desktop grid computing, these applications do not run on dedicated servers, but on user devices, such as personal computers. Desktop grid systems have in common that they are based on open, distributed systems without central control (e.g., the Internet) in which several heterogeneous clients act on behalf of users and cooperate in order to reach a goal. For this reason, these applications feature a highly dynamic structure. Furthermore, since desktop grid applications run on user devices, they have to share resources with other applications, such as other desktop grid applications.

In desktop grid systems, clients create work units that are expected to be processed by the grid. For this purpose, each client can act in the role of a *submitter* and *worker*. In the role of a submitter, a client submits work units to the grid that should be processed by another client in the role of a worker. Having processed a work unit, a worker returns the result to the submitter.

By being able to delegate work to other clients, the advantage of grid computing is that it provides a way to decrease computing time. Therefore, a user of such a grid usually expects correct results as fast as possible. However, there may be clients that plan to exploit or damage the system. For example, these are clients that do not contribute to the grid as they do not process work units, clients that return wrong results or no results at all, as well as others that flood the grid with work units. Thus, since each client may behave uncooperatively, the big challenge of desktop grid computing is to have efficient, robust systems in spite of uncertainty introduced by their clients.

As a reference architecture for desktop grid computing applications, the Trusted Computing Grid [3] provides concepts to master this challenge. Since clients work together to process work units, each client has to decide which client should process a work unit and whether or not to process a work unit on behalf of another client. Summarized, a client has to find suitable interaction partners. Furthermore, to deal with uncooperative, i.e., untrustworthy, clients, we propose the utilization of the TEM's trust mechanisms and functionality.

The next sections give an insight into the Trusted Computing Grid's system-class-specific concepts (see Sect. 4.1) as well as relevant concepts of the TEM (see Sect. 4.2). An example application is shown in Sect. 4.3.

4.1 Trusted Computing Grid: System-Class-Specific Concepts

Systems based on the Trusted Computing Grid reference architecture (see Figure 2) consist of multiple interacting agents (`TCGAgent`), each representing a client. These agents participate in the grid in the role of a worker and a submitter in order to process data. Since we regard desktop grid computing applications, the user of a Trusted Computing Grid application (`TCGUser`) activates, deactivates, configures, and constrains the `TCGAgent`. For example, the user states how many resources (`Resource`) may be allocated. As a result, a `TCGAgent` usually cannot use all resources provided by a given system.

Each agent can run multiple applications (`TCGApplication`) on the generic architecture provided by the Trusted Computing Grid, such as an application for video encoding or face recognition. Applications create jobs (`TCGJob`) that

are computationally intensive tasks that are expected to be performed by the grid. For example, a job is the task to search for specific people in a number of pictures. Having created a new job, it is split into smaller pieces, called work units (`TCGWorkUnit`), in an application-specific way by making use of an instantiation of `TCGSplitter`. Subsequently, the agent that created the job manages the distribution of work units to other agents, thus acting as a submitter. In this role, an agent is self-interested and therefore always tries to maximize the number of successfully processed work units by choosing suitable workers. We distinguish four different prototypical types of workers:

1. *Altruistic agents* that accept and process all work units submitted by trustworthy agents.
2. *Free riders* that reject all work units.
3. *Egoistic agents* that accept work units but might cancel their processing.
4. *Rational agents* that decide whether to accept or reject a work unit based one or more conditions, e.g., their trust in the submitter, or their current work load.

For choosing a suitable interaction partner, agents can evaluate the trustworthiness of other agents by making use of a trust metric (`AggregatedTrustMetric`) that aggregates an agent's credibility calculated by credibility metrics for direct trust or reputation (`CredibilityMetric`, `CredibilityReputationMetric`) as well as reliability (`ReliabilityMetric`, `ReliabilityReputationMetric`) measured by the TEM. For example, an agent's credibility can be influenced by the accuracy of its computational results. Whenever an agent receives a request for processing a work unit, it decides in its role as a worker whether the request should be accepted or rejected. Accepted work units can be processed by the corresponding application of the worker. Having finished the processing of a work unit, the worker returns the result (`TCGWorkUnitResult`) to the submitter which combines the results from all agents that processed a work unit by using an application-specific `TCGCombiner`.

4.2 Trusted Computing Grid: TEM Layer

As explained in the previous section, `TCGAgents` show anticipatory behavior and must be able to communicate. For this reason, in the platform-specific model that we present here, `TCGAgents` are modeled as `PeriodicServices` which do not only provide proactive behavior but also have the ability to communicate by exchanging messages (`EventMessage`, `MessageTransport`). In order to be notified when an agent becomes unavailable (e.g., because a user shuts down the computer), `TCGAgents` can register with the `NodeAvailabilityService` provided by the TEM.

The TEM's `TrustMetricInfrastructure` can be used by agents to store experiences with their interaction partners. To evaluate an agent's trustworthiness based on these experiences, the Trusted Computing Grid proposes to combine reliability and credibility of agents to an aggregated representation of an agent's trustworthiness. For this purpose, it utilizes the `AggregatedTrustMetric` that aggregates the results of the Trusted Computing Grid's `CredibilityMetric` and `CredibilityReputationMetric`, and that aggregates the results of the TEM's

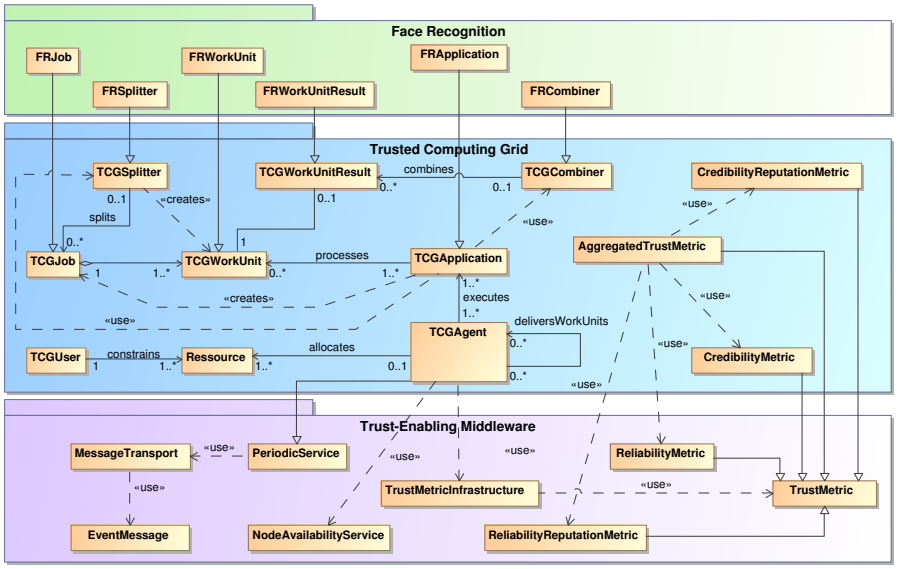

Figure 2: The Trusted Computing Grid Reference Architecture

`ReliabilityMetric` and `ReliabilityReputationMetric`. All these metrics adhere to the interface and functionality defined by the TEM's `TrustMetric`.

The following section presents an application that makes use of the concepts defined by the Trusted Computing Grid reference architecture.

4.3 Trusted Computing Grid: An Exemplary Application Layer

In this section, we present a desktop grid computing application for face recognition that adheres to and concretizes the basic concepts introduced by the Trusted Computing Grid reference architecture. In this application, a job (see `FRJob` in Figure 2) is to identify persons by using a face recognition algorithm applied to photos. A typical job contains multiple photos and description models of several faces to be recognized. Correspondingly, a typical work unit (`FRWorkUnit`) contains one or more of these photos and the description model of a specific face. The face recognition application (`FRApplication`) states how to create `FRJobs` on the one hand, and how to process `FRWorkUnits` by supplying a suitable algorithm for the task on the other hand. The `FRSplitter` splits `FRJobs` into `FRWorkUnits`. The results (`FRWorkUnitResult`) of processed `FRWorkUnits` are combined by the `FRCombiner` and returned to the application afterwards.

As stated before, applications in the field of desktop grid computing have in common that they have to deal with uncooperative clients. For this reason, and because the functionality for the distribution and acceptance of work units can be defined in a generic way, we model the `TCGAgent` as a trust-aware, adaptive agent that makes decisions with respect to the trustworthiness of other agents. Note that the Trusted Computing Grid makes metrics for determining the credibility and reliability of agents available in a generic way.

When a new work unit is ready to be processed, `TCGAgent` chooses a suitable interaction partner with regard to its direct trust in and the reputation of potential interaction partners as well as other criteria, such as current work load. For this purpose, an agent's trustworthiness is appraised with the help of the `AggregatedTrustMetric`. Once an interaction is completed, an agent gains experience with its interaction partner which is stored with the help of the `TrustMetricInfrastructure`. An interaction is completed if a work unit is rejected, processed and returned to the submitter, or if a timeout occurs. For determining whether an interaction has a positive or negative outcome, agents evaluate the interaction partner's behavior in the course of the interaction, e.g., by checking the correctness of the result or by resolving whether or not the rejection of a work unit was justified. Each experience influences the trust in an interaction partner. For example, if an agent returns faulty results, its credibility decreases.

As agents prefer trustworthy interaction partners, they create implicit organizations that build upon these trust relations. These organizations are called *Implicit Trusted Communities* [7]. A Trusted Community is a dynamic organization of agents that mutually trust each other. They are implicit because agents do not know any of the existing Trusted Communities. By choosing trustworthy interaction partners, each agent's performance as well as the system's efficiency and robustness increases. If an agent notices that no other agent is willing to cooperate, it can assume that it has been excluded from all Implicit Trusted Communities. In this case, it can adapt its strategy in order to become trustworthy and, therefore, a member again. For example, an egoistic agent could decide to become more altruistic in order to increase its trustworthiness.

5 Discussion and Conclusion

In this report, we introduced two reference architectures, called Trusted Energy Grid and Trusted Computing Grid, that can be used as a template for the construction of trustworthy applications in the field of energy management and desktop grid computing. Since applications of these system classes have to deal with uncertainty, we presented the reference architectures in the form of a platform-specific model that builds upon a target platform, called Trust-Enabling Middleware, that facilitates the utilization of trust. To this end, it provides an infrastructure that comprises trust metrics, which are concretized by applications or reference architectures, and basic trust mechanisms in a service-based middleware. Furthermore, we outlined two example applications that demonstrate the usage and instantiation of the given reference architectures and how trust enables robust and efficient operations in systems consisting of a great number of different participants with unknown behavior.

On the basis of the Trusted Energy Grid, we showed the application of Autonomous Virtual Power Plants that divide a power plant landscape into groups of power plants, and that create and adjust schedules of power plants in order to balance energy supply and load in spite of unforeseen supply and load changes at all times. For this purpose, Autonomous Virtual Power Plants extend the Trusted Energy Grid by different types of power plants, the concept of an Autonomous Virtual Power Plant itself, as well as load and output predictions. Since the objective of energy applications based on the Trusted Energy Grid

may vary from application to application and under the assumption that functionalities, such as forecasting and prediction algorithms, are vendor-specific, only physical properties are generic in the Trusted Energy Grid. Consequently, the behavior of participants has to be defined by the application.

As an example based on the Trusted Computing Grid, we proposed a grid computing application for face recognition that is performed on multiple devices in parallel. As each application based on the Trusted Computing Grid has the aim to solve a computationally intensive arithmetic problem as fast as possible, in contrast to the Trusted Energy Grid, the behavior of participants and all necessary concepts can be modeled independently from a specific application. This includes the functionality for the distribution and acceptance of work units because of uniform submitter and prototypical, heterogeneous worker behavior. Nevertheless, an application has to concretize some concepts defined by the Trusted Computing Grid, including grid jobs, work units, and the algorithms that, among other things, generate jobs and process the work units.

The examples showed that the presented reference architectures prove to be helpful concepts for the creation of systems consisting of multiple interacting participants in uncertain environments. Future work includes a reference architecture for trustworthy multi-user multi-display environments in which privacy is of utmost interest. These environments typically feature a public display and one private device per user. To ensure privacy by hiding private data from unauthorized persons, private content displayed on public displays is migrated to user devices.

Acknowledgment

This research is partly sponsored by the research unit “OC-Trust” (FOR 1085) of the German Research Foundation (DFG).

References

[1] G. Anders, F. Siefert, J.-P. Steghöfer, H. Seebach, F. Nafz, and W. Reif. Structuring and Controlling Distributed Power Sources by Autonomous Virtual Power Plants. In *Proceedings of the Power & Energy Student Summit 2010 (PESS 2010))*, pages 40–42, October 2010.

[2] B. Becker, F. Allerding, U. Reiner, M. Kahl, U. Richter, D. Pathmaperuma, H. Schmeck, and T. Leibfried. Decentralized Energy-Management to Control Smart-Home Architectures. *Architecture of Computing Systems-ARCS 2010*, pages 150–161, 2010.

[3] Y. Bernard, L. Klejnowski, J. Hähner, and C. Müller-Schloer. Towards Trust in Desktop Grid Systems. In *Cluster Computing and the Grid, IEEE International Symposium on*, pages 637–642, Los Alamitos, CA, 2010. IEEE Computer Society.

[4] R. Kiefhaber, B. Satzger, J. Schmitt, M. Roth, and T. Ungerer. The Delayed Ack Method to Measure Trust in Organic Computing Systems. In *Proceedings of the Trustworthy Self-Organizing System Workshop 2010 at the Fourth IEEE Conference on Self-Adaptive and Self-Organizing Systems*, pages 27–32. IEEE Computer Society Press, 2010.

[5] R. Kiefhaber, F. Siefert, G. Anders, T. Ungerer, and W. Reif. The Trust-Enabling Middleware: Introduction and Application. Technical Report 2011-10, Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, 2011.

[6] N. Ruiz, I. Cobelo, and J. Oyarzabal. A Direct Load Control Model for Virtual Power Plant Management. *IEEE Transactions on Power Systems*, 24(2):959–966, 2009.

[7] J.-P. Steghöfer, F. Nafz, W. Reif, Y. Bernard, L. Klejnowski, J. Hähner, and C. Müller-Schloer. Formal Specification and Analysis of Trusted Communities. In *Proceedings of the Trustworthy Self-Organizing System Workshop 2010 at the Fourth IEEE Conference on Self-Adaptive and Self-Organizing Systems*, pages 33–38. IEEE Computer Society Press, 2010.

[8] J.-P. Steghöfer, R. Kiefhaber, K. Leichtenstern, Y. Bernard, L. Klejnowski, W. Reif, T. Ungerer, E. André, J. Hähner, and C. Müller-Schloer. Trustworthy Organic Computing Systems: Challenges and Perspectives. In *Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC 2010)*. Springer, October 2010.

[9] W. Trumler. *Organic Ubiquitous Middleware*. PhD thesis, University of Augsburg, 2006.