Reference Architectures for Trustworthy Energy Management, Desktop Grid Computing Applications, and Ubiquitous Display Environments

Gerrit Anders[1], Jan-Philipp Steghöfer[1], Lukas Klejnowski[3], Michael Wissner[2], Stephan Hammer[2], Florian Siefert[1], Hella Seebach[1], Yvonne Bernard[3], Wolfgang Reif[1], Elisabeth André[2], Christian Müller-Schloer[3]

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

Report 2013-05 April 2013

Institut für Informatik
D-86135 Augsburg

Reference Architectures for Trustworthy Energy Management, Desktop Grid Computing Applications, and Ubiquitous Display Environments*

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

Michael Wissner, Stephan Hammer, Elisabeth André
Institute of Computer Science
Augsburg University, Germany
E-Mail: {wissner, hammer, andre}@informatik.uni-augsburg.de

Lukas Klejnowski, Yvonne Bernard, Christian Müller-Schloer
Institute for Systems Engineering – SRA
Leibniz Universität Hannover, Germany
E-Mail: {klejnowski, bernard, cms}@sra.uni-hannover.de

Abstract

This report presents three reference architectures that can be used as architectural blueprints for applications of three different system classes. The first system class comprises applications in the field of energy management; the second one contains applications in the domain of desktop grid computing; the third system class contains multi-user multi-display applications. 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. Multi-user multi-display applications have to assess the social relationships between its users and adapt based on trustworthiness facets such as transparency and controllability. Therefore, the reference architectures given here are based on a multi-agent system 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. Additionally, the platform provides the basis for open systems in which agents enter and leave dynamically.

*This paper is a revision and extension of [1]

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 three reference architectures, each based on a domain model of the system class they can be employed in. We understand a system class as a more narrow term than the more general "domain" and denote with it a class of systems that serve the same purpose and work under similar assumptions. The domain model captures the abstract concepts of a specific system class as well as their interdependencies. These concepts are then grouped in components and interfaces between them are defined to yield reference architectures that provide a more abstract view on the system classes.

The first reference architecture serves as a template for architectures for trustworthy applications in the field of energy management, whereas the second one can be used in the system class of desktop grid computing applications. Finally, we propose a reference architecture for multi-user multi-display applications.

A characteristic of applications of these system classes is that they are open, heterogeneous systems that either have to ensure safety or provide high performance in spite of untrustworthy, self-interested participants. Our approach to deal with such situations 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* [17]. 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. Trust thus enables cooperation in systems with various participants.

All three domain models 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 multi-agent system (MAS) that provides basic agent abstractions, handles communication, and offers 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. On top, the application layer specifies the concepts of one or more applications based on the platform's concepts. The reference architectures themselves group the concepts of the middle layer in components and suggest interfaces between them. Components are assigned to nodes that can represent physical hardware. The actual deployment of software components to hardware is, however, highly specific to the requirements of the actual system and the assignment should thus only be regarded as a suggestion.

Our notation includes the stereotypes `use` and `creates`. The former implies that a class uses another class' functionality for certain purposes without an explicit association between them. It is introduced to denote how important functions of a class are delegated to other classes. The latter indicates that a class serves as a factory for instances of another class. In the text, a `typewriter font` indicates a concept while *italic font* indicates a component.

In Sect. 2, we briefly introduce the MAS that is used as the implementation basis that is depicted in the domain models. 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 and the architecture for multi-user multi-display applications is detailed in Sect. 5. Sect. 6 concludes this report.

2 A Trust-Enabling Multi-Agent System

All presented reference architectures are based on the *Trust-Enabling Multi-Agent System* (TEMAS) [2]. It provides common concepts and functionality that are useful to all three reference architectures and the respective applications based on them. For this reason, the lowest layer of the domain models is denoted as *TEMAS layer* in the following.

The TEMAS is a MAS for open environments. While it is based on the *Trust-Enabling Middleware (TEM)* [9], which itself is based on the adaptive, organic middleware OCμ [13] that features self-x properties such as self-healing and self-optimization, it incorporates an infrastructure that provides a variety of MAS concepts. Apart from facilities for communication in local and distributed environments and a yellow pages service, it allows itself and the agents to use application-specific metrics to derive trust values for different facets from prior experiences with the *Trust Metric Infrastructure* provided by the TEM. In addition to the possibility to derive trust values, the TEM allows to store experiences in persistent data bases, query reputation values (i.e., trust values that originate from experiences of other system participants), and assess the reliability of agents and devices out of the box. In the TEMAS, agents run on *nodes*, a form of container similar to those used in peer-to-peer networks. They often represent physical devices and can host several agents or reactive services. To enable fast reaction in case of malfunctioning nodes, agents can register with a so-called *Node Availability Service*. It monitors the availability of nodes by analyzing communication between services [8] and informs registered agents in case a node goes off-line.

The TEMAS provides MAS-specific concepts the TEM does not provide on its own. These concepts are consolidated in the *MASConcepts4TEM*. The *MASConcepts4TEM* extend the TEM primarily by `Agents` and `TrustAgents`, additional trust concepts such as `TrustBasedScenarios`, and basic time concepts. With respect to the TEM, it further serves as a facade because it hides the complexity of the underlying infrastructure consisting of nodes and services and dependent interfaces from higher level applications. This results, e.g., in simpler, more common, and natural interfaces for messaging and the application of trust in MAS.

Because of these functionalities, the TEMAS is an ideal basis for the construction of trustworthy MAS. In the following, we present three architectures based on this MAS. While the TEMAS is useful as a basis for systems that require the use of trust values and features such as automatic evaluation of an agent's reliability, other MAS that provide similar functionality can be used as well. For more information on the TEMAS and the Trust Metric Infrastructure, we refer to [2].

3 An Architecture for Trustworthy Energy Management Applications

In this section, we present a domain model that defines basic concepts for a system class in the domain of energy production and consumption as well as a reference architecture based on these concepts. Both models can serve 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* (TEG).

Applications in the field of energy production and consumption have several properties in common. First, they are usually large-scale systems since there are many power consumers and producers involved. Second, 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. Third, they are heterogeneous, open systems since participating components are made by different manufacturers and their behavior is hard to predict. The TEG, as a reference architecture for various kinds of applications, has to deal with these properties, too. For example, applications on top of the TEG could coordinate and manage the electrical storage provided by the batteries of electric vehicles, as investigated in [5], or group controllable consumers in order to sell load reduction to electricity suppliers [15]. 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 intermittent, unpredictable, and uncontrollable participants.

To be able to cope with uncertainty in power networks, the TEG must provide tools to regard trust of participants and of information they make available to the system. Since the TEMAS provides trust mechanisms and functionality out of the box, we present the Trusted Energy Grid as a platform-specific model based on this MAS.

In the following, we show the characteristic concepts of the system class described by the TEG (see Sect. 3.1) as well as the concepts of the TEMAS that are useful in this context (see Sect. 3.2). Subsequently, we give an example of an application that utilizes the concepts introduced (see Sect. 3.3) before we introduce the reference architecture in Sect. 3.4.

3.1 TEG: System-Class-Specific Concepts

As a reference architecture for energy management applications, the TEG's main concepts are power consumers (`GenericPowerConsumer`) and power plants (`GenericPowerPlant`), both based on `GenericPowerProsumer` as depicted in Figure 1. They are all connected to the power grid (`PowerGrid`), which is modeled by an entity called `TransmissionAgent` that is responsible for determining the power line frequency by comparing energy production and consumption and can be specialized to, e.g., 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.

Power plants feed energy into the power grid. Usually, a differentiation between controllable (`ControllablePowerPlant`) and stochastic power plants

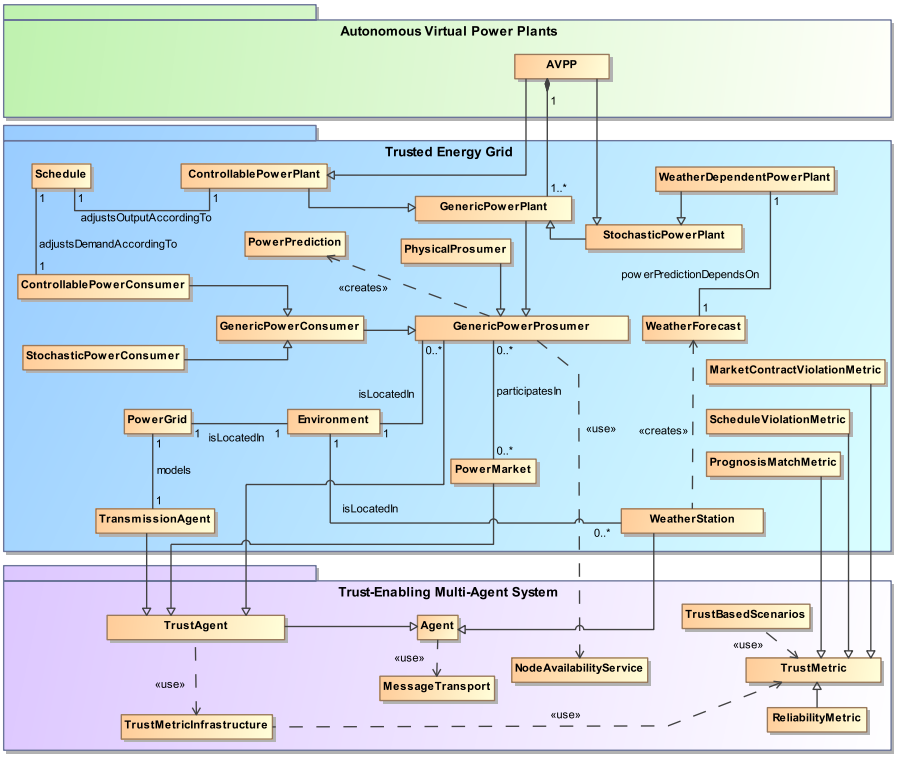

Figure 1: The Trusted Energy Grid Domain Model

(`StochasticPowerPlant`) is made. Controllable 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 controllable 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 intermittent 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.

Power consumers come in two flavors as well: `ControllablePowerConsumers` can be regulated with schedules, while `StochasticPowerConsumers` can not be controlled and request demand when needed.

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. All prosumers can make predictions of their future output (`PowerPrediction`). Since the output of weather-dependent producers depends on current weather conditions, the `PowerPrediction`'s of `WeatherDependentPowerPlants` depend on weather forecasts.

Prosumers, the power grid, as well as weather stations are embedded in an `Environment` that, among other things, provides a geographical location for each agent.

Additionally, both power plants and consumers are able to participate in the energy market (`PowerMarket`) where they can sell and buy energy at auctions. Moreover, the power market is an abstract concept that serves as an interface to enable interactions between different applications running in parallel on the TEG. For realistic scenarios, the `PowerMarket` can, e.g., be implemented based on the model of the European Energy Exchange [16].

Finally, different trust metrics are used in the system, each with its own purpose: The `PrognosisMatchMetric` allows to derive a trust value based on the quality of the power predictions of a prosumer. The deviation from the schedule of a controllable prosumer is captured by the `ScheduleViolationMetric` while the adherence to contracts made on the power market is measured by the `MarketContractViolationMetric`.

3.2 TEG: TEMAS Layer

In the TEG, prosumers, the transmission agent, weather stations, and the power 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 an `Agent` 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 (`MessageTransport`).

As presented in Sect. 2, the TEMAS estimates the reliability of nodes and services by monitoring the availability of nodes. For the TEG, the knowledge about the reliability of prosumers 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 TEG, almost all `Agent`s 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 agent that represents this power plant should also be unavailable so that its reliability decreases. However, since the TEMAS currently measures the availability for each node and not for every single agent, each `Agent` that represents a physical component should run on its own node[1]. Hence, whenever a physical component fails, the corresponding node goes off-line. Such a distribution is depicted in the component diagram in Figure 2.

Additionally, to store experiences from which trust can be derived, the TEG makes use of the `TrustMetricInfrastructure` provided by the TEMAS. The transformation and interpretation of this information into a trust value is done with the help of application-specific metrics that adhere to the interface defined by an abstract trust metric (`TrustMetric`). For example, the `ReliabilityMetric` can be used to determine the reliability of services. To enable proactive scheduling, trust-based scenarios [3] (`TrustBasedScenarios`) that use the values derived by the trust metrics to calculate different possible futures of the system and their respective probabilities can also be employed. All `Agent`s that, in one way or another, use the `TrustMetricInfrastructure` are specializations of `TrustAgent`.

[1] The TEMAS will support monitoring the availability of agents and services independent of the availability of underlying nodes in future releases.

3.3 TEG: An Exemplary Application Layer

In this section, we demonstrate an example application based on the TEG that we call *Autonomous Virtual Power Plants* (AVPPs) [4].

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, intermittent 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 with 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, hydro, or nuclear power plants; but also weather-dependent producers, such as solar power plants, wind turbines or wind farms, as well as other stochastic power plants like domestic CHP units. However, these types of power plants are not specific to the application and are therefore not depicted in Figure 1.

In order to enable proactive measures to ensure stable operation, prosumers predict their future load or consumption (`PowerPrediction`). For consumption, this data is based on information about their former energy consumption. A producer's prediction 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 TEMAS 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 TEMAS'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, in order to be notified when a power plant goes off-line, an AVPP registers with the TEMAS's `NodeAvailabilityService`. In such a case, the AVPP triggers the recalculation of schedules.

3.4 TEG: Components and Interactions

The classes of the TEG domain model can be grouped into several components as shown in Figure 2. This modularization fosters separation of concern as well as high cohesion in the independent parts of the system. Each component can be exchanged with one that offers similar functionality as long as the interface is implemented. In applications based on the TEG, different scheduling mechanisms or prediction algorithms can, e.g., be used and easily swapped at design time or even at runtime. Please note that the assignment of components to nodes in the diagram is merely a suggestion. Depending on the system requirements, trusted hardware located with the utilities can, e.g., house several prosumer agents that communicate with the physical prosumer over existing interfaces.

Weather stations, the power market, and the power grid control are all distinct nodes that can either be distributed or centralized. Weather stations, e.g., can be modeled as agents, running TEMAS as well and interacting with the prosumers directly. The power market will often be centralized, especially when prosumers are participating in a market such as the European Energy Exchange. When using a trust-aware market that adapts prices and market access based on the trust value of the market participants, a trust-management component is necessary.

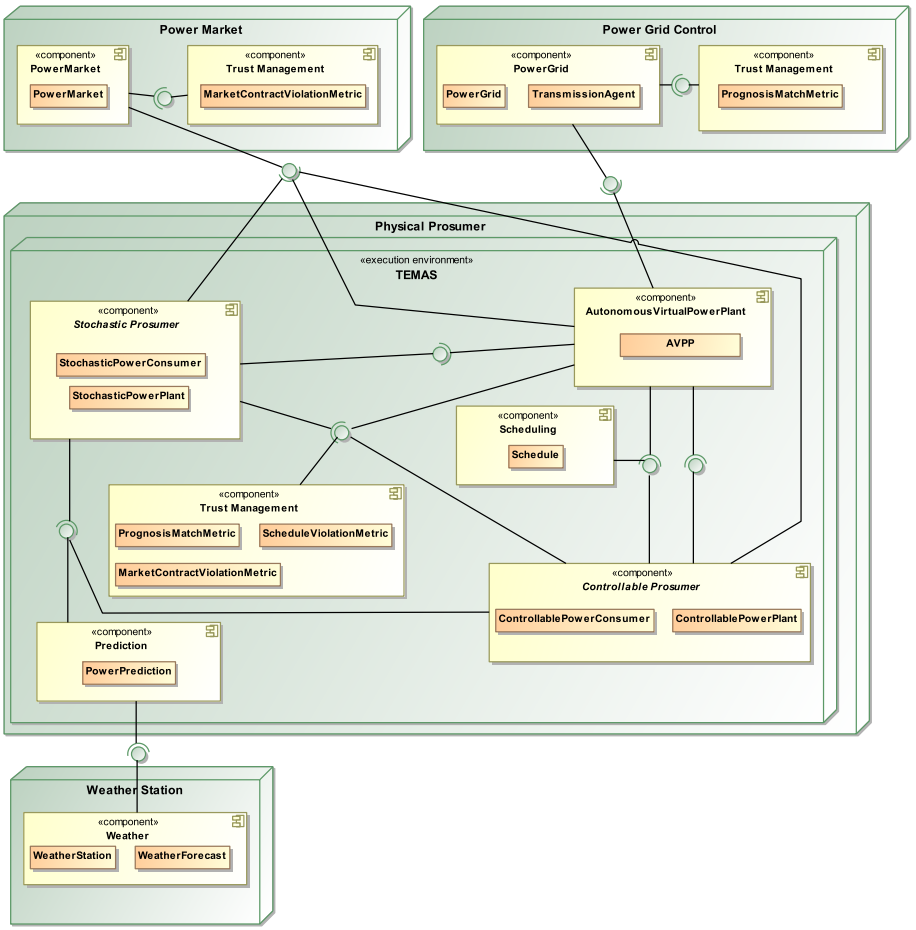

Figure 2: Components of a Trusted Energy Grid

A physical prosumer runs the TEMAS and is usually either a *StochasticProsumer* or a *ControllableProsumer*. In all cases, the agents use trust values to determine the quality of their and others' prognoses, their adherence to schedules, and their adherence to market contracts, making use of the *Trust Management* component. A *Controllable Prosumer* makes use of a *Scheduling* component that allows it to create optimized schedules according to its parameters and internal model.

To show the interaction with an actual application, Figure 2 includes an *AutonomousVirtualPowerPlant* component, modeling the application described in Section 3.3. Each physical prosumer can take on the role of an AVPP. An AVPP controls other prosumers or AVPPs and schedules them accordingly, incorporating the trust values gathered by the *Trust Management* component to adapt the schedules accordingly. An AVPP can also participate in the *PowerMarket*.

A system based on the TEG consists of a number of prosumers as well as one or more weather stations, power grid control, and, optionally, a power market. While it can not be assumed that all prosumers are based on the TEMAS in an open system, there will be interfaces that allow the interaction of prosumers.

Such interfaces will also be available for weather stations and other external elements of the system so that an execution environment is not assumed for them in Figure 2.

Having described the TEG and an example based on the concepts, 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* (TCG) 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 addition, it can not be assumed that all devices are part of a common administrative domain.

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* or of a *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 TCG [7] 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 TEMAS's trust mechanisms and functionality.

The next sections give an insight into the TCG's system-class-specific concepts (see Sect. 4.1) as well as relevant concepts of the TEMAS (see Sect. 4.2).

An example application is shown in Sect. 4.3 and the reference architecture is introduced in Sect. 4.4.

4.1 TCG: System-Class-Specific Concepts

Systems based on the TCG reference architecture (see Figure 3) 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 TCG 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 TCG, 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 its speedup by choosing suitable workers that process its work reliably and correctly.

The selection of a suitable interaction partner is delegated to an appropriate strategy. The `SubmitterStrategy` is responsible for selecting interaction partners when the agent is submitting work units in an effort to get a job done. For this purpose, it 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 TEMAS. 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, the `WorkerStrategy` decides whether the request should be accepted or rejected. This decision can also be based on trust values, e.g., the trust value of the submitter, as well as on other criteria such as the current work load. By utilizing the submitter's credibility here, an incentive mechanism is installed that enforces cooperation between the agents. Accepted work units can be processed by the workers's corresponding application with a specialization of `TCGProcessingAlgorithm`. 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`.

Both `SubmitterStrategy` and `WorkerStrategy` can be defined by the user. Additionally, they can incorporate learning mechanisms or other facilities to adapt themselves to a changing environment. This way it is possible, e.g., to let agents behave strategically in order to secure good standing with other agents before a big job has to be distributed or minimize the own efforts if no jobs are queued.

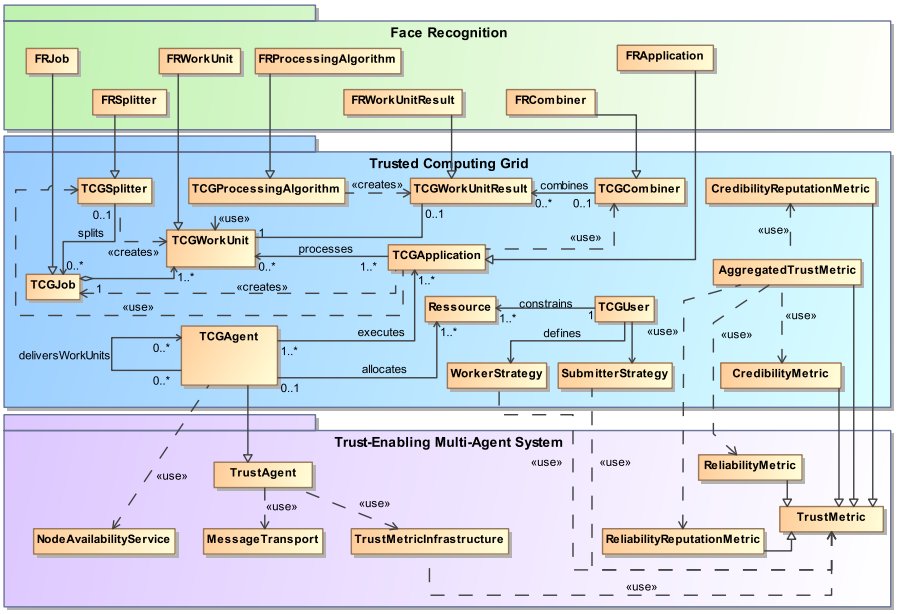

Figure 3: The Trusted Computing Grid Domain Model

4.2 TCG: TEMAS Layer

As explained in the previous section, `TCGAgents` show anticipatory behavior and must be able to communicate. For this reason, `TCGAgents` are specializations of `TrustedAgent` allowing them to use the `MessageTransport` facilities. In order to be notified when an agent becomes unavailable (e.g., in a situation in which a user shuts down the computer), `TCGAgents` can register with the `NodeAvailabilityService` provided by the TEMAS.

The TEMAS'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 TCG 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 `TrustMetrics` for credibility defined in the TCG and for reliability defined in the TEMAS. All these metrics adhere to the interface and functionality defined by the TEMAS's `TrustMetric`.

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

4.3 TCG: 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 TCG reference architecture. In this application, a job (see `FRJob` in Figure 3) 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`. Work units are processed by the `FRProcessingAlgorithm` and the corresponding `FRWorkUnitResult` 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 TCG 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, the `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. To determine whether an interaction has a positive or negative outcome, `TCGAgents` 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* [18]. 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 a member again. For example, an egoistic agent could decide to become more altruistic in order to increase its trustworthiness.

4.4 TCG: Components and Interactions

The classes of the TCG domain model can be grouped into several components as shown in Fig. 4. This modularization fosters separation of concern as well as high cohesion in the independent parts of the system. Each component can be exchanged with one that offers similar functionality as long as the interface is implemented. In Trusted Desktop Grids, this becomes especially important as different applications can use the same *TCGAgent* to submit and work on tasks.

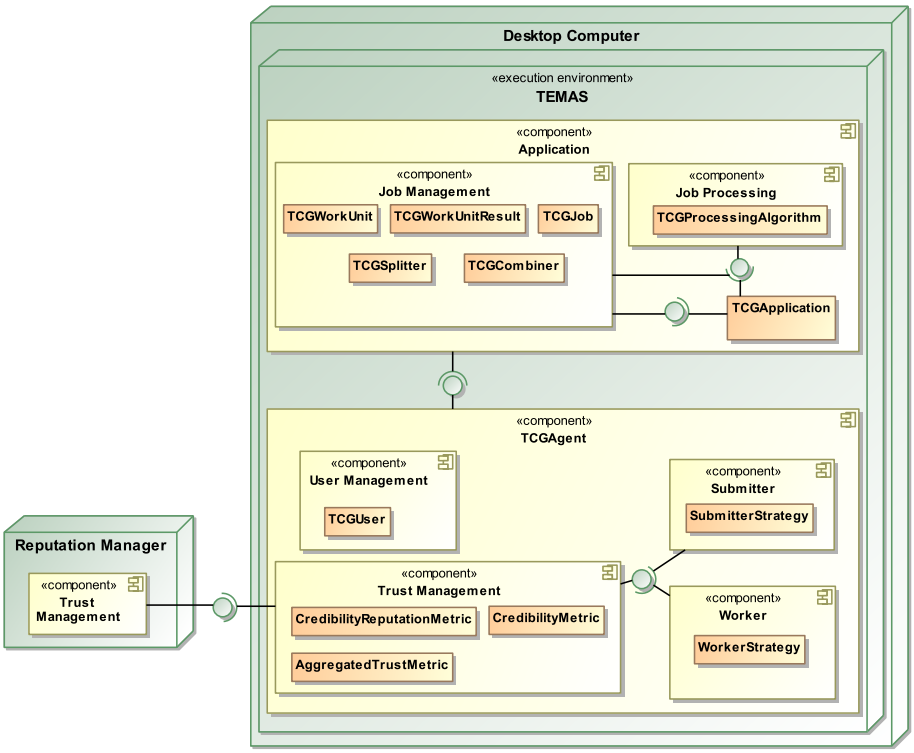

Figure 4: Components of a Trusted Computing Grid

The *TCGAgent* encapsulates the generic functionality of each agent that participates in the Trusted Desktop Grid. It represents its user in the system through the *User Management* component that manages the user's preferences w.r.t. the use of resources and strategies. The component *Submitter* deals with the selection of workers when a new job has to be submitted to the system. It employs a strategy chosen by the user or learned by the agent over the course of its interaction with the system. The component *Worker* deals with work units submitted to the agent based on a worker strategy, again either chosen by the user or learned by the agent. In addition, the agent component provides communication means for the composing components, thus defining the types of interactions between the agents.

On top of this generic agent are the different applications that create jobs and have the algorithms to process work units. The *Application* component has a *Job Management* component that splits `TCGJobs` into `TCGWorkUnits` and is able to recombine them. In addition, it houses a component for the processing of `TCGJobs` that encapsulates the `TCGProcessingAlgorithm`.

Both the *TCGAgent* and the *Application* are deployed in a TEMAS execution environment and usually run on a Desktop Computer. The TCG consists of a large number of *TCGAgents* running on many desktop PCs and interacting with each other through the TEMAS communication facilities. In some cases, it can be useful to access an external trust management system such as a reputation manager. The Reputation Manager can either be deployed on one of the desktop PCs or on a separate server.

5 An Architecture for Trusted Display Grid Applications

This section introduces the *Trusted Display Grid* (TDG) reference architecture. *Ubiquitous Display Environments* usually consist of one or more large displays in a public place (such as an university, a shopping mall, an airport, or a train station) as well as the mobile displays of the users interacting with them. Apart from simply reading various informations (such as weather, news, or schedules) off the public display, users can also interact with the public display with their mobile displays, taking interesting data with them, leaving it behind for other users, or somehow influence the way current information is displayed.

According to [14], the highly adaptive nature of such systems (changing the displayed content, the layout, the used modality, accommodating for other users as they approach, etc.) is not always understandable and self-explanatory for users. If they cannot follow the rationale behind a system adaptation or think that adaptation is unfitting for the current situation, their trust can be impaired, which, in the worst case, can cause them to stop interacting with the system. In addition, the authors of [12] have found that users are concerned with their privacy and the protection of their data when interacting with these systems.

Based on our previous work on managing user trust in Ubiquitous Display Environments [6], the TDG is a reference architecture for such highly adaptive systems which aim at managing their users' trust with regard to the challenges presented above. This includes means (such as sensors) to discern a user's current situation, identify threats to the user's trust and finally system adaptations to mitigate these threats.

Since the TDG involves various devices (such as servers, sensors, public displays, and mobile displays) and relies on their proper functionality and accuracy, as well as their ability to reliably communicate with each other, employing the TEMAS is advised, as it offers the necessary features to ensure this. As a mobile version of the TEMAS is currently in the prototype stage, all devices can be integrated this way.

In the next sections, we will show the concepts specific to the system class of TDG applications (see Sect. 5.1) and how those concepts can be integrated into the TEMAS (see Sect. 5.2). Finally, we will give an example of a TDG application employing the introduced concepts (see Sect. 5.3) as well as a component overview of TDG systems (see Sect. 5.4).

5.1 TDG: System-Class-Specific Concepts

One of the central concepts for TDGs (see Figure 5) are multiple displays `(Display)` that are part of such environments. They include public displays `(PublicDisplay)` as well as private displays `(PrivateDisplay)`, like for example smartphones or tablet PCs, that are controlled by the users `(User)`. The capabilities of the displays, such as their methods of input, their screen size and resolution, and other technical aspects are described by the `CapabilityProfile`. This information is used by the layout manager `(LayoutManager)` to create GUI layouts `(GuiLayout)` (using the factory pattern) that arrange GUI elements `(GuiElement)` in such a way that the layout works well on the different displays. Furthermore, the concept `GuiLayout` is used as the view element in

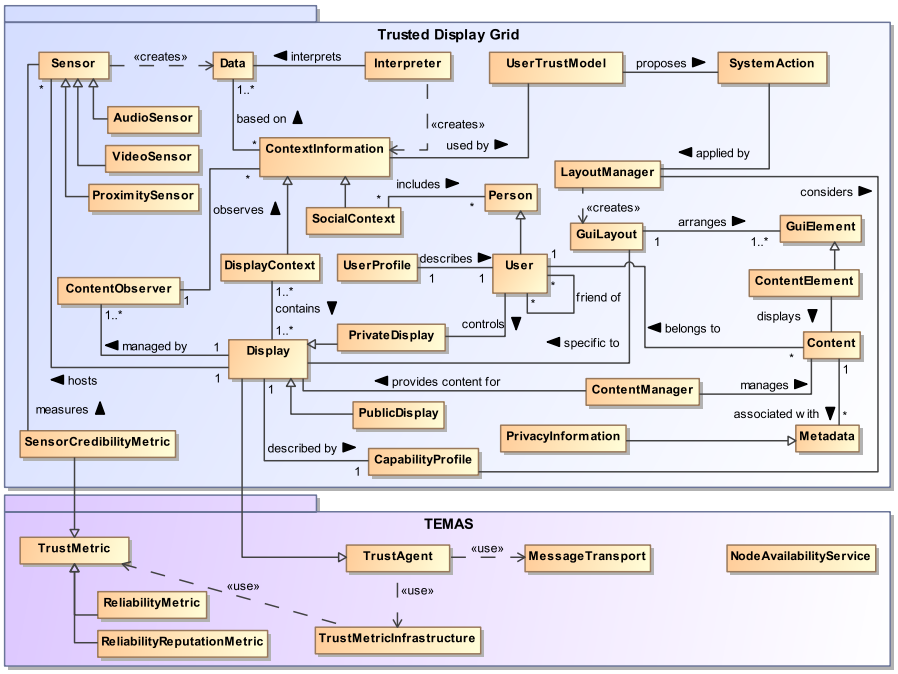

Figure 5: The Trusted Display Grid Domain Model

a Model-View-Controller-Pattern. The `Content` concept provides the model while the `ContentObserver` in conjunction with the `ContentManager` and the `LayoutManager` provide the controller part.

The model element of the MVC-pattern is the users' content `(Content)` that is displayed in specific GUI elements `(ContentElement)`. Each content is associated with metadata `(Metadata)` that describe, for example, if the content contains private data `(PrivacyInformation)`.

The control instance is divided into the management of content in non-trust-relevant interactions, such as pressing a "next" button, and the proposal of system actions `(SystemAction)` in trust-critical situations. The `ContentObserver` is responsible for deciding which of the two cases have occurred and to delegate the next step to the appropriate controller as shown in Figure 6.

In case no trust-critical user interaction occurred and no change in the `SocialContext` was detected, the `ContentObserver` reacts to a user request and calls the `ContentManager` that in turn retrieves the required content from the application and provides it to the related display.

Otherwise, the `UserTrustModel` is activated and proposes system actions to mitigate the trust-critical situation. To choose between the possible system actions, the `UserTrustModel` uses different kinds of `ContextInformation` that describe the current situation. This context information includes the relations between the present public and private displays `(DisplayContext)` and the current social context `(SocialContext)`. The social context contains information about the presence of other persons `(Person)` or users, their proximity to a display, and other relevant data such as the current noise level. If these persons

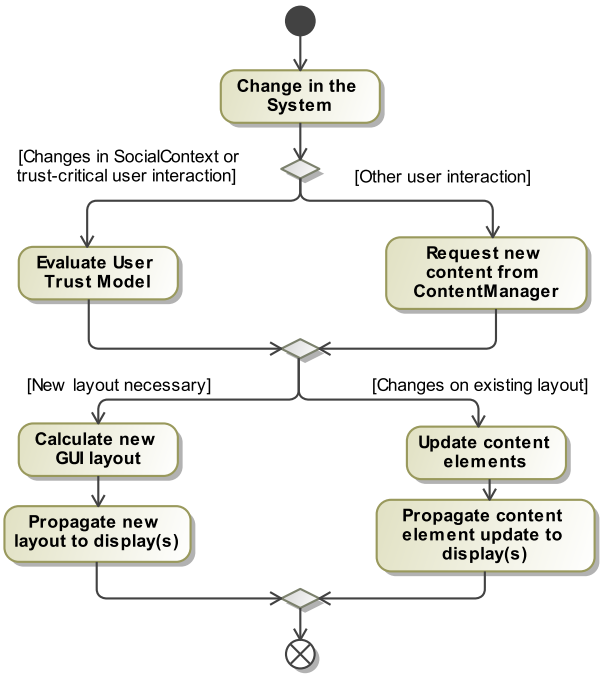

Figure 6: Based on information provided by the context, control is delegated to either the `ContentManager` or the `UserTrustModel`.

are also registered users `(User)`, the social context also provides information about the relationships between the present users. Users can, e.g., be friends, acquaintances, or strangers.

By knowing that displayed content contains a user's private data and that a stranger is standing next to this user, the User Trust Model, could propose that the privacy-critical data should be protected and, e.g., migrated to the user's private display.

However, to recognize such situations, a TDG has to be equipped with several sensors `(Sensor)`. These sensors, for example, recognize present persons (`ProximitySensor`), use video cameras (`VideoSensor`), or record ambient noise `(AudioSensor)`. The generated data `(Data)` are then processed by an `Interpreter` that serves as the factory for `ContextInformation`. The latter concept contains a high-level, semantically enriched situation description based on the sensor data. The accuracy of the sensors is measured with a specialized metric (`SensorCredibilityMetric`) that compares measurements of different sensors to detect deviations in the measured signal. This way, the credibility of the data can be assessed and used accordingly in the decision process.

Once a system action has been determined or new content has been selected for display, a new `GuiLayout` has to be calculated by the `LayoutManager` as described above. The new layout is then propagated to the displays (see Figure 6). Some system actions have no effect on the actual layout, e.g., if an existing content element has to be masked. In this case, only the content element update is propagated to the displays.

5.2 TDG: TEMAS Layer

Each `Display` is a `TrustedAgent` and uses the communication facilities provided by TEMAS's `MessageTransport` as well as the `NodeAvailabilityService` to determine which displays are currently active in the system, thus providing data for the `DisplayContext`.

The TDG environment uses different `TrustMetrics` to determine the reliability and accuracy of sensors. Since TDG applications usually rely on a variety of sensors, individual sensor `Data` can be cross-referenced to determine which sensors are most accurate in which situations. If, e.g., one of several microphones used by the application to determine the presence of people reports high noise levels without other microphones reporting similar data, the trust value of the microphone can decrease and thus be considered less when data is interpreted. Such an approach selects for the most accurate sensors and allows adaptivity with regard to changing environmental conditions that influence sensor accuracy.

5.3 TDG: An Exemplary Application Layer

In this section, we demonstrate an example application based on the TDG called Friend Finder [11]. Friend Finder is an interactive campus map that is displayed on `PublicDisplays` that are installed across a campus. It shows the current location (`PositionData` collected by `GPSSensors`) as well as the names and the pictures of the users' friends. A `FriendFinderUser` (see Figure 7) can browse through the displayed friends and retrieve the directions to a selected friend. The browsing and selection can be operated via the user's `PrivateDisplay`.

Since the friends' names, pictures, and locations are considered privacy-critical data [10], a privacy protection mechanism is integrated into the Friend Finder application. This mechanism, called `UserTrustModel`, is able to decide which `SystemActions` should be executed to protect the user's private data and thus to obtain the user's trust in the system. In the Friend Finder application, the five provided `FriendFinderSystemActions` are:

1. Do Nothing: Private data remain on the display.
2. Minimize: Private data shrink in size, but remain on the display.
3. Mask: Private data are occluded with solid blinders.
4. Remove Private Part: Private data are completely removed from the screen. The neutral elements, such as uniform icons (Friend Finder) remain on the public display.
5. Remove All: All data are removed. The screen displays only the map.

In case of actions 3-5 where the user's private `Content` is no longer visible on the `PublicDisplay`, the `Content` migrates to the user's `PrivateDisplay`, enabling the user to continue the interaction. Which `Content` is displayed on which `Display` is managed by the `FriendFinderContentManager`.

To recognize privacy-critical situations, the application utilizes cameras to recognize faces (`FaceRecognitionSensors`) and microphones to measure the ambient noise (`AudioSensors`). The `Data` (`FaceDetectionData`, `AudioData`)

collected by these `Sensors` are analyzed by `Interpreters` to recognize possibly approaching or already present other `Persons` by using the `AudioInterpreter` or the `FaceRecognitionInterpreter`.

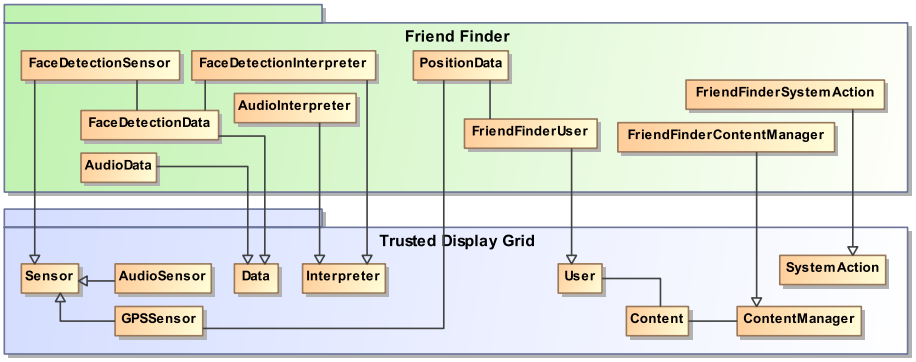

Figure 7: Excerpt from the class diagram of Friend Finder, an application based on a Trusted Display Grid

5.4 TDG: Components and Interactions

The classes of the TDG domain model can be grouped into several components as shown in Fig. 8. This modularization fosters separation of concern as well as high cohesion in the independent parts of the system. Each component can be exchanged with one that offers similar functionality as long as the interface is implemented. This way, different user trust models or different sensors, among others, can easily be deployed.

The *Trust Management* component contains the `UserTrustModel` (UTM) and the `SystemActions` that can be proposed by the UTM. It interacts with the displays from which it gathers `CapabilityProfiles` and `Data`, the *Layout Management* component which enacts the system actions proposed by the UTM, and the *Interpreter Component* which provides the `ContextInformation` that is required as the input for the UTM.

The *Layout Management* component contains the `LayoutManager` itself, as well as `GuiElements` and `ContentElements`. It is responsible for the creation of `GuiLayouts` that position these elements on the screen of the displays. Information about the displays is provided by the *Content Management* component and by the *Trust Management* component by the `SystemActions` it proposes.

The *Content Management* component contains the `ContentObserver` that handles all `Content` changes if they are not trust-critical on the one hand, as well as the `ContentManager` that provides all application content on the other hand. These classes also manage the `Metadata` and `PrivacyInformation` attached to content.

The *User Management* component handles the `Users` as well as their corresponding `UserProfiles` and provides this information to the *Interpreter Component*. This component in turn contains the `Interpreter` that uses this data to create the `SocialContext`. It also handles other `Data`, for example, to derive the `DisplayContext`. For this purpose, it uses the *Environment Tracking* component that handles `Sensors` and the `Data` created by them.

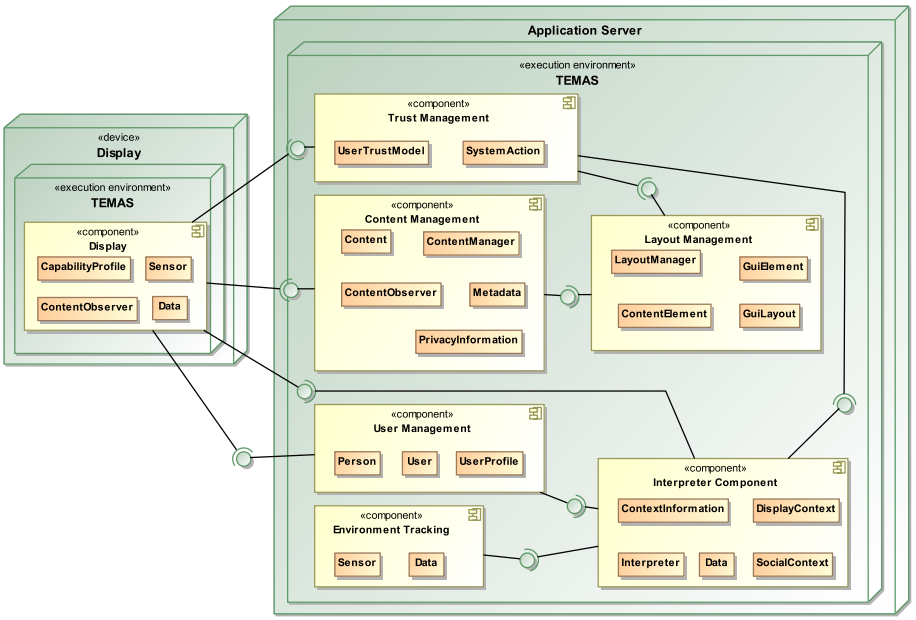

Figure 8: Components of a Trusted Display Grid

The `Displays` themselves have a `ContentObserver` to react to content changes directly if possible and to interface with the server's *Content Management* component. They can also provide `Sensors` and `Data` that are used in the *Interpreter Component* to determine the status of the display or of its surroundings. It also interfaces with *User Management* if it is a user's personal display to login and exchange personal information.

6 Discussion and Conclusion

In this report, we introduced three reference architectures for the Trusted Energy Grid (TEG), the Trusted Computing Grid (TCG), and the Trusted Display Grid (TDG), that can be used as templates for the construction of trustworthy applications in the field of energy management, desktop grid computing, and multi-user multi-display environments respectively. Since applications in 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, the Trust-Enabling Multi-Agent System (TEMAS), 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. Furthermore, we outlined three 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 TEG, we showed the application of Autonomous Virtual Power Plants that divide a power plant landscape into groups of power plants, and create and adjust schedules of power plants in order to balance energy sup-

ply and load in spite of unforeseen supply and load changes at all times. For this purpose, Autonomous Virtual Power Plants extend the TEG by the concept of an Autonomous Virtual Power Plant. Since the objective of energy applications based on the TEG may vary from application to application, only the basic behavior and properties of prosumers and other generic system participants are defined in the TEG.

As an example based on the TCG, we proposed a grid computing application for face recognition that is performed on multiple devices in parallel. As each application based on the TCG has the aim to solve a computationally intensive arithmetic problem as fast as possible, in contrast to the TEG, 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 TCG, including grid jobs, work units, and the algorithms that, among other things, generate jobs and process the work units.

Regarding the TDG, we presented the Friend Finder as an application based on a infrastructure of public and private displays allowing the users to locate friends on an interactive campus map and retrieve directions to them. The TDG provides mechanisms to protect the users' private data, e.g., their names, pictures, or locations, by migrating this data in privacy-critical situations from public to private displays and thus preserves the users' trust in the system. To recognize privacy-critical situations, the environment is equipped with sensors providing information about the presence of other people. The application defines the system actions used to protect the users' privacy.

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.

Acknowledgment

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

References

[1] G. Anders, L. Klejnowski, J.-P. Steghöfer, F. Siefert, and W. Reif. Reference Architectures for Trustworthy Energy Management and Desktop Grid Computing Applications. Technical Report 2011-11, Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, 2011.

[2] G. Anders, F. Siefert, N. Msadek, R. Kiefhaber, O. Kosak, W. Reif, and T. Ungerer. TEMAS – A Trust-Enabling Multi-Agent System for Open Environments. Technical Report 2013-04, Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, 2013.

[3] G. Anders, F. Siefert, J.-P. Steghöfer, and W. Reif. Trust-Based Scenarios – Predicting Future Agent Behavior in Open Self-Organizing Systems. In *Proceedings of the 7th International Workshop on Self-Organizing Systems (IWSOS 2013)*, May 2013.

[4] 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.

[5] 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.

[6] K. Bee, S. Hammer, C. Pratsch, and E. André. The Automatic Trust Management of Self-Adaptive Multi-Display Environments. In *Trustworthy Ubiquitous Computing*, volume 6 of *Atlantis Ambient and Pervasive Intelligence*, pages 3–20. Atlantis Press, 2012.

[7] 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.

[8] 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.

[9] 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.

[10] E. Kurdyukova, E. André, and K. Leichtenstern. Trust-centered design for multi-display applications. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '10, pages 415–420, New York, NY, USA, 2010. ACM.

[11] E. Kurdyukova, K. Bee, and E. André. Friend or Foe? Relationship-based Adaptation on Public Displays. In *Proceedings of the Second International conference on Ambient Intelligence (AmI'11)*, pages 228–237. Springer, 2011.

[12] C. Röcker, S. Hinske, and C. Magerkurth. Intelligent Privacy Support for Large Public Displays. In *Proceedings of Human-Computer Interaction International 2007 (HCII'07)*, pages 198–207, 2007.

[13] M. Roth, J. Schmitt, R. Kiefhaber, F. Kluge, and T. Ungerer. Organic Computing Middleware for Ubiquitous Environments. In *Organic Computing – A Paradigm Shift for Complex Systems*, pages 339–351. Springer Basel, 2011.

[14] L. Rothrock, R. Koubek, F. Fuchs, M. Haas, and G. Salvendy. Review and reappraisal of adaptive interfaces: Toward biologically inspired paradigms. *Theoretical Issues in Ergonomics Science*, 3(1):47–84, 2002.

[15] 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.

[16] F. Siefert, G. Anders, M. Sommer, and W. Reif. A Generic Framework for Simulating the EEX Power Market in Agent-Based Energy Management Applications. In *Tagungsband des Power and Energy Student Summit 2012*, January 2012.

[17] 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 B. Xie, J. Branke, S. Sadjadi, D. Zhang, and X. Zhou, editors, *Autonomic and Trusted Computing*, volume 6407 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg, 2010.

[18] 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.