

# Challenges of Secure and Reliable Data Management in Heterogeneous Environments

Norbert Siegmund, Janet Feigenspan, Michael Soffner, Jana Fruth, Veit Köppen  
Otto-von-Guericke University Magdeburg, Germany  
{nsiegmun, feigensp, soffner, fruth, vkoeppen}@ovgu.de

## ABSTRACT

Ubiquitous computing is getting more important since requirements for complex systems grow fast. In these systems, embedded devices have to fulfill different tasks. They have to monitor the environment, store data, communicate with other devices, and react to user input. In addition to this complexity, quality issues such as security and reliability have to be considered, as well, due to their increasing use in life critical application scenarios. Finally, different devices with different application goals are used, which results in interoperability problems. In this paper, we highlight challenges for interoperability, data management, and security, which arise with complex systems. Furthermore, we present approaches to overcome different problems and how an integrated solution can be realized using software product line techniques.

## Keywords

Digital Engineering, Security, Data Management, Software Product Lines

## 1. INTRODUCTION

In the near future, decreasing size and price of *embedded systems (ESs)* make information on real world objects cheap and easy to access and trace. These devices can collect and store data in nearly every place. States and environment of objects should be measured [8] and transferred to other systems in order to monitor a system or react to changes. For example, sensor networks are used for forest fire detection, flood warning, and environmental monitoring [3]. Even smaller devices such as *radio-frequency identification (RFID)* chips are used to describe and identify objects such as goods [12]. With this technique, information can be passed through different consumers (e.g., shipping agents or custom control) together with the object information itself. This enables new possibilities for locating a certain object and tracing the flow of goods, so stakeholders are able to monitor place and status of objects at any time. To make such information ubiquitously accessible, many different devices have to cooperate and communicate with each other. For example, to register and track cargo, RFID chips have to be scanned and information must be represented in a suitable format for stakeholders [16]. Sensors measure the state of goods, e.g., temperature of food, and transfer the collected data to an observation system.

Not only the heterogeneity of the used systems raises challenges. Data that is collected must be made available to

different stakeholders, which may require different views on the data. Besides the problem from making the data available, security issues have to be considered, because not each information should be accessible for every stakeholder. For example, a ramp team of a logistic hub requires only information about origin and destination of cargo, whereas the custom control is interested in the declared and real content of a package or container.

The above described examples are common for a *logistic hub*, where cargo arrives and leaves [25]. It usually consists of a large number of devices, which are located in different places. The distributed nature of such complex systems requires new approaches to cope with data reliability, security, and performance issues. For example, important data should not be stored only on a single system, but redundantly, to prevent data losses. However, the huge amount of incoming data makes a complete redundancy infeasible. Furthermore, embedded devices often have no direct power supply, so too much communication would result in a considerably decreased life time. Additionally, data management techniques have to consider data integration problems that are based on the heterogeneous data representation of the different systems and on the semantic differences of data description [20]. Considering data distribution and the huge amount of data, intelligent data management concepts are necessary to query the data in a high-performance way. All these addressed data management problems lead to the need for a new architecture, which fulfills all requirements.

The mentioned challenges for interoperability, security, and data management cannot be addressed in isolation. A concept for data distribution has to consider requirements for secure data storage and access, whereas the security mechanisms like encryption must consider the desired interoperability between different systems. For instance, to encrypt data, an asymmetric encryption algorithm may be suitable for a PDA, but not for a sensor with limited resources. In such systems, symmetric algorithms are used to address the resource constraints.

In this paper, we highlight problems of large distributed systems using a logistic hub as a running example. We present ideas to overcome interoperability problems and secure data management by creating tailor-made services. On top of our interoperability platform, data distribution and security mechanisms can be implemented. We describe such approaches and discuss future work that is required to

realize a complex system like the logistic hub.

The increasing relevance of embedded systems in digital engineering requires to consider software behavior during the development. By integrating the proposed platform into a virtual environment, we can model and simulate embedded systems including their software. This enables the detection of software failures and software vulnerabilities including its influence on hardware and interdependencies of different non-functional properties. For example, we can monitor the effect of security mechanisms on the performance of data management of an embedded system. This way, we can evaluate and test a product for different scenarios (e.g., security attacks and unreliable data) in an early design phase already within its intended environment. With these simulations, we can detect design flaws at an early stage, which saves money and development time.

## 2. PROBLEM STATEMENT

Due to the complexity of large systems, different problems arise. In this section, we describe challenges of data management in heterogeneous environments, possible data losses, and resulting threats to security.

### 2.1 Heterogeneous Environment

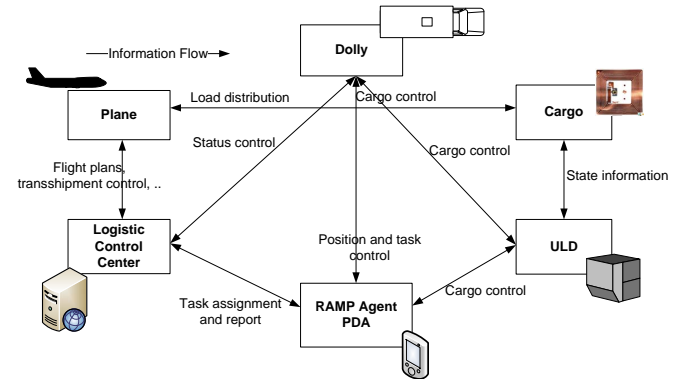
Complex systems consist of a number of smaller subsystems, i.e., components that fulfill concrete tasks. These subsystems have to cooperate to achieve the goals of an application scenario. A subsystem in turn is composed by devices, e.g., ESs, that are tailored to requirements of a specific domain. For example, a component in our logistic hub example can be a sensor network, which consists of sensors to measure and monitor the state of cargo. A PDA of a ramp team member is another component and is responsible to assign transport tasks and protocol and control the cargo flow (see Figure 1). Due to the fact that both subsystems have to cooperate, problems arise, such as different hardware devices, communication infrastructure and protocols, and semantic viewpoints on the same data. Furthermore, used devices are often delivered from different companies, which results in differently equipped systems. Moreover, due to the absence of standards, the interoperability of components cannot be guaranteed. Achieving interoperability is a major challenge in this area and needs to be addressed on different granularity levels. To ease the development, we divide the challenges for interoperability into three classes: a) syntactical interoperability, b) semantical interoperability, and c) pragmatical interoperability.

**Syntactical Interoperability.** Syntactical interoperability describes the ability to overcome differences between multiple systems regarding communication. In this class, different interfaces, data models, and hardware have to be integrated. In practice, data between sensors is often transmitted as pure ASCII code, while the communication of PDAs is usually realized with an XML based protocol. Interfaces are needed to overcome such differences. Furthermore, the lack of uniform data description, e.g., with a standardized XML schema, rises question about data integration. In small systems, such problems are often solved with a manual adaption of the protocols or implementation of the participating systems. However, this approach cannot scale for large and complex systems, in which the devices might frequently change over

life time. For example, if different logistic companies, each with their own cargo descriptions within RFID chips, send their goods over the same hub, the hub must provide a variable mechanism to read the data. A manual adaption would not be efficient and applicable.

**Semantical Interoperability.** Semantical interoperability is the capability to integrate data and functionality from different sources. Ensuring this kind of interoperability between different components requires methods to obtain a consistent view on described data or data models and functionality. The meaning of data and system functionality needs to be integrated to avoid misinterpretations and system failures. For example, in the logistic hub, the weight of cargo can be given in pound (for US) or in kilogram (for EU). This must be considered when data has to be aggregated from several sources, which is the case if a plane is loaded. Before information can be integrated, syntactical interoperability is necessary to gain the required data.

**Pragmatical Interoperability.** Pragmatical interoperability describes the usage of a system by different stakeholders or user groups. Process information from different data sources is needed to guarantee an efficient and reliable working process. The cooperation between multiple user groups must be considered, too, including their own functional and non-functional requirements and views on data. Thus, this class of interoperability requires that the semantical interoperability is already realized and users can act on integrated models and information.



**Figure 1: Participating components in the logistic hub.**

In Figure 1, we depict a visualization of participating components in a simplified logistic hub. Between these components, we show examples of transferred information. Even in such a simplified scenario, we have to consider different devices with varying system capabilities as for communication hardware, different granularity and amount of data, and so on. To overcome these differences, a uniform method of accessing and providing information is needed. Moreover, the high implementation effort caused by a large amount of necessary system adaptations needs to be handled efficiently. In Section 3.1, we present our approach of a *service-oriented architecture (SOA)* to provide the required uniformity [25, 17]. We highlight, how the different limitations of the devices and the varying requirements on the functionality of a service including scenario specific adaptations can be addressed by

generating a service based on a *software product line* [7].

## 2.2 Security Challenges in Distributed Environments

Embedded systems show great promise for industrial sectors, e.g., the production industry, logistics, and medical engineering. For example, RFID technology is expected to be used for efficient processes on airports in the future. To ensure the security of heterogeneous ESs, their characteristics have to be considered. In the following, exemplary properties of complex systems, consisting of several ESs, and their importance for the system security are described to motivate the development of new adapted security concepts and measures.

As described in Section 2.4, complex systems like the logistic hub consist of various subsystems, which are responsible for different subprocesses. Besides data heterogeneity, heterogeneous hardware and software influence the system security as well. For example, RFID scanners can be equipped with different hardware encryption modules, which encrypt the communication between RFID tags and scanner in different ways. The usage of a weak encryption algorithm in one RFID system can lead to a system vulnerability, which can be potentially misused through eavesdropping and modification of communicated data.

Since mobile systems are often used in indoor and outdoor environments, they cannot be protected sufficiently by common infrastructure measures like access-controlled buildings or locked rooms. Therefore, the possibility of attacks through physical access on distributed components has to be taken into account and proper countermeasures need to be developed. The mobile behavior of several ESs often implicates the use of wireless communication between different components of the system, e.g., the communication between the ramp team and the control center. The air is used as shared communication medium, which eases remote attacks without the physical access to technical components of the logistic system. Known threats to wireless communication are, e.g., eavesdropping, unauthorized manipulating of communicated and stored data on ESs, or *Denial-of-Service (DoS)* attacks. Affected data are, e.g., sensor data values (GPS coordinates of valuable freights, control commands, etc.) and transportation instructions to a ramp agent. Mobile distributed systems are often non-closed systems. Their system borders are extended because of their mobile behavior and usage of wireless communication of several ESs, which constitute ad hoc networks. For example, a device that is infected by malicious code of an ad hoc network can threaten the security of all devices linked to the wireless network. Malicious code can also be introduced by non-secure devices, which are connected through the Internet. The higher the complexity of a system that consists of networks of several ESs, the higher the number of objects that have to be protected against attacks [10]. Besides direct functional impacts to security, indirect structural impacts, e.g., interdependencies of security and safety incidents have to be considered, too. For example, deliberate jamming of RFID frequencies can prohibit the reading of all cargo tags in the range of an RFID scanner. These DoS attacks can disturb the reliability of logistic processes.

Various stakeholders, which use parts of the system on differ-

ent locations, can have different requirements to the system, which implicate different access to data and system resources. For example, a ramp team needs detailed information on certain cargo, e.g., weight and destination. The control team needs information about all cargo on the apron. In this context, the correct handling and access to sensitive data, e.g., person related data, is important to avoid misuses. Furthermore, there are stakeholders with competing activities, e.g., different cargo handlers that compete for an order to transport cargo on the apron. The system has to ensure correct data access to avoid misuse.

## 2.3 Reliability in Distributed Data Storage

Requirements of continuous data availability are needed for reliable data management solutions, e.g., when devices are unavailable. In order to enhance robustness, gateway or proxy concepts are standard in the domain of sensor networks (SNs). Data are recorded and transmitted via proxies to storage systems. Considering the facts of high hardware costs, system size, and lack of scalability, these solutions are inappropriate for wireless SNs. A more feasible approach is to deploy system functionality, which compensates single device failures. Concepts for reliable distributed data storage and its organization already exist for the domain of PCs and servers. Nevertheless, constraints of the wireless SN domain are usually not considered in existing solutions.

Different components of a logistic hub measure, compute, and record various data. Different hardware systems and different user requirements, which interact with complex systems, lead to heterogeneity that complicate a solution. Detailed mapping of process information increase data capacities. All data objects, aggregated or from direct sources, have to be transmitted in a complex system to enable analyses. However, due to the magnitude of components and transmitting restrictions as well as the amount of data, a centralized data storage solution is not efficient or feasible.

As a consequence, we need distributed data storage, which allows visualization (e.g., in Virtual Reality systems [17]), simulation, or analyses, which are common in the domain of Business Intelligence. Furthermore, a reliable architecture has to consider additional requirements that go along with the heterogeneity and quality requirements. Hardware failures, limited information communication due to bandwidth restrictions, storage capacity and power supply of ESs, and new methods and concepts (e.g., mobile On-line Analytical Processing solutions [19]) are some examples.

A reliable data management solution also depends on the application domain (cf. Section 2.4 for security issues for data management). For different users, data has to be prepared according to the process state and user's requirements. Based on derived information, different time constraints have to be considered. For instance, information about hazardous cargo for the ramp team or accidents, which influence the overall logistic hub for the control center team, has to be processed and delivered as soon as possible. We distinguish between two classes of information concerning the level of importance. On the one hand, high priority information, such as anomalies, accidents, and hazardous cargo, has to be communicated in real time with all related data. On the other hand, process information, e.g., cargo status, can be

aggregated and transmitted on request or at certain time intervals.

## 2.4 Secure Data Storage

A logistic hub consists of heterogeneous components that need and produce heterogeneous data, for example cargo, dollies, and surveillance cameras. *Cargo* enters and leaves the hub and is transported and stored within the hub. During transportation, different requirements for different cargo must be fulfilled, for example a constant low temperature (cooling chain), or special transportation and storage requirements for hazardous cargo like chemicals. For storing the data, cargo can be tagged with RFID chips, which store data about the cargo's origin and destination and whether it needs to be cooled or is hazardous. *Dollies* are used to transport cargo within the hub and have information about destination, estimated time of arrival, traffic jams, etc. *Surveillance cameras* record events occurring in the hub produce streaming data that describe the events.

As described in detail in Section 2.2, numerous threats to security occur in complex systems. For example, RFID tags of cargo or data of surveillance cameras can be manipulated. In order to prohibit threats to stored data, security measures for data storage have to be implemented. This poses problems because of the heterogeneity of the components. For example, an asymmetric encryption algorithm may be suitable for a normal PC (used in the control center) or PDA (used by the ramp teams), but for embedded devices like in ULDs or RFID, the resource constraints forbid such an extensive algorithm. How can we assure that encrypted data can be transferred between all those components without having conflicts due to different encryption algorithms? At the same time, how can we assure sufficient protection of the data when we use light-weight encryption algorithms, which may be not as secure?

If security breaches occurred despite all careful design, they need to be brought to attention as soon as possible. For small environments this may sound trivial (e.g., imagine a security guard who has to monitor few surveillance cameras), but for complex systems, not everything can be monitored (even with enough man power). Hence, simply observing all information about processes in the logistic hub does not allow a timely intervention for security breaches. Instead of letting people sift through all data, user-friendly warnings need to be created to assure that security threats and breaches are handled as soon as possible. This is problematic especially in our case due to the heterogeneity of the data.

From those diverse data, how can security warnings be generated? How can a just-in-time intervention be allowed? These are difficult questions for a large complex system: data and components are heterogeneous, requirements change depending on the component, domain, and application scenario. In order to deal with all those problems, we propose the use of variability mechanisms, as we outline in Section 3.4.

## 3. APPROACHES TOWARD A UNIFIED SOLUTION

The challenges stated above are only an excerpt of possible facts that have to be considered in right-time process control systems. We design a first step architecture realized with

software product lines techniques that meets our addressed requirements of interoperability, data reliability, and security seamlessly. In this section, we first introduce the concept of software product lines. We continue with the description of concrete approaches for every problem domain and highlight questions that are still open.

*Software Product Lines.* A *software product line (SPL)* can be defined as "[...] a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of [...] assets in a prescribed way." [7]. The "particular market segment" or *domain* the SPL is developed for has to be agreed on by stakeholders of the domain, e.g., managers, developers, or customers. There are many examples for the domain of ESs, such as software for diesel engines, satellite ground control system, or software for mobile phones<sup>1</sup>. *Assets* can be understood as the modules that encapsulate different concerns. Once developed, these assets can be reused in different products resulting in a decreased time-to-market and development effort. Further advantages of SPLs include efficient tailor-made applications, increased quality, reduction of maintenance effort as well as the ability to better cope with complexity and evolution of software systems [7]. Features of an SPL represent end-user visible characteristics of the program [15]. To derive a product, a user selects features of an SPL to fulfill her requirements. These features are mapped to assets that implement the desired functionality. Afterwards, a product is generated using a certain implementation technique. Common implementation techniques include preprocessors (e.g., C/C++ #ifdefs), component-based approaches, aspect-oriented programming, and feature-oriented programming.

With a preprocessor, developers frame source code fragments with #ifdef and #endif statements to denote that they belong to a certain feature. If a product is generated, source code that belongs to features that are not selected is removed. Products can also be created by composing source code modules. Components are often coarse grained, such that often multiple feature map to one component. New programming paradigms like an aspect-oriented and feature-oriented programming enable the implementation of a feature in a single module. This is even possible when features represent cross-cutting functionality (i.e., functionality that interacts with the functionality of other features).

Feature models are used to model features and the variability of an SPL. In the lower part of Figure 2, we show two feature models. The root node represents the concept of the SPL. By defining relationships between features, we can constrain the configuration of an SPL to derive only valid products. Such relationships are expressed with filled dots for a *mandatory* feature selection, empty dots for an *optional* selection, and *alternative* features (empty arc) and *inclusive* or features (filled arc).

### 3.1 Interoperability Platform

The heterogeneity of hardware and the differences in required functionality rise the need of a platform, which is capable of fulfilling the requirements of interoperability. We propose to

<sup>1</sup>[http://www.sei.cmu.edu/productlines/spl\\_case\\_studies.html](http://www.sei.cmu.edu/productlines/spl_case_studies.html)

use services as a uniform interface to access and to provide data that is stored or processed on an ES. This is based on the fact, that our scenario is defined by business processes and services map these processes well. Furthermore, such a service-oriented architecture provides the flexibility to deploy on-demand functionality (as a service) on a device. Then, the communication can be implemented, either as service subscription to retrieve data of another source or as service publication to enable the access on the device's data. In [25], we demonstrate the technical realization of such an interoperability platform. We extended this approach to include solutions for distributed data storage and security requirements in the services. Moreover, we explicitly address the influence of *non-functional properties* for underlying devices and current application goals. Non-functional properties are characteristics of software and need to be optimized depending on the application scenario. For example, for ESs non-functional properties like power consumption, binary size, and consumed working memory are such important properties. Optimizing the power consumption can lead to a significantly increased system life time, whereas the optimization of binary size may result in cheaper systems.

In Figure 2, we show the structure of a service consisting of different components that are generated from different dependent SPLs. Features in with gray background have further subfeatures that are not shown. A component is not a strictly separated implementation module as commonly considered, but constitutes a logically separated unit of functionality required for an abstract design of a service. On the implementation level, different SPLs are entangled and composed to form one physical running service. In Section 3.4, we outline how this can be realized. The port component of a service is responsible for external communication, i.e., the interface of the service. An excerpt of this SPL is shown in the upper part of Figure 2. The SPL consists of features that implement various communication protocols, support different communication hardware, and the functionality to retrieve or send data. The data is maintained in a buffer-like form to be accessible by other components of the service. To enable the access of the port's functionality, the implementation of other component SPLs are entangled with this SPL. That is, the port SPL is enriched by features that enable the interaction with other components. For example, the security component is responsible for authentication, encryption, etc. Thus, incoming data have to be verified according to origin and decrypted accordingly. Data that need to be send might be certificated and encrypted. These functions are defined in a security SPL (see lower part of Figure 2) and the implementation of *send* and *receive* methods of the port SPL are refined by the necessary functionality. Afterwards, data can be processed by other application related SPLs, e.g., data can be aggregated, analyzed, or stored. The actual needed functionality depends on the application scenario for which the service is deployed. Thus, there are different possible SPLs fulfilling various tasks for which the service is intended to be used. The distributed data storage is one important use case for this functionality, which we discuss in Section 3.3.

As we mentioned above, non-functional properties are important for ESs or when certain quality attributes such as performance are vital system criteria. Due to changing en-

vironments, different non-functional properties have to be optimized at different points in time. On the one hand, we may need maximal performance in some situations, e.g., when a hazard is detected. On the other hand, we may want to save as much power as possible for a normal system behavior. To enable these changes, we have to integrate a monitoring mechanism to observe the system state, e.g., current network load, and to analyze data, e.g., to detect anomalies. If the monitoring mechanism detects an important change, there are different possibilities to react to this event:

- **State Changes.** The common case to react to an event is to change system properties like increasing the bandwidth for communication, if the buffer is too small.
- **Algorithm Switch.** The execution path of an application is changed via a switch. This is necessary, e.g., to go into an idle mode for power savings.
- **Reconfigure Service.** The most complex change is to reconfigure an SPL of the service and to adapt the running service. This is necessary, e.g., if the complete binary code of all SPLs does not fit on a device. Then, the service has to be adapted according to the new functionality. In [25], we presented a technique to enable this kind of change. These large changes might be required, if related devices fail and a new device must take over the tasks.

Implementing these mechanisms requires to refine existing functionality of application SPLs. For instance, the port SPL's *send* method must be refined to measure and store the current load balance. Rules for defining non-normal states and resulting actions can be separated in an own non-functional property SPL that is reused in different systems. The strict separation of different domains that contribute to a single service eases the design and development. However, it increases complexity for the derivation of a valid service. In Section 3.4, we address this problem.

## 3.2 Security Concepts for Distributed Embedded Systems

In this section, selected security concepts, which represent solutions for the outlined challenges for embedded systems are described. These concepts are known from the desktop IT domain or constitute isolated security solutions for embedded systems. At the end of this section, a motivation for a new holistic security concept for ESs is given.

The mobile nature of some ESs and the increasing probability of attacks by physical access are challenges for system security of ESs. Exemplary countermeasures to avoid physical access of embedded components to the airport apron are authentication measures and access controls (e.g., RFID smart cards and biometric authentication). Countermeasures to prohibit manipulations on components, if a physical access is possible, are for example cryptographic measures [13], which are also suitable for protecting wireless communication against potential attacks. For example, unauthorized reading of communicated or stored data from the RFID tagged cargo can be avoided via mutual authentication [14] between the participants of a communication, during which the mutual knowledge of a secret key is proven. Encryption (see Figure 2), e.g., stream ciphering, is a common countermeasure

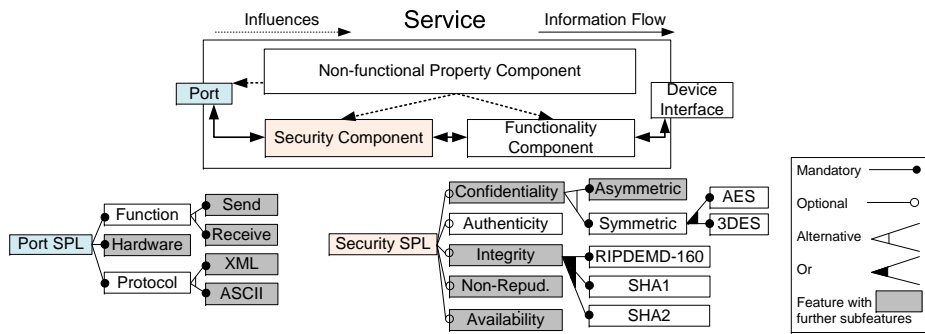


Figure 2: General architecture of a service composed by multiple software product lines.

against eavesdropping of wireless communications and replaying of data [12]. The manipulation of data can be detected with hash functions (see Figure 2, Security SPL - Integrity). These measures have to be adapted for their application in ESs, which require lightweight solutions [1, 5] whilst ensuring an equivalent security protection. New holistic solutions to protect these complex ESs against aforementioned threats have to be developed in the future [10].

Currently only separate concepts and methods for the protection of safety and security exist. Concepts that consider the interdependencies of security and safety incidents to ensure the correct function of logistic processes and avoid damages to objects or humans are rare. First solutions for a holistic security protection of ESs are developed for the automotive domain [24]. New approaches are needed for the logistics domain.

Different requirements of various stakeholders can be satisfied with different views on data. Various access control models exist, like Bell-La Padula [2] or role-based access control [11], which have to be adapted to special requirements of distributed ESs. Sensitive data, like personal data, has to be protected against access of unauthorized persons or parties. Measures to implement data privacy laws include anonymization methods, cryptographic encryption, and access control mechanisms [6].

The IT-Grundschutz standard [4] of the German Federal Office for Information Security recommends a security modeling for a baseline IT security of Desktop IT systems. Furthermore, in this standard modular security measures are described. Our concept is based on the aforementioned standard. It includes the modeling of all technical components, their stored and communicated data, and the information flow between different components to secure a network that consists of several distributed ESs. Furthermore, the security requirements of the components and adequate security measures have to be integrated into the model. Existing modeling languages, like SysML<sup>2</sup> or Secure UML [18], are not suitable to model all the security-related properties and requirements of distributed ESs [9]. Therefore, these languages have to be adapted to model all necessary security aspects. The modelling of non-functional properties of embedded systems and simulations of interdependencies between safety and security impacts in complex systems (see introduction) will lead

to more safe and secure systems.

Our example of the logistic scenario illustrates the benefits of a holistic security approach. A possible application is a monitoring system located in the control center of the airport, which visualizes anomalies, e.g., potential attacks or failures. For this monitoring system, a model of all the components of the logistic system, their properties, and possible information flows between components and security requirements is needed. The monitoring system shows potential threats and their causes as well as possible consequences to system security or safety. Furthermore, non-functional properties of ESs, like security, are illustrated in form of visual or acoustic warnings, which inform operators in the control center about potential threats and recommend activities. One example should illustrate the idea. The monitoring system identifies two cargo units with the same RFID identification number on the airport. The cause can be a potential replay attack, which implicates, that security personnel has to be informed.

Besides modeling technical components, modeling the behavior of attacker has to be considered, so that risks to secure and reliable functions of distributed ESs can be rated.

### 3.3 Distributed Data Management

In order to address challenges of data distribution, we start with a classification for sensor data and corresponding measures. This helps us to create a data distribution schema, which constitutes the basis of our distributed data architecture. The architecture is implemented on top of the interoperability platform, which overcomes problems of hardware heterogeneity (cf. syntactical interoperability in Section 2.1). In [27], we classified data according to time constraints. This classification categorizes data sources regarding the required access time. In the following, we give an overview of the three classes:

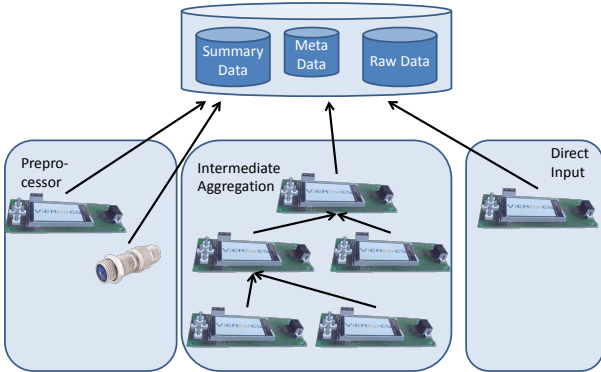
- Data sources, which *preprocess* data directly on the device, are reducing the amount of transmitted data. Optimization of the network load is focus of these devices. Preprocessing includes data aggregation, data checks, data cleaning, and data filtering.
- Data sources, which only aggregate data, are also reducing the network load. However, we also include transmitting devices into the class of *intermediate aggregation*, that send their data to aggregators. Furthermore, the compensation of single device failures, due to power loss or communication breaks achieves an

<sup>2</sup><http://www.omg.sysml.org/>

increased reliability.

- Data sources, which directly transfer their data to analyzing systems, such as monitoring or decision support systems, are necessary for time critical requirements. In the domain of data warehousing, these *direct input* devices can be seen as traditional data warehouse sources. In our logistic hub example, mission critical data is straight forward transferred to the control center.

Based on this classification, a first idea of an architecture that takes DBMS functionality into account is shown in Figure 3. For example, analyses such as outlier detection of measured values can be executed on data of our preprocessor class.



**Figure 3: Time-constraint-driven architecture for distributed data sources.**

Due to the fact that this basic architecture only considers time constraints, it is not sufficient for a complex system. Additionally, we need to integrate further facets of data usage. As an example, take the importance of data: if important data get lost, the costs are higher than if low priority data get lost. Hence, high priority data should be stored redundantly. Data have to be distributed, such that it can be accessed efficiently, but on the other hand do not get lost because of limited storage capacity. The amount of data, process, and resource constraints have to be considered as well. Furthermore, with distributed data, one has to take into account that communicating data causes high power consumption. Thus, a distributed schema is required, which considers non-functional properties, like limited bandwidth and power consumption. Due to perpetual changes of complex systems, an integrated schema has to be adapted dynamically to system changes.

This leads to a classification of data according to the following criteria: a) access time, b) resource constraints, c) process constraints, d) data priority, and e) quality issues.

In future work, we need to combine and integrate the classifications to be able to derive a data distribution schema for complex systems, which deals with our addressed requirements.

### 3.4 Secure Data Storage

In Section 2.4, we outlined the problems regarding secure data storage in heterogeneous systems and derived the need for variability mechanisms. In our work, we propose the use of SPLs, because they have proven useful in the context of

embedded systems<sup>3</sup>.

One example of a *data base management system (DBMS)* SPL for embedded systems is FAME-DBMS<sup>4</sup> [23, 26]. It is highly customizable to meet the requirements of different embedded systems. In its smallest variant, FAME-DBMS requires only 10KB storage capacity. This variant supports only in memory data storage and is not capable of specialized search indexes or security mechanisms. Such a DBMS is often used in deeply embedded systems, in which resources are very limited. Using the SPL approach, we can configure FAME-DBMS to support more advanced data management functionality, too, such as a B-tree search index, support for permanent data storage for different hardware (e.g., EEPROM, flash), and advanced buffer strategies. Furthermore, we can support an SQL-like query engine for ad-hoc queries for devices with according resources, for example a PDA.

Because of its customizability for ESs, FAME-DBMS is a suitable starting point for data storage in complex systems. However, it misses security features (see Security SPL in Figure 2). In order to use FAME-DBMS, we need to integrate security features. One idea to proceed without adapting the implementation of FAME-DBMS is to combine FAME-DBMS with a security SPL. This concept of combining SPLs to benefit from their advantages is referred to as *multi software product lines*, as described in [22, 21]. Using this approach, we create an SPL with security features, and combine the security SPL with FAME-DBMS according to formalized rules [21], depending on a concrete application scenario. If we have features that are not orthogonal, i.e., share code, the formalized rules describe the combination via constraints between SPLs. *Constraints* ensure that when a feature of one SPL is selected, required features of other SPLs are selected, as well. For example, if feature Access Control of the security SPL was selected, a constraint would make sure that the required feature Rights Management of FAME-DBMS is also selected.

Now, having an approach to solve the data storage problems, we need to think of ideas to bring threats to security to attention. Like for adding security, we can create an SPL that is responsible for visualizing the security status of the components. Combined with FAME-DBMS and security SPL, we have an approach to solve the problems described in Section 2.4.

## 4. CONCLUSION

In this paper, we outlined the challenges for security, data management, and interoperability in complex systems. We showed, how heterogeneity of different hardware, communication protocols, etc., can be handled by using services as a uniform interface. Services are generated from different SPLs that fulfill different tasks. For example, one SPL is responsible to enable the communication infrastructure, another SPL enforces certain security mechanisms, and a third SPL realizes the data management. Using SPLs, we can reuse implementation of functionality across different systems and tailor a service to functional and non-functional requirements. The combination and integration of all SPLs

<sup>3</sup>[http://www.sei.cmu.edu/productlines/spl\\_case\\_studies.html](http://www.sei.cmu.edu/productlines/spl_case_studies.html)

<sup>4</sup><http://fame-dbms.org>

can be solved by using the multi software product line approach that we developed. There are more challenges in the logistic hub scenario, e.g., process safety issues, which need to be addressed in future work. Additionally, we will realize a subdomain of our logistic hub example, in which we need to integrate all participating SPLs.

## Acknowledgments

Janet Feigenspan, Jana Fruth, Norbert Siegmund, Michael Soffner, and Veit Köppen are funded by the German Ministry of Education and Science (BMBF), project 01IM08003C. The presented work is part of the ViERforES<sup>5</sup> project. The authors would also like to thank Tobias Hoppe, Jana Dittmann, Frank Ortmeier, and Olaf Poenicke for their assistance and preliminary work.

## 5. REFERENCES

- [1] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-Key Cryptography for RFID-Tags. In *PerCom Workshops*, pages 217–222, 2007.
- [2] D. E. Bell and L. J. L. Padula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, The MITRE Corporation, Bedford, MA, 1973.
- [3] L. Bodrozic, D. Stipanicev, and M. Stula. Agent based data collecting in a forest fire monitoring system. *International Conference on Software in Telecommunications and Computer Networks*, 0:326–330, 2006.
- [4] Bundesamt für Sicherheit in der Informationstechnik (BSI). Standard 100-2: IT-Grundschutz Methodology, 2008.
- [5] S. Canard, J. Etrog, and I. Coisel. Lighten Encryption Schemes for Secure and Private RFID Systems. In *1st Int'l Workshop on Lightweight Cryptography for Resource-Constrained Devices – WLC'10*, 2010.
- [6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, pages 84–88, 1981.
- [7] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.
- [8] T. DeMarco. *Controlling Software Projects: Management, Measurement and Estimation*. Yourdon Press, Englewood Cliffs, NJ, 1982.
- [9] M. Eby. Integrating Security Modeling into Embedded System Design. Master's thesis, Graduate School of Vanderbilt University, Nashville, Tennessee, 2007.
- [10] C. Eckert. Ambient Intelligence: Neue Herausforderungen für die IT-Sicherheit. *TU-Darmstadt, thema Forschung*, pages 22–27, 2007.
- [11] D. F. Ferraiolo and D. R. Kuhn. Role-Based Access Control (RBAC). In *15th National Computer Security Conference*, pages 554 – 563, 1992.
- [12] K. Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards and identification*. Wiley, 2003.
- [13] A. Juels. Strengthening EPC Tags Against Cloning. In *ACM Workshop on Wireless Security 2005*, pages 67–76. ACM Press, 2005.
- [14] A. Juels and S. A. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology: Proc. of CRYPTO 2005*, volume 3621 of LNCS, pages 293–308. Springer-Verlag, 2005.
- [15] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
- [16] V. Köppen and G. Saake. Herausforderungen im Digitalen Engineering: Einsatz von Virtueller Realität im Prozessmanagement. *Industrie Management*, 26:45–48, 2010.
- [17] V. Köppen, N. Siegmund, M. Soffner, and G. Saake. An architecture for interoperability of embedded systems and virtual reality. *IETE Technical Review*, 26(5):350–356, 2009.
- [18] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *5th Int'l Conf. on The Unified Modeling Language*, volume 2460, pages 426 – 441, 2002.
- [19] I. Michalarias. *Multidimensional Data Management in Mobile Environments*. PhD thesis, Freie Universität Berlin, 2007.
- [20] F. Naumann, U. Leser, and J. C. Freytag. Quality-driven integration of heterogenous information systems. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *VLDB*, pages 447–458. Morgan Kaufmann, 1999.
- [21] M. Rosenmüller and N. Siegmund. Automating the Configuration of Multi Software Product Lines. In *Proc. Int'l. Workshop on Variability Modelling of Software-intensive Systems*, pages 123–130, 2010.
- [22] M. Rosenmüller, N. Siegmund, C. Kästner, and S. S. ur Rahman. Modeling Dependent Software Product Lines. In *GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering (McGPLE)*, number MIP-0802, pages 13–18. Department of Informatics and Mathematics, University of Passau, 2008.
- [23] M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich, O. Spinczyk, and G. Saake. FAME-DBMS: Tailor-made Data Management Solutions for Embedded Systems. In *EDBT'08 Workshop on Software Engineering for Tailor-made Data Management*, pages 1–6, 2008.
- [24] S. Schulze, T. Hoppe, J. Dittmann, and G. Saake. Pauschalisierte Sicherheitsbetrachtungen automotiver Systeme. In *DACH Security 2009*, Bochum, 2009.
- [25] N. Siegmund, M. Pukall, M. Soffner, V. Köppen, and G. Saake. Using software product lines for runtime interoperability. In *Proc. of Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE)*, pages 1–7. ACM Press, 2009.
- [26] N. Siegmund, M. Rosenmüller, G. Moritz, G. Saake, and D. Timmermann. Towards robust data storage in wireless sensor networks. *IETE Technical Review*, 26(5):335–340, 2009. Workshop on Database Architectures for the Internet of Things (DAIT).
- [27] M. Soffner, N. Siegmund, and V. K. Mario Pukall and. Towards real-time data integration and analysis for embedded devices. In *In Post-Proc. GI-Workshop on Foundations of Databases*, pages 51–55, 2009.

<sup>5</sup><http://vierfores.de>