Model-based Security Analysis for Mobile Communications*

Jan Jürjens†
The Open University, UK
http://www.jurjens.de/jan

Jörg Schreck
O_2 (Germany)
Joerg.Schreck@acm.org

Peter Bartmann
University of Augsburg, Germany
peter.bartmann@wiwi.uni-augsburg.de

ABSTRACT

Mobile communication systems are increasingly used in companies. In order to make these applications secure, the security analysis has to be an integral part of the system design and IT management process for such mobile communication systems. This work presents the experiences and results from the security analysis of a mobile system architecture at a large German telecommunications company, by making use of an approach to Model-based Security Engineering that is based on the UML extension UMLsec. The focus lies on the security mechanisms and security policies of the mobile applications which were analyzed using the UMLsec method and tools. Main results of the paper include a field report on the employment of the UMLsec method in an industrial telecommunications context as well as indications of its benefits and limitations.

Categories and Subject Descriptors: D.2.2 Software Engineering: Design Tools and Techniques -Computer Aided Software Engineering (CASE), D.2.4 Software Engineering: Software/Program Verification

General Terms: Security.

Keywords: Mobile Telecommunication Systems, Security, Model-based Software Engineering, UML, UMLsec.

1. INTRODUCTION

The use of mobile communication technologies has experienced an explosive growth. However, this usage carries critical risks concerning information security that are particularly significant for mobile systems, due both to the inherent vulnerability of the devices and the significant complexity of the architectures (as explained in Sect. 2). In order to address these risks and enable secure mobile communication systems, the security analysis has to be embedded in the development and management of the systems.

This work presents the results of the model-based security analysis of parts of the corporate security architecture and security policies for mobile communication systems at the German telecommunications company O_2 (Germany). The security critical parts of the system were analyzed using UMLsec [4], a UML extension which allows the user to embed security related information into the system design, as well as to conduct security analyses on the model layer. The goal of this work was to gain experiences in the use of the UMLsec method in an industrial telecommunications context and to show its benefits and limitations.

Empirical studies on the use of model-based development in general have so far been limited. The challenges faced by empirical software engineering in general are discussed in [9], and specifically regarding model-based developments in [10]. A report on using UMLsec in industrial development of security-critical software can be found in [1]. Another case study employing a method close to the UMLsec approach in an industrial context is [3], reporting on a project with a major German bank. [12] reports on a student project using a security extension of the AutoFocus tool (that was developed jointly and in parallel with the UMLsec extension) to develop a payment application. [2] reports on a case-study using UMLsec regarding an intranet search machine at BMW. An overview on other applications of model-based security engineering in practice is given in [6]. A case-study on using model-based approach to secure software engineering and management in telecommunication systems (and in particular mobile communication systems) has to the extent of our knowledge not been published so far. Although this paper does not aim to be a complete or controlled empirical study, we hope to contribute to filling this gap using our report on the industrial case-study presented here.

The paper is structured as follows. In Sect. 2 we give some background on the security challenges to software for mobile systems in an industrial telecommunications environment. In Sec. 3, we explain what the goals of the security analysis were, we introduce the UMLsec based analysis methodology. In Sect. 4, we present and discuss the results of the security analysis at O_2 (Germany) at the hand of some representative fragments of the models that were constructed. In Sect. 5, we discuss some lessons learned from the case study. We end with a summary indicating further work.

*Part of the work presented here was performed when the first author was at TU Munich and the third author at O_2 (Germany).

†Partly funded by the Royal Society through an international joint project with TU Munich on model-based security analysis of crypto-protocol implementations.

ICSE'08, May 10–18, 2008, Leipzig, Germany.

2. SOFTWARE SECURITY CHALLENGES IN MOBILE COMMUNICATIONS

We discuss the particular characteristics of mobile communication systems regarding security requirements, and how software engineering and software management processes have to be adapted to the application domain of mobile telecommunication systems in order to deal with the resulting challenges.

2.1 Characteristics of Mobile Security

Mobile networks differ from fixed networks in many respects. We give a rough overview over the most obvious differences with respect to fixed networks which can be found in almost every company. With the term *mobile security*, we refer to all security aspects which are relevant for mobile networks, many of which are also relevant for fixed networks.

2.1.1 Architecture

The most distinguishing aspect of mobile networks is their architecture. Whereas static structures and homogeneous components predominate fixed networks, the situation is usually quite the opposite for mobile networks. In the following we discuss some representative aspects of this which lead to significant complexity:

Different access media.

The type of access medium to the network depends not only on the need of the network's users but also on environmental conditions such as costs or availability. Often, one access medium alone is not sufficient because differences in throughput, latency, cost, and availability justify having alternatives. The access media evolved over time and comply to different standards, such as:
WLAN: IEEE 802.11{a,b,e,g,h,i}, WEP, WPA{1,2}
Bluetooth: IEEE 802.15.1, Version 1.1, 1.2, 2.0
Infrared: IrDA (IrLAP, IrCOMM, IrOBEX, IrLAN)
Telephony: GSM, GPRS, UMTS.
Unfortunately, the different standards for one type of access medium (e.g. WLAN) offer different security features and must thus be considered separately. The standards may be "downwards compatible" and security features may be treated as cumulative. Even within one standard, the security considerations have to consider the different subsets of the standard. This is the case, for instance, with the different *profiles* that can be activated while using Bluetooth.

Variety of end user devices.

Another difference is the variety of end user devices which take part in mobile networks. With respect to security considerations, the most important criteria include:
Form factor: laptops, PDAs, cellphones, smartphones etc.
Operating system: Windows (XP, 2000), Windows Mobile, Symbian OS, Palm OS, Linux, Blackberry etc.

Due to different operating systems and hardware capabilities several security requirements (e.g., separation of duties of different system users) are already fulfilled, can be added to the system, or cannot be fulfilled due to restrictions of the underlying system. This leads to various combinations depending on the set of requirements essential for the given usage scenario. Some of them are described in the following.

Different protection mechanisms.

As described above, different security requirements may be applicable for different usage scenarios. Within those we can identify a subset of requirements which are applicable for almost each usage scenario. The following list shows the most important and gives some explanation why they cannot be treated in general for all possible archtitectures.

Authentication: The required grade of authentication depends on the data and services to be protected. While public information might be accessed without authentication, this is not acceptable for private data or services which are offered to employees only. In addition to the different levels which might require authentication (e.g., the end user device, network connections, internal services) there are also several grades of authentication (e.g., password, one-time passwords, two-factor authentication, biometrical procedures) to choose from which increases the number of possible combinations that have to be checked.

Network security: Mobile devices usually offer very limited means to secure the network layer. A reason for this are their limited resources which often do not allow for transparent encryption (e.g., for a VPN tunnel) and the limited availability of the necessary software for the chosen platform (e.g., firewall, virus scanner).

Application security: Mobile devices which are not always connected cannot be secured like their counterparts within a fixed network. E.g., the provision of patches, update of signatures for virus scanners, and central processing of log file entries can only performed the next time the device is connected. Also well-established security measures are often not feasible on certain handhelds, e.g., due to their low performance (such as an IPS running on a smartphone).

Secrecy: As stated above, the type of encryption of data (e.g., transparent or after usage) depends on the performance of the device. Moreover, the secrecy of the data is also influenced by the physical capabilities of the device. Some are able to use enhanced encryption techniques like smartcards or biometric components while others are not.

Availability: As mentioned, developers of mobile systems often need to design secure architectures for different access media and often also to include different access media into one architecture. More complexity is added when only one security concept must be developed with disregard to the access media - which might change during the usage of the system due to environmental conditions.

2.1.2 Usage

The above-mentioned differences are mainly due to technical restrictions. Beside these restrictions there are also restrictions with origin in the users' behavior when using mobile networks. Some of them are sketched below:

Spontaneous usage: Beside users who take advantage of mobile networks as a replacement of their office desk, there is a high amount of *nomadic users* who have to adapt to different environmental conditions or time constraints with only very limited options several times a day, e.g., travelers at an airport. In contrast to users in an office who are willing to accept (once a day) a fixed procedure to gain (and suspend) access to resources and services, these nomadic users usually do not accept login or logoff procedures which are not negligible with respect to the usage period. Since the usage periods are often quite short, the procedures for preparing the service have to be much shorter. Unfortunately, this often leads to reduced acceptance of full-blown security checks and therefore to a lower security standard.

Limitation of services: On the other hand, the reduced usage period entails also a limited set of services which can be used in it. As a consequence, users accept a reduced

amount of services compared to the ones they are offered in an office environment. Often, it is sufficient to provide access to emails, appointments, and addresses.

Always on / ABC: Users of mobile networks often want to be connected always in order to receive the latest information and to have necessary information at their disposal. Also, they want the best combination between reliability/throughput and price of the access media (*ABC, always best connected*). This shows once again the necessity to change the access medium within a session.

Average useful life: The average useful life of devices used within mobile networks usually is much shorter than that of components within fixed networks. As a rule of thumb, the average useful life should not expected to be longer than 18 months (in contrast to 36 months), although one should note that a hard end-point on this can usually not be defined. This entails that the respective architecture has to be evaluated with each new generation of end devices which leads to the double amount of assessment.

2.2 Implications for Software Engineering and Management

Based on the discussion of the particular characteristics of mobile telecommunication systems in particular wrt. security requirements in the previous section, one can derive the following characteristics that have to be taken into account specifically when designing mobile communication architectures:

- higher variety of access media
- higher variety of devices and operating systems
- less protection mechanims included by default
- spontaneous use of usually less services
- cost of additional software might exceed device cost
- biased compromise between security and usability
- reduced time for amortization and frequent redesign
- only few audits or certifications available.

Taking into account all different possible combinations of the above mentioned criteria we are faced with an amount of several thousands of different (possible) architectures which must be analyzed and optimized for their proper implementation of security requirements in order to choose the most effective, the cheapest, and the most comfortable. Each of these architectures may in itself be quite complex and non-trivial to analyze for the given security requirements.

The discussions above and in the last subsection make clear that the necessity of analyzing all of these potentially quite complex architectural alternatives against the security requirements makes this task only feasible at a satisfactory level of trustworthiness if it is supported by an systematic and efficient process which makes use of automated security analysis tools for the architectural options as far as possible. To perform this task at the given application at O_2 (Germany), we therefore developed such a process, which is explained in the following section and then applied in Sect. 4.

3. SECURITY ANALYSIS OF MOBILE COMMUNICATION ARCHITECTURES

3.1 Requirements on the Security Analysis

The main goal of a security analysis is a satisfactory level of confidence that a given security policy or particular security requirements are fulfilled. We give some further requirements on the security assessment process for mobile communication architectures, motivated by the discussions in the previous section, in particular by the high number of architectural alternatives that may need to be analyzed:

Reproducability: The results need to be reproducable for a given architecture without risk of misinterpretation.

Delegability: It is required that at least parts of the analysis can be delegated to be feasible in practice.

Efficiency: The analysis must be performed in a given time-frame with a defined expectation regarding thoroughness and scope. The necessary amount of work done by a human security expert should be reducable by limiting the scope of the analysis.

Parallelization: It must be possible that parts of the analysis can be performed in parallel and independently.

Traceability: Results of the analysis must be traceable and give guidance how negative results can be improved on.

Expressiveness: The results must carry enough information to enable an overall risk analysis of a given architecture.

To achieve these requirements, we decided to evaluate the use of a security assessment process which includes the use of models related to the given architectures and security requirements, and of automated tools to analyze these models against the given security requirements. To keep the amount of additional training bounded, we chose an approach based on the Unified Modeling Language (UML), and one of the options available here is the security extension UMLsec of the UML (to be introduced in the next subsection).

3.2 Security Analysis using UMLsec

Model-based Security Engineering (MBSE, [4, 5, 8]) provides a soundly based approach for developing security-critical software where recurring security requirements (such as secrecy, integrity, authenticity and others) and security assumptions on the system environment, can be specified either within a UML specification, or within the source code as annotations (cf. Fig. 1a). Various analysis plugins in the associated UMLsec tool framework [11] (Fig. 1b) generate logical formulas formalizing the execution semantics and the annotated security requirements. Automated theorem provers and model checkers automatically establish whether the security requirements hold. If not, a Prolog-based tool automatically generates an attack sequence violating the security requirement which can be examined to determine and remove the weakness. Thus we encapsulate knowledge on prudent security engineering and make it available to developers who may not be security experts. Since the analysis that is performed is too sophisticated to be done manually, it is also valuable to security experts. Part of the MBSE approach is the UML extension UMLsec for secure systems development which allows the evaluation of UML specifications for vulnerabilities using a formal semantics of a simplified fragment of the UML. The UMLsec extension is given in form of a UML profile using the standard UML extension mechanisms. *Stereotypes* are used together with *tags* to formulate the security requirements and assumptions. *Constraints* give criteria that determine whether the requirements are met by the system design, by referring to a precise semantics of the used fragment of UML. The security-relevant information added using stereotypes includes security-relevant information covering the following aspects:

- Security assumptions on the physical system level, for example the stereotype « encrypted », when applied to

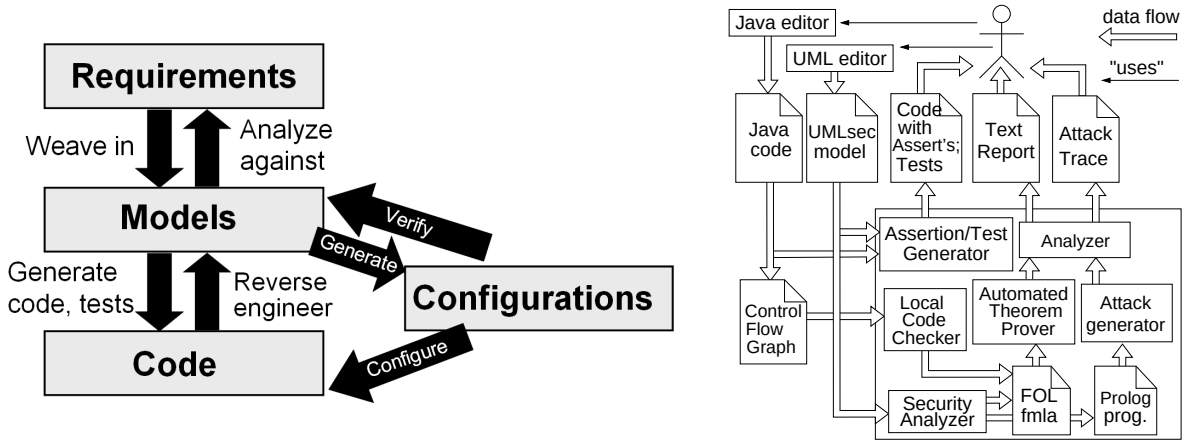

Figure 1: a) Model-based Security Engineering; b) Model-based Security Tool Suite

a link in a UML deployment diagram, states that this connection has to be encrypted.

- Security requirements on the logical level, for example related to the secure handling and communication of data, such as «secrecy» or «integrity».
- Security policies that system parts are required to obey, such as «fair exchange» or «data security» (see Sect. 4 for explanations).

The UMLsec tool-support in Fig. 1b can then be used to check the constraints associated with UMLsec stereotypes mechanically, based on XMI output of the diagrams from the UML drawing tool in use [11, 5]. There is also a framework for implementing verification routines for the constraints associated with the UMLsec stereotypes. Thus advanced users of the UMLsec approach can use this framework to implement verification routines for the constraints of self-defined stereotypes. The semantics for the fragment of UML used for UMLsec is defined in [4] using so-called *UML Machines*, which is a kind of state machine with input/output interfaces and UML-type communication mechanisms. On this basis, important security requirements such as secrecy, integrity, authenticity, and secure information flow are defined.

3.3 Modeling of Security Requirements

Starting from a given set of security requirements (i.e., the company's security policy) which focuses on mobile security, we conducted an analysis to identify the parts of the UMLsec framework which are most suitable to model these requirements and allow for a subsequent security analysis.

The main objectives for the selection of requirements, parts of the frameworks, and modeling objects have been:

- focus on requirements which affect the architecture
- reduction of complexity by abstraction and generalization
- reusability of the model

Therefore, the target of the modeling process and the resulting models are:

- data
- data flows
- network objects and their connections
- services
- protection goals
- protection measures and devices
- processes.

As a result of this modeling process, we develop a model focused on particular parts of a given security policy, which is considered to be complete, consistent, rather stable, and independent of the concrete architecture to be analyzed.

3.4 Analysis Process

In order to support the goals of the security analysis in Sect. 3.1 we developed an analysis process which takes advantage of three different kinds of models. These models are sketched in Fig. 2 and described as follows (see Sect. 4 for examples):

Security Requirements Model: The creation of this model is described in Sect. 3.3. To be able to specify the security requirements, this model may include architectural or behavioral views of the system, but in a prescriptive rather than descriptive way. This model satisfies all relevant security requirements by construction.

Usage Model: The Usage Model is a less formalized model (usually not using UMLsec) which describes the requirements of a target architecture from a user's point of view.

Concretized Model: The Concretized Model describes a concrete architecture by means of UMLsec. It is considered to satisfy the requirements of the Usage Model.

The Concretized Model can be obtained by adapting the architecture-independent Security Requirements Model until it fits the Usage Model with respect to, for instance, required functionality or available protection measures.

Thus, we obtain an architecture-dependent Concretized Model which usually differs from the Security Requirements Model. By making use of different parts of the UMLsec notation and tool-support (see Sect. 4 for examples) we can:

- identify gaps between these two models by applying the framework to the Concretized Model
- document security requirements which are not met by the Concretized Model
- improve the Concretized Model through iterations and reassessment

By following this iterative process we are able to improve a target architecture with respect to security requirements. Due to the restrictions of the Usage Model and the factors described in Sect. 2, it is advisable to follow such an iterative approach in order to keep the protection measures that are

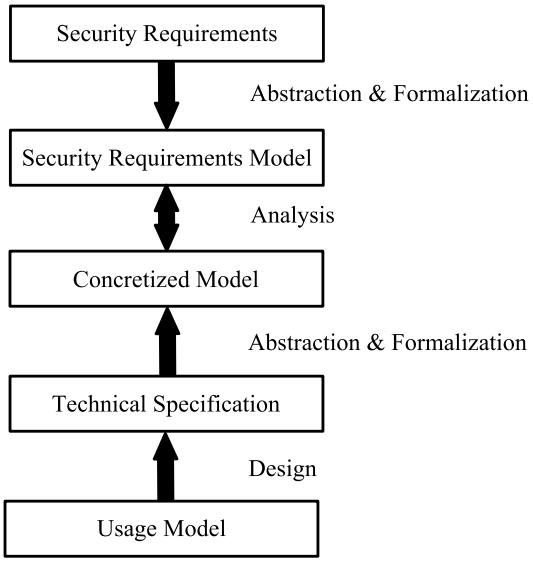

Figure 2: Models used in the Analysis Process

needed to satisfy a given security policy subject to a given usage scenario minimal.

4. APPLICATION AT O_2 (GERMANY)

We now explain how the process for model-based security analysis explained above was used at an application at the German telecommunications company O_2 (Germany). We illustrate this with a few representative examples.

Overall, we extracted 62 security requirements from the security policy which we formalized for developing this model by employing the following stereotypes respectively extensions (see [4] for definitions and explanations):

- «fair exchange» and «provable» for 21 process-related requirements, which are formalized within eight activity diagrams.
- «secure links» for 10 security requirements regarding secrecy and integrity of data which have to meet on the physical layer. All requirements are integrated in one deployment diagram.
- Three requirements concerning role-based access control.
- An extension of UMLsec based on logical formulas for formalizing 15 security requirements concerning network services and dataflows which have to be regulated by firewalls resp. tested for malware by anti-virus software (see Sect. 4.3). The logical formulas which formalize all of these requirements are based on one network architecture model.
- For 13 requirements we were not able to find an appropriate UMLsec representation.

In each of the examples presented in this section, we will consider two kinds of UMLsec models: A model which represents the security requirements that should be realized according to the company security policy (Security Requirements Model) and a model which represents an implementation of the Security Requirements Model, which may already be in place within the company, or which is intended to be implemented (Concretized Model). One can then use the UMLsec tools to compare the Concretized Model with

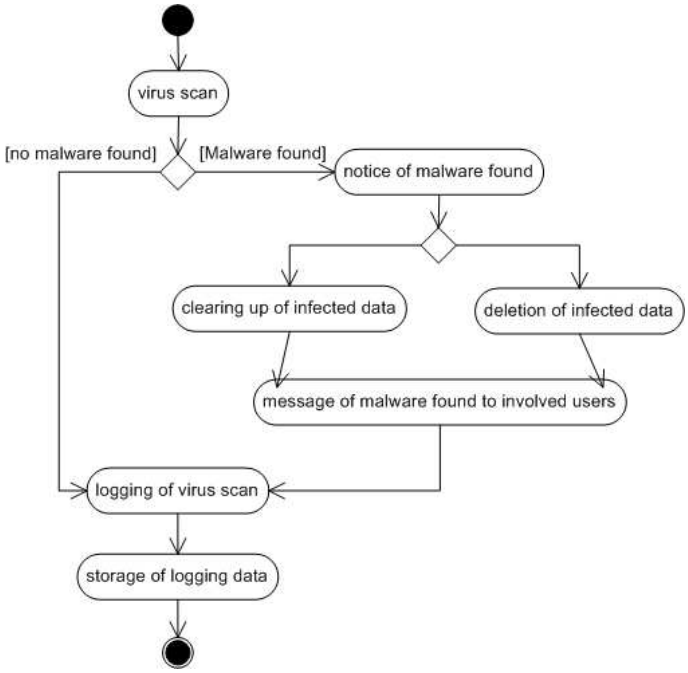

Figure 3: Security Requirements Model for virus scan scenario as a UMLsec activity diagram

the Security Requirements Model to determine whether the Concretized Model enforces all the security requirements that should be in place according to the Security Requirements Model. In that sense, the Security Requirements Model plays the role of a "blueprint" which demonstrably enforces the security requirements given in the company security policy. Against that, the Concretized Model, which represents the existing or planned implementation, has to be compared.

4.1 Scenario 1: Virus Protection

For the first example we consider four security requirements concerning virus scans on a mobile device:

Requirement 1: If malware is found, the infected data has to be cleaned up or to be deleted completely.

Requirement 2: Furthermore, all involved users have to be informed about the malware that was found.

Requirement 3: The results of every virus scan have to be logged.

Requirement 4: A log-file has to be stored on a central storage device.

These requirements are described on a highly abstract level. So, for example, they only formulate that infected data has to be cleared up or to be deleted. They do not specify a detailed way in which these measures have to be accomplished. Thus the requirements listed above can be formalized on a process-related level. Fig. 3 shows the Security Requirements Model representing these security requirements in the notation of a UML activity diagram.

To complete the formalization of the security requirements, the stereotype «fair exchange» is applied to the acticity diagram and the associated tags `{start}` and `{stop}` are defined for each requirement separately:

- Tags for requirement 1:
  `{start}` ="notice of malware found",
  `{stop}` ="clearing up of infected data",

{stop} ="deletion of infected data"

- Tags for requirement 2:
  {start} ="notice of malware found",
  {stop} ="message of malware found to involved users"
- Tags for requirement 3:
  {start} ="virus scan",
  {stop} ="logging of virus scan"
- Tags for requirement 4:
  {start} ="virus scan",
  {stop} ="storage of logging data"

The constraint for the stereotype « fair exchange » requires that whenever a {start} state in the contained activity diagram is reached, a {stop} state will also eventually be reached (for a detailed description of UMLsec see [4]). For example, if the {start} state "notice of malware found" is reached in the process shown in Fig. 3, the {stop} state "clearing up of infected data" or the {stop} state "deletion of infected data" will subsequently be reached. There is no sequence in which the {start} state is reached but afterwards none of the {stop} states (and this can also be checked automatically using the UMLsec tools). Thus the activity diagram combined with the stereotype « fair exchange » and the defined tags fulfill requirement 1, so they provide a Security Requirements Model for the first security requirement. Analogously, it can be shown that the other security requirements hold in the Security Requirements Model as well.

It is important to note that this Security Requirements Model is only reasonable under some assumptions: The Security Requirements Model only requires the existence of the process actions described above and the right embedding of these actions in the whole process, but does not describe how the particular actions itself have to work. So this security model can only be applied to a real architecture in a useful way when it is additionally ensured that these actions (e.g., "clearing up of infected data") work accurately in a real world context. That is, we do not analyze the quality of these actions (e.g., the quality of an encryption algorithm), but their correct usage.

As the next step, a Concretized Model has to be built which can then be compared with the Security Requirements Model developed above. A Concretized Model is always based on a Usage Model as introduced in Sect. 3.4 which describes the software to be applied in the examined infrastructure. In this example, we assume the usage of an anti-virus software which provides functionality for deleting found malware and for logging positive results of a virus scan. Fig. 4 shows a simplified part of such a Concretized Model which visualizes the process of scanning data for malware on a mobile device.

Before analyzing this Concretized Model, the stereotype « fair exchange » and the tags defined for the security model are applied to the associated activity diagram. By using the appropriate analysis plugin "UMLsec Static Check I" in the UMLsec tool framework [11], this Concretized Model can then be tested separately for each security requirement formalized in the Security Requirements Model above. The results of the analysis using the plugin "UMLsec Static Check I" show that the first requirements hold in the Concretized Model: In every possible sequence of the process the state "deletion of infected data" will be reached after the state "notice of malware found" is reached. Analogously, the Concretized Model fulfills the second requirement. In contrast,

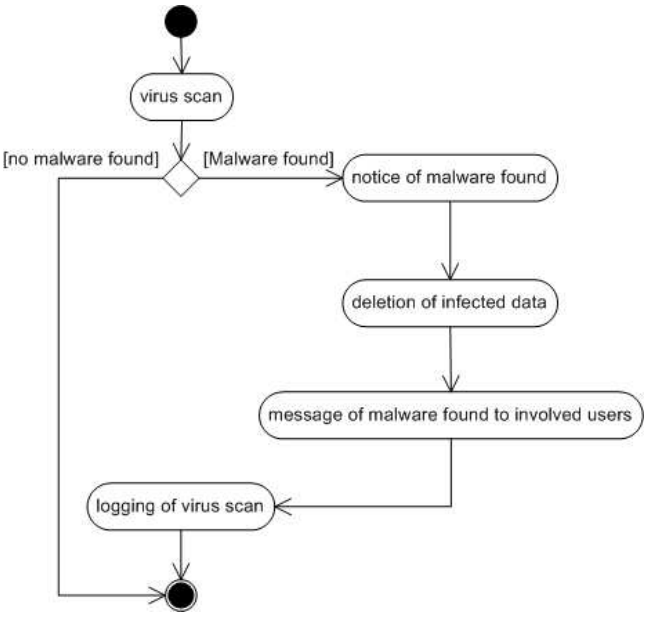

Figure 4: Activity diagram for Concretized Model

the third and fourth requirements do not hold in the Concretized Model. If no malware is found, the process does not reach the state "logging of virus scan", and the state "storage of logging data" is completely missing in the process. Thus the third and fourth security requirements are not met, which can be detected by the process described above. The gaps are displayed explicitely (e.g., by enumerating missing tags) and can be used for a subsequent improvement cycle or a risk analysis.

4.2 Scenario 2: Mobile Communication

We want to develop the second example on the basis of the following exemplary security requirements which refer to the communication between a mobile device and external storage devices respectively the company's intranet:

Requirement 1: All data sent between a mobile device and the intranet is allowed to be read only by employees.

Requirement 2: The data stored on a mobile device is allowed to be stored only encrypted on an external storage device such that only the owner of the related mobile device is able to read the data.

Both items specify the requirement of secrecy of data sent between the mobile device and the intranet or stored on an external storage device like a memory card, which can be achieved by encryption. Since the stereotype « secure links » is used to ensure that security requirements on the communication are met by the physical layer, this stereotype seems to be appropriate to formalize these requirements. Fig. 5 shows a deployment diagram labeled by the stereotype « secure links » and is used as basis of the Security Requirements Model.

Part of the Security Requirements Model is the definition of the attacker types against which the security requirements have to hold. For our example we define two different attacker types, and for each the actions they can apply to various communication links (cf. Tables 1 and 2).

For building an exemplary Concretized Model (which we can test for fulfilling the security requirements listed above) we assume the usage of an encryption software which en-

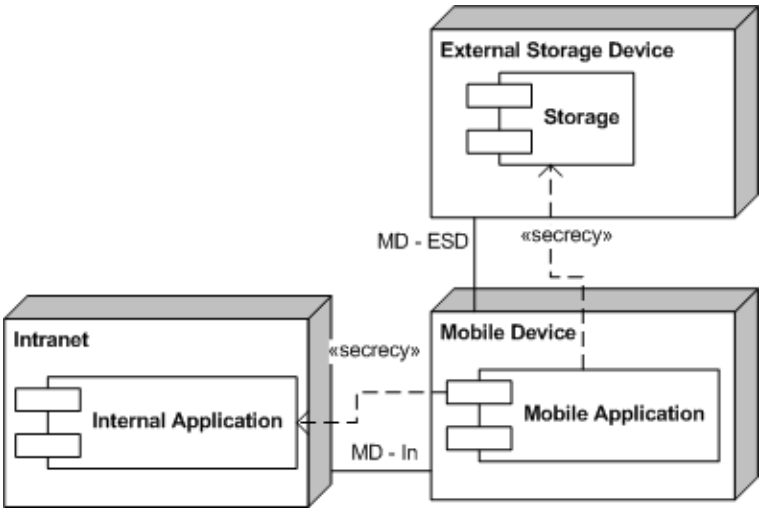

Figure 5: Deployment diagram for Security Requirements Model

crypts data sent between the mobile device and the intranet and data stored on an external storage device with a company-wide key (which every employee is assumed to know). The deployment diagram in Fig. 6 labeled with the stereotype « secure links » contains the relevant subsystem of the Concretized Model.

An analysis of this Concretized Model with the plugin "UMLsec Static Check I" in the UMLsec tool framework shows that the model only satisfies the first security requirement. The constraint associated with the « secure links » enforces that for a dependency with stereotype « secrecy » between two objects on different nodes we have a communication link between these nodes such that a defined attacker type is not able to perform the threat "read" on that communication link. According to the first requirement, an external attacker should not be able to read the data sent on the communication link between the nodes "mobile device" and "intranet". Because this communication link is labeled with the stereotype « encrypted with company key », an external attacker is not able to read any data sent on this link according to the definition of the related attacker type. By contrast, an internal attacker is able to read data sent on this communication link. However, the first requirement does not specify an employee as unauthorized to read this data. Hence the first requirement holds in the architecture model above. Analogously, an external attacker is not able to perform the threat "read" on the communication

communication links	threats
plain	delete, read, insert
encrypted with company key	delete
encrypted with user key	delete

Table 1: Threats from an external attacker

communication links	threats
plain	delete, read, insert
encrypted with company key	delete, read, insert
encrypted with user key	delete

Table 2: Threats from an internal attacker

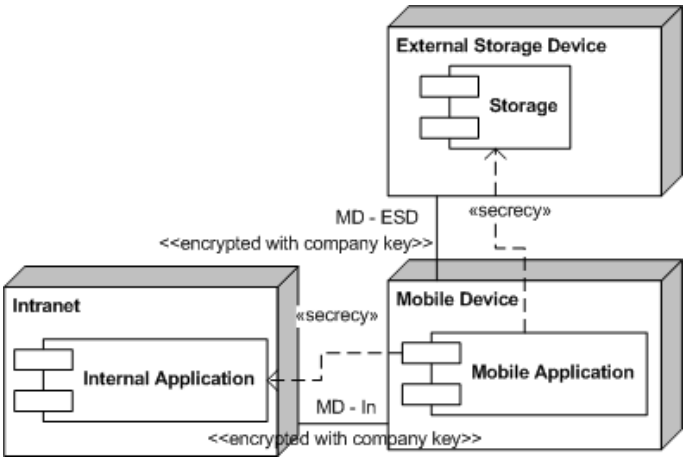

Figure 6: Deployment diagram for Concretized Model

link between the nodes "mobile device" and "external storage device", but an internal attacker is. Because the second requirement specifies that only the owner of the mobile device is allowed to read data stored on external memory, this requirement is violated by the architecture model. A solution could be to use encryption software for encrypting data with an individual key of the user.

4.3 Scenario 3: Network Security Architecture

Within the project, we identified a group of similar security requirements for whose formalization we did not find an appropriate UMLsec notation element and tool plugin. Most of these security requirements specify that there should not exist any network services or dataflows between an insecure component like the Internet and a critical component of the company's intranet like internal server applications. So we developed a method to test a network architecture model for potentially dangerous network services and dataflows making use of automated theorem provers which can be easily integrated into UMLsec. Here, a potentially dangerous network service is understood as a service which is not regulated by a firewall and a potentially dangerous dataflow is understood as a dataflow not scanned for malware. The following exemplary requirements concerning the architecture in which a mobile device can be integrated belong to this group of security requirements:

Requirement 1: Every network service incoming from or outgoing to the Internet must be regulated by a firewall if an internal server application is using this network service.

Requirement 2: All data coming from the Internet must be tested for malware before being received and processed by an internal server application.

Again, one can create a Security Requirements Model formalizing these requirements and an associated Concretized Model that is supposed to describe the implemented security requirements for a given architecture. To be able to verify the Concretized Model for this kind of security requirements, we have developed an approach based on a formalization of the Security Requirements Model and the Concretized Model using first-order logic (FOL). The logical formulas arising from the Concretized Model can then be automatically verified against those arising from the Security Requirements Model using automated theorem provers (ATPs) for

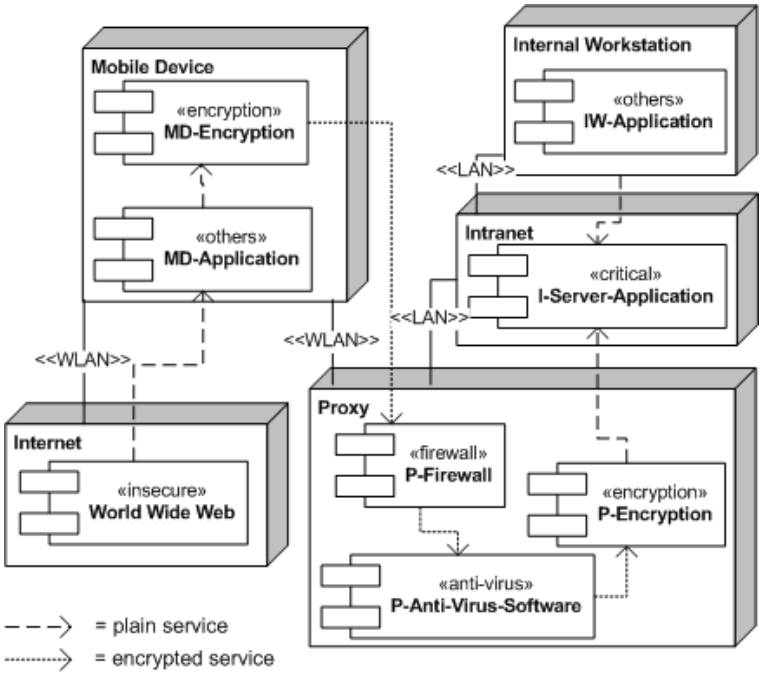

Figure 7: Deployment diagram for Concretized Model

first-order logic, such as SPASS[1].

The set of formulas is composed of three main categories, as explained in the following paragraphs:

Network architecture: The first set of formulas formalizes the network architecture model to be tested. This part formalizes the Concretized Model which has a deployment diagram specifying the network architecture model as its basis. Within this deployment diagram, devices are modeled as nodes which may contain several components (for example software applications). For every node the available access medium is specified, and the nodes can be interconnected by communication links of different types of access media (such as LAN, WLAN, etc.). The components can be labeled with stereotypes describing the type of the component (such as firewall, anti-virus software, encryption software, critical component to be protected, etc). Network services and dataflows are modeled as directed dependencies between components. For each dependency, the protocol (such as HTTP, FTP, IMAP etc.) on which the related network service or dataflow is based is also specified using a stereotype. Additionally, for each component of the type «firewall» and «anti-virus» one needs to define which ports or protocol services they can regulate respectively scan. Fig. 7 shows such a deployment diagram of a simplified network architecture to be analyzed for the security requirements. Fig. 8 shows an exemplary logical formula which defines the component "I-Server-Application" as a critical component of the node "Intranet" and "IW-Application" as component of "Internal Workstation" with the stereotype «others». In addition, this formula specifies that LAN is an active access medium on both nodes ("Intranet" and "Internal Workstation"), that on this access medium an HTTP network service can be established, and that there exists an HTTP network service between the components "I-Server-Application" and "IW-Application".

[1]http://spass.mpi-sb.mpg.de

```
input_formula(network_architecture_model,axiom,(
  is_component_of(internal_workstation,iw-application) &
  is_component_of(intranet,i-server-application) &
  type_of_component(iw-application,others) &
  type_of_component(i-server-application,critical) &
  access_media_availability(internal_workstation,lan) &
  access_media_availability(intranet,lan) &
  service_on_access_media(http,lan) &
  connection(service,iw-application,i-server-application,
    http,plaindata))).
```

Figure 8: Formulas for Network Architecture Model

Threat model: The second set of formulas formalizes the threat model against which the Concretized Model should be analyzed. On the one hand these logical formulas provide the rules for generating possible attacks. For example, if the same access medium standard is active on two different nodes, these two nodes can be connected by a communication link of this type of access medium. Furthermore, all kinds of network services and dataflows which can be established on this communication link are added to the network architecture model between the components of these two nodes, in a way that a network service between two components in one direction always implies a dataflow between these two components in both directions. Fig. 9 shows a logical formula which formulates the rule for generating dataflows between two components in both directions from a directed network service between the same two components. More precisely, the formula formalizes that if there exists a network service of any type of protocol and encryption standard between the component ComponentX and the component ComponentY, there also exist a bidirectional dataflow of the same type of protocol and encryption standard between ComponentX and ComponentY. On the other hand, the formulas describe the rules for marking all potentially dangerous network services and dataflows as well as for connecting these dangerous services and dataflows transitively.

Security requirements: The third set of formulas formalizes the security requirements against which the Concretized Model has to be analyzed (i.e., the Security Requirements Model). For every security requirement a particular predicate has to be formulated. Intuitively, a formula formalizing the first security requirement specifies that there exists no transitive network service from an insecure component (in the example in Fig. 7, the component "World Wide Web") to a critical component ("I-Server-Application") or vice versa not passing a component of the type «firewall». More precisely, because of the way this requirement will be verified by the automated theorem prover (see below), we need to formalize the negation of this requirement (since the requirement is then fulfilled if the prover reports that this formalization is not derivable from the formulas explained above). For example, in the input notation for the FOL theorem provers, the formalization of the first requirement is shown in Fig. 10. Intuitively, this formula formalizes the logical conjecture that there exist an insecure component ComponentX and a critical component ComponentY which are connected by a transitive network service without fire-

```
input_formula(connection_of_service_to_connection_of_dataflow,axiom,(
! [ComponentX,ComponentY,Service,DataEnc] : ( (
  connection(service,ComponentX,ComponentY,Service,DataEnc) ) => (
    connection(dataflow,ComponentX,ComponentY,Service,DataEnc) &
    connection(dataflow,ComponentY,ComponentX,Service,DataEnc))))).
```

Figure 9: Formulas for Dataflow Model

```
input_formula(requirement_1,conjecture,(
? [ComponentX,ComponentY,Service] : (
  connection_without_firewall_regulation
    (dataflow,ComponentX,ComponentY,Service,plaindata)&
  type_of_component(ComponentX,insecure) &
  type_of_component(ComponentY,critical)) )).
```

Figure 10: Formalized Security Properties

wall. If the theorem prover can derive this conjecture from the formulas formalizing the network architecture and the threat model, this means that there is a potential vulnerability. If the theorem prover reports that such a derivation does not exist, there is no such vulnerability. The predicates such as type_of_component() used in this formulas have to be defined in the first set of formulas. The second requirement (malware scan) can be formalized by a similar formula.

To analyze the Concretized Model, the formulas explained above are given as input to the automated theorem prover which then tries to deduce the predicates formalizing the security requirements from the set of formulas describing the architecture model and the rules. This is done completely automatic (i.e. without any human interaction). Also, the formulas can be generated automatically from the Security Requirements Model and the Concretized Model so that the user can use this new verification technique in an automated way as part of the general UMLsec tool-flow (this generation plugin is currently in the development phase). If the theorem prover finds a deduction of one of these predicates, the related security requirement does not hold in the Concretized Model. If the theorem prover reports that no such deduction exists, the security requirement does hold. (In principle, it can happen that the theorem prover does not return a result at all, in which case one cannot draw any conclusions, but this did not happen with the models in the application reported here.)

Assuming that the access medium "LAN" is active on the nodes "Mobile Device" and "Internal Workstation" and an HTTP network service can be established on a LAN communication link, the first security requirement does not hold in the Concretized Model in Fig. 7. For example, these two nodes can be connected by a communication link of the type LAN and hence an HTTP network service can be build on the following path not passing a firewall component: "World Wide Web" -> "MD-Application" -> "IW-Application" -> "I-Server-Application". Thus there exists a network service between the Internet and an internal server application which is not regulated by a firewall. A solution could be to integrate a firewall on the device "Internal Workstation" or to deactivate the access medium "LAN" on this device.

The second requirement is violated in the Concretized Model when assuming, that "P-Anti-Virus-Software" is not able to scan encrypted dataflows for malware. Because the dataflow coming from "MD-Application" over "MD-Encryption" and "P-Firewall" is decrypted by "P-Encryption" after passing "P-Anti-Virus-Software", there exists a dataflow from the Internet ("World Wide Web") to an internal server application without being tested for malware.

5. LESSONS LEARNED

In this section, we provide a discussion of the lessons learned from this application experience at the hand of some guiding questions.

Are there ways in which the application of UMLsec did not go as expected?

Generally, the application of UMLsec worked as expected, though analyzing sophisticated security mechanisms turned out to be less easy than applying simple consistency checks. One main reason is, that depending on the chosen level of abstraction the formalization of security requirements emdedded and described in security policies always entails the reduction of real world complexity and thus loosing some of its context. Hence such a formalization of requirements for developing formal security models and analyzing architecture models only produce reasonable results in consideration of well-formed an sophisticated assumptions. Note that most users will not design their own security mechanisms but instead just want to make sure that they use existing mechanisms correctly.

Did the method have to be changed or adapted to work properly, and if so, in what way? Did UMLsec rules have to be changed to fit the application to mobile communication systems with their specific characteristics as discussed in Sec. 2? Were there security mechanisms in the system that UMLsec does not cover?

Of the 62 security requirements that were considered, 34 could be treated using UMLsec rules without having to be changed or adapted specifically to mobile systems. A further 15 were treated with a new verification technique that allows one to analyse a network architecture as to whether a sufficient number of firewall and virus scan nodes are in place. This analysis can be done automatically using automated theorem provers for first-order logic. This extension to UMLsec has been developed and added to UMLsec for this purpose as part of this project (see Sect. 4 for details). Also, all security mechanisms in the system could be analyzed using the UMLsec notation and the extension mentioned above.

Furthermore, there were some recommendations for desirable reasonable improvements of the analysis plugins in the associated UMLsec tool framework. For example, a lot of plugins are only able to test a UML diagram for only one requirement at one time. But often it turned out to be necessary to formalize more than one requirement in one diagram. Hence it would be desirable to add some functionality to these plugins for testing a diagram for many requirements at one time. Due to the amount of repeated checking tasks we thus enhanced the tool to automatically verify a number of arbitrary tags within one activity diagram.

Did the method yield interesting results?

The method showed that the system under consideration is indeed secure with respect to the security requirements and adversary model that were considered, which is certainly interesting giving the high number of insecure or untested systems. In particular, this seems to be the first time UMLsec has been used in a telecommunications environment (in particular for a mobile communication architecture), which in itself is a new and interesting result.

Did it not pick up issues that you would have hoped it would or should?

We are not aware of any issues that the approach did not notify but which were nevertheless present. In particular, the architecture under analysis was found to provide the desired level of security.

How did its use differ from previous uses?

To our knowledge, this was the first application of UMLsec to mobile telecommunications systems. It differs from previ-

ous applications in so far as the mobility-specific properties of the application had to be taken into account, which was done rather easily using the adversary definition mechanism built into UMLsec.

Can you say anything specific about the security of the application/system now that you have done the modeling?

Using the UMLsec approach, we were able to precisely demonstrate that the main security requirements central to the mobile communication architecture at hand are actually correctly enforced.

How can you be sure you have applied the method correctly or even optimally? Are there other ways in which you could have applied it? Would you use the UMLsec approach again to this kind of system?

We tried to apply the UMLsec method optimally in the sense that we focussed on the security-critical core of the mobile communications system. Since the identification of this part was done informally, there always exist the possibility that security weaknesses may have gone undetected for this reason. However, we believe that this approach is cost-effective in a practical application in industry. In particular, we developed a security assessment process based on UMLsec which seems to be particularly suitable for mobile systems (see Sect. 3.4). Generally, the conclusion was the the use of UMLsec in the given application was successful and it would be worthwhile to apply it again next time to a similar kind of system.

Requirements on Security Analysis.

Coming back to the requirements on the security analysis process that were formulated in Sect. 3.1, we can say that each of them has been reached to a satisfactory degree:

Reproducability: The results turned out to be reproducable for a given architecture with apparently little risk of misinterpretation.

Delegability: Analysis tasks could be delegated.

Efficiency: The analysis process was reasonably efficient and could be scaled down by limiting its scope.

Parallelization: It was possible to perform parts of the analysis in parallel and independently.

Traceability: The results of the analysis could be traced to the architecture and guidance derived how negative results can be improved on.

Expressiveness: The results carried enough information to enable an overall risk analysis of a given architecture.

General Discussion.

The use of UMLsec was generally appropriate for this case study. We were able to start the security analysis based on the given company security policy and proceed down to the technical details of the security analysis of the mobile communication architecture. Some areas for future improvement of the UMLsec method remain. While the definition of some simple (e.g., static security requirements) is quite intuitive for the user, the modeling of more sophisticated (e.g., behavioral or cryptography-related) security aspects turned out to be non-trivial for a user who has only a basic background in security. Future improvements on the usability of the method would therefore be useful, although it may never be achievable to give such a method to someone without any kind of prior knowledge in security. Conclusions from the model-based security analysis need to be drawn carefully, since it is based on the assumption that the actual software implementations that are used are themselves secure. Research on linking model-level security analysis to the implementation level is currently under way [7].

Although this case-study was not aimed at assessing the usefulness of model-based software development techniques in general, it turned out that such techniques do come with an added overhead with respect to training of the user and with respect to effort and time. One may speculate therefore that uptake of model-based software development in industry be fastest for application domains that involve highly sophisticated and critical requirements, such as security-critical systems, since here the effort is most justified (although again, a controlled comparative study on this observation was not part of the scope of the current case-study but would be very interesting future work).

6. SUMMARY

This paper presented a field report on the deployment of the UMLsec method in an industrial context. A model-based security analysis was conducted on a mobile communications system at a major German telecommunications company. The focus was on the application's security mechanisms and policies. Using the UMLsec notation, the user was able to annotate his models with information regarding the security critical aspects of the system in a concise and clear way. Employing the UML profile of UMLsec, developers familiar with the extension mechanisms of the UML should have no problem to learn UMLsec quickly. Furthermore, by embedding the security analysis directly into the IT development and management process, a better understanding and clearer communication of these issues is made possible.

7. REFERENCES

[1] A. Apvrille and M. Pourzandi. Secure software development by example. *IEEE Security & Privacy*, 3(4):10–17, 2005.

[2] B. Best, J. Jürjens, and B. Nuseibeh. Model-based security engineering of distributed information systems using UMLsec. In *29th International Conference on Software Engineering (ICSE 2007)*, pages 581–590. ACM, 2007.

[3] J. Grünbauer, H. Hollmann, J. Jürjens, and G. Wimmel. Modelling and verification of layered security-protocols: A bank application. In *SAFECOMP 2003*, LNCS. Springer, 2003.

[4] J. Jürjens. *Secure Systems Development with UML*. Springer, 2004.

[5] J. Jürjens. Sound methods and effective tools for model-based security engineering with UML. In *27th Int. Conf. on Softw. Engineering (ICSE 2005)*. IEEE, 2005.

[6] J. Jürjens. Model-based security engineering for real. In *14th Intern. Symposium on Formal Methods (FM 2006)*, volume 4085 of *LNCS*, pages 600–606. Springer, 2006. Industry Day Invited Paper.

[7] J. Jürjens. Security analysis of crypto-based Java programs using automated theorem provers. In S. Easterbrook and S. Uchitel, editors, *21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006)*. ACM, 2006.

[8] J. Jürjens and P. Shabalin. Tools for secure systems development with UML. *Intern. Journal on Software Tools for Technology Transfer*, 2007.

[9] D. Perry, A. Porter, and L. Votta. Empirical studies of software engineering: a roadmap. In *ICSE - Future of SE Track*, pages 345–355, 2000.

[10] J. Schalken. Research methods for the empirical assessment of software processes. In *The 12th Doctoral Consortium at CAiSE 05*, 2005.

[11] UMLsec tool, 2001-08. http://computing-research.open.ac.uk/jj/umlsectool.

[12] M. Vetterling, G. Wimmel, and A. Wisspeintner. Secure systems development based on the Common Criteria. In *10th International Symposium on the Foundations of Software Engineering (FSE-10)*, pages 129–138. ACM, 2002.