DRAFT

Full paper published in the proceedings of

22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP 2003)

23-26 September 2003, Edinburgh, UK

An Approach to Trust Case Development

J. Górski[1], A. Jarzębowicz[2], R. Leszczyna[2], J. Miler[2], M. Olszewski[2]

[1]Technical University of Gdańsk, Narutowicza 11/12, 80-952 Gdańsk, Poland
[2]Project IST-DRIVE

Abstract. In the paper we present an approach to the architectural trust case development for DRIVE, the IT infrastructure supporting the processes of drugs distribution and application. The objectives of DRIVE included safer and cheaper drugs distribution and application. A *trust case* represents an argument supporting the trustworthiness of the system. It is decomposed into *claims* that postulate some trust related properties. Claims differ concerning their abstraction level and scope. To express a claim we need a language and a conceptual model. We used UML to represent *claim models* and related *context models* of the trust case. To specify claims we introduced *Claim Definition Language* – CDL. The paper gives a deeper description of the above concepts and illustrates how they were applied in practice.

1. Introduction

As we are becoming more and more dependent on software (both, in the individual and in the group dimensions) there is an increasing need for *software trustworthiness* understood as a guarantee that the trust that the system will meet the most critical expectations (e.g. safety and/or security) is well justified and based on evidence rather than on beliefs. This evidence can include analytical results showing that the system objectives, scope and requirements are adequately identified and understood, probabilistic failure profiles of systems and components, design decisions of which impact on safety and security is known and well understood, results of additional analyses, proofs of conformity to accepted and recommended standards and guidelines etc.

The concept of a *trust case* refers to the need of providing a complete and explicit argument justifying trust in the computer system being used in a given application context. As the trust case encompasses both, safety and security guarantees, it could be (conceptually) split into two parts: the safety case and the security case.

Safety case is a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment. It is recommended that safety case is being developed in parallel with the design. The standard [1] stresses that "the Safety Case should be initiated at the earliest possible stage in the Safety Program so that hazards are identified and dealt with while the opportunities for their exclusion exist". The structure of safety cases has been examined by some EU sponsored projects, e.g. [2].

Much work has been done on security evaluation of products and systems. Two emerging standards seem to be especially influential: [6] for security management and [5], [4] for security evaluation of products and systems. A security case is the counterpart of the safety case and collects the evidence justifying the trust in that the system is sufficiently secure.

Safety and security are two different (although not disjoint) system qualities. If both matter in a given situation (as it is true for medical applications) they can sometimes be in conflict. As an example take a patient related data that should have its access controlled and restricted due to privacy considerations (a security related requirement) and simultaneously should have high availability with relaxed access control in emergency situations (a safety related requirement). In such applications the trust case should cover both, safety and security viewpoints and in addition it should consider possible conflicts and their resolutions.

The IST-DRIVE project, performed within the 5th FP, focused at medical care, undoubtedly a trust related domain. In DRIVE both safety and security mattered and consequently both aspects had to be adequately covered. We used a more neutral concept of trust in order to start with a single target and then to specialize it into security and safety targets.

In the subsequent sections we present the trust case conceptual structure that we adopted in our approach, the language used to specify the trust case structure and contents and then the experiences from applying those concepts to analyze the trustworthiness of a complex IT infrastructure developed by DRIVE. In conclusions we summarize our contribution (as compared to other approaches) and present plans for future research.

2. Project DRIVE – Objectives and Scope

The objective of the 5th Framework R&D project called DRIVE - DRug In Virtual Enterprise (IST-1999-12040) was to create a safer and smarter hospital environment by means of innovative and trustworthy IT solutions. The project involved eight partners from Italy, Sweden, Spain, France and Poland representing hospitals, research institutions, software companies, pharmaceutical companies and hospital equipment manufacturers. The strategic objective of DRIVE was to improve the quality of patient care and patient safety by reducing the drug administration errors while simultaneously reducing the supply chain costs. The project resulted in an integrated IT solution (hereafter called *DRIVE solution*) supporting the drug distribution, administration and application processes from pharmaceutical companies through warehouses and pharmacies down to a patient in a hospital. A pilot version of the DRIVE infrastructure was installed and is in operation in Milan, Italy.

DRIVE solution focuses on three key areas of healthcare optimization:

- Clinical process: representing the drug related processes within a hospital directly or indirectly aimed at the improvement of patient's health state, from the admission until discharge.
- Supply chain: representing the flows of the pharmaceutical products and related information, from the manufacturer to the point of use (patient's bedside).
- Trust: representing the stakeholder requirements for the protection of critical assets: personal clinical records, hospital professional accountability, enterprise value chain and privacy within the entire business model.

The structure of the clinical and supply chain processes is shown in Fig. 1.

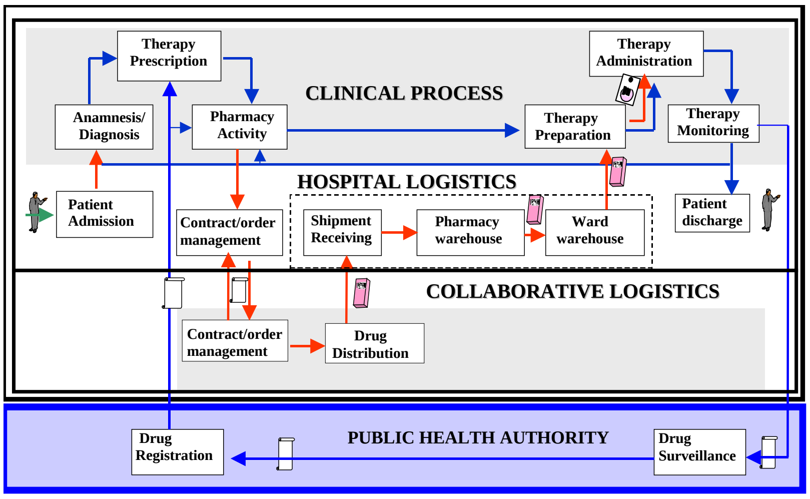

Fig. 1. The scope of DRIVE Solution (Note: the part marked as "Public Health Authority" is not covered by DRIVE)

All identification data is captured in a 2D coded wristband worn by a patient from admission to discharge. DRIVE solution supports the physician during the anamnesis, diagnosis and therapy prescription phases. It also supports nurses in the preparing and administrating therapies, providing for avoiding human errors that often occur during these phases.

The hospital logistics part of DRIVE solution supports the head nurses in maintaining the proper stock of pharmaceutical products in ward cabinets. A new 2D label, specially studied for DRIVE, encodes all the information needed for tracing the drug flows (product code, expiration date, lot number, serial number).

The collaborative logistics part of the DRIVE solution provides for sharing of logistics information between hospitals and pharmaceutical companies through a web based infrastructure. It provides for automated exchange of price lists, orders, order confirmations, advanced shipping notes and invoices.

A number of design decisions were related to trust, e.g. drug and wristband labels to uniquely identify drugs and patients, smart cards and PIN codes to identify and authenticate healthcare professionals, digital signatures to protect integrity of important assets and to provide for non-repudiation, role-based access control etc. The need of providing the *trust case*, an integrated overall argument explicitly stating the trust objectives and linking them with the supporting evidence was recognized later in the project course. It resulted in extending the project scope by adding an additional work package devoted to the trust case development.

3. Trust Case Conceptual Structure

A trust case is developed by making an explicit set of claims about the system and then collecting and producing supporting evidence and developing a structured argument that the evidence justifies the claims. A conceptual UML model describing the trust case elements and showing their relationships is shown in Fig. 2 (the arrows show the direction of reading the association names). The conceptual model of our trust case is based on the results of SHIP [2].

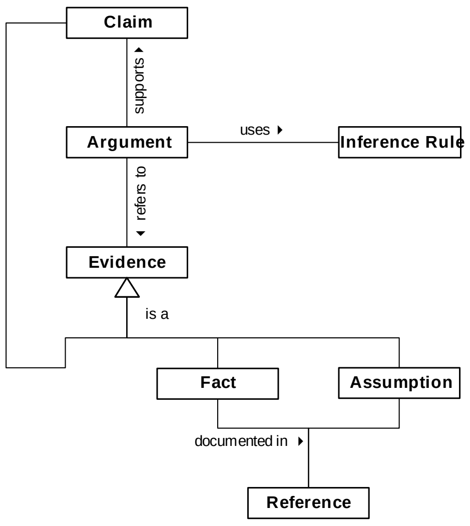

Fig. 2. The conceptual model of a trust case

The evidence used to support an argument can be of the following type:

- Facts, e.g. demonstrating adherence to established principles, documented design decisions, results of performed analyses.
- Assumptions, which are used by the argument and do not require explicit justification (nevertheless, they can be later converted into claims and supported by further argumentation).
- Sub-claims, that are developed further down by giving arguments that support them).

Facts and assumptions are linked to references (documents, reports, data, models, etc.) maintained together with the trust case.

The inference rules used in arguments are of three basic types: *quantitative* (e.g. justifying the failure probability postulated by a claim from the probabilities given by the supporting evidence), *logic* (establishing the validity of a claim from the logical assertions given by the supporting evidence) and *qualitative* (establishing the validity of a claim by referring to the common acceptance of good practices that were adhered to as demonstrated by the evidence supporting the claim). The qualitative argumentation can in particular refer to accepted standards, guidelines or so called "engineering judgement". The nature of software faults differs from the nature of hardware faults in that software is not subjected to physical degradation through aging, vibration, humidity, dust etc. Software faults are design faults (caused by humans) and their statistical properties are very hard to quantify. Therefore, for software intensive systems, the role of the logical and qualitative arguments is increased at the expense of the probabilistic arguments.

Trust case develops down into a tree-like structure (or multi tree, in case we have multiple trust targets) of arbitrary depth and is completed when all the leaves represent (well documented) facts and assumptions.

4. Modeling Claims

To define claims we need a language. A natural language (e.g. English) is the first choice, but due to its obvious limitations (the lack of precision and possible ambiguities), the natural language expressions can sometimes be misinterpreted. Another problem is that while specifying claims in a natural language it is difficult to control the scope and it is easy to mix the levels of abstraction a given claim refers to that adversely affects the soundness of the supporting argumentation.

To overcome those difficulties we decided to control the language associated with a claim by introducing what we call the Claim Definition Language – CDL. CDL introduces the following means that help the analyst to be more precise and unambiguous while building a trust case:

- A graphical language that provides constructs to represent claims, arguments, facts and assumptions and a labeling scheme that helps with their unambiguous identification. The identifier indicates the position of a given construct within the overall trust case. Each claim is associated with its *claim model* presenting a given claim, its supporting argument and all the evidence the argument refers to in a direct way.
- A graphical language that provides for representing, for each claim, its associated *context model* showing all the (physical and logical) objects that are referred to in the claim together with their relationships. To provide for avoiding possible ambiguities we also made some steps towards formalization of the meaning of the relationships occurring in context models.

The constructs used to define claim models are shown in Fig. 3.

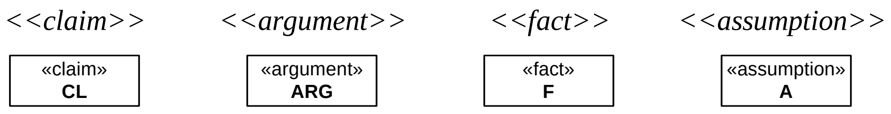

Fig. 3. Elements of the claim model

All the elements of the claim model are defined as UML stereotypes. This helped us in using a UML-conscious graphical tool ([9] in our case) to maintain the trust case. An example of a claim model is shown in Fig. 4.

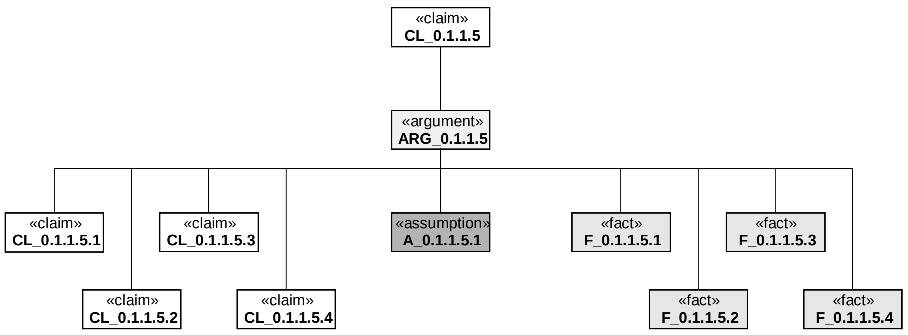

Fig. 4. Example claim model

Different shadows are used to distinguish elements of different types (claims, facts, assumptions, arguments). Each model element has its unique identifier that consists of its type identifier ('CL' for claim, 'ARG' for argument, 'A' for assumption, 'F' for fact) and a numeric identifier. The numeric identifier represents the element's position within the whole trust case. The example given in Fig. 4 represents the model of the claim CL_0.1.1.5. Such higher-level claim is called a *super-claim* and the claims that are referred to in the associated argument are called *sub-claims*. The argument has the same number as the claim it supports. The identifying numbers of sub-claims, assumptions and facts supporting the super-claim are constructed in such a way that they use the super-claim's numeric identifier as the prefix extended by the ordering number of this element (from left to the right) separated by a dot. For instance, CL_0.1.1.5.3 is the third sub-claim of super-claim CL_0.1.1.5 and F_0.1.1.5.2 is the second fact supporting CL_0.1.1.5.

In general, claims are structured in two dimensions: vertically (refined claims used by the arguments justifying the higher level claims) and horizontally (complementary claims that together are used in the same argument justifying a higher level claim).

5. Modeling Claim Contexts

We defined a graphical language that provides means to grasp and represent the context within which a given claim model is being interpreted. The model is expressed in terms of UML. Object orientation and UML are becoming de-facto standard in software development. One of the advantages is that they constitute conceptual and linguistic means that can be applied at the system as well as at the software levels, providing for bridging the gap between the two views. The recent attempts to extend UML to business processes (e.g. [7]) provide for covering both, the artifact being developed (the computer system and its software) and the target environment within which it is being used. Such uniform framework can be applied not only to model the system and its target environment but also to model other important processes the trust depends on, like the development process, maintenance, installation etc. Our claim context models adhere to the common UML style and therefore can be applied to represent a broad range of business processes that can be expressed in UML. This puts the analytical task of trust case development within a uniform modeling framework.

In addition to the standard modeling means offered by UML we introduced a set of class stereotypes to represent typical objects that occurred while building our claim context models. They were especially useful while we were working with the higher level claims that were not supported by the models already developed during the system construction. Those extensions are shown in Fig. 5.

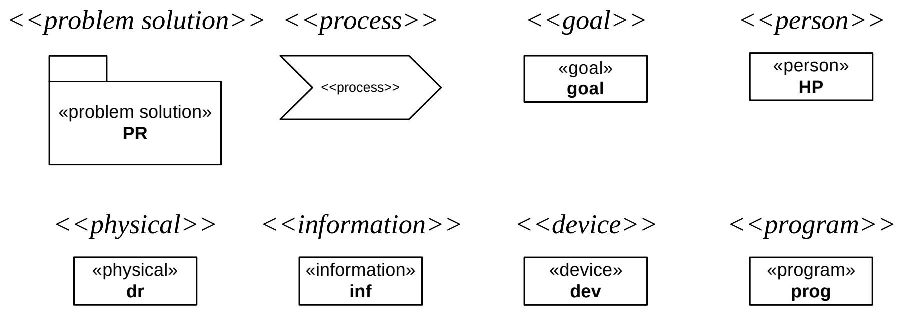

Fig. 5. Stereotypes used in claim context models

An example claim context model [8] is given in Fig. 6. The context model has been derived from the documentation resulting from the DRIVE development process and shows the relations between patients, drugs, their physical identifiers and the corresponding logical entities. The model presents the object classes and their relationships that provide the conceptual and linguistic context used while expressing the following claim:

CL_0.1.1.6
Drug Labels of the Drugs ***applied*** *to Patient are* ***consistent*** *with the Prescription Data in the* ***corresponding*** *Patient Record* ***identified*** *by the Patient's Wristband Label.*

Note that the context model in Fig. 6 includes all the objects referred to in the corresponding claim. The words given in bold in the claim definition distinguish those relationships between the objects of the claim context model the claim refers to. If those relationships are not present in the claim context model, they have to be interpreted in terms of the relationships given explicitly.

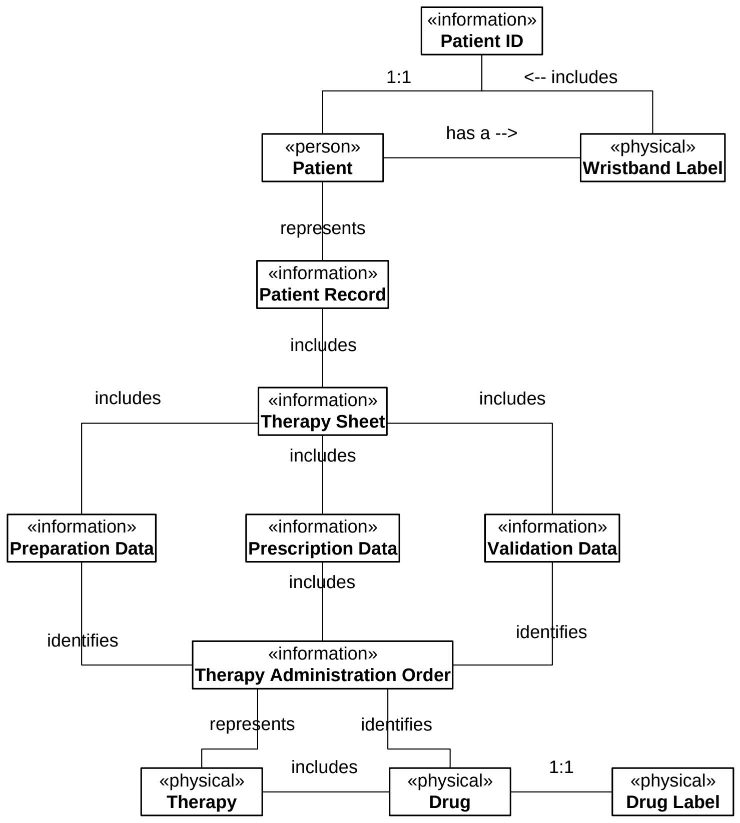

Fig. 6. Example of a claim context model

Although the objects referred to in the claim are precisely specified in the corresponding claim context model, the relationships the claim refers to can lead to ambiguous interpretations, e.g. the meaning of: "Drug *applied* to a Patient". This problem can be mitigated by formalizing the language.

6. Formalizing the Language

Claims, arguments, assumptions and facts are specified in a natural language referring to the elements introduced by the associated claim context model. However, such natural language specifications can sometimes give rise to ambiguous interpretations. To provide for unambiguous interpretation of claims we made steps towards formalizing the meaning of the basic operations that can be performed on objects and the relationships among the objects represented in a claim context model. Then we followed a convention that to help to resolve possible misinterpretations, for each claim, its natural language specification is supplemented its definition given in the formalized language. Such a formalized specification of claim CL_0.1.1.6 is given below:

If
 d:**Drug**<*physical*> is applied to p:**Patient**<*person*>
  then exist dl:**Drug Label**<*physical&information*>, tao:**Therapy Administration Order**<*information*>, pd:**Prescription Data**<*information*>, ts:**Therapy Sheet**<*information*>, pr:**Patient Record**<*information*>, pwl:**Patient Wristband Label**<*physical&information*>, pr:**Patient Record**<*information*> such that
   d(1:1)dl and pwl(1:1)p and tao(includes)pd(includes)ts(includes)pr and pr(represents)p and tao(identifies)dl and pwl(uniquely identifies)pr

This specification imposes restrictions on each object set being an incarnation of the context model shown in Fig. 6. It explicitly refers to objects of the classes given in the corresponding context model, specifies their stereotypes (in brackets and italics) and refers to the terms (underlined in the above text) that are explicitly defined in the *CDL Dictionary*.

Below we recall from the CDL Dictionary the definitions of those terms that occur in the above formalized specification of CL_0.1.1.6 (the symbol "==" stands for "is defined as").

Resource1 is applied to **Resource2** == **Resource1** is consumed and its **State** is used to update **State** of **Resource2**

Resource1 identifies **Resource2** == **Identity** of **Resource2** can be derived from **Attributes** of **Resource1**

Resource1 uniquely identifies **Resource2** == **Resource1** identifies **Resource2** and if exists **Resource3** such that Class(**Resource2**) = Class(**Resource3**) then **Resource3** = **Resource2**

consume== to **read** the *Resource* **State** and to **delete** the *Resource*

update== to **read** and **write** the *Resource* **State**

7. Trust Case Targets for DRIVE

The trust case for DRIVE was developed in a top-down manner. We started with the top level claim CL_0 that postulates that the DRIVE Solution is trustworthy as shown in Fig. 7.

CL_0
The DRIVE solution is trustworthy in its intended context

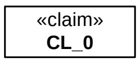

Fig. 7. Top level claim model (the model for the whole Trust Case)

CL_0 was then decomposed into more specific trust targets which resulted in the claim model shown in Fig. 8.

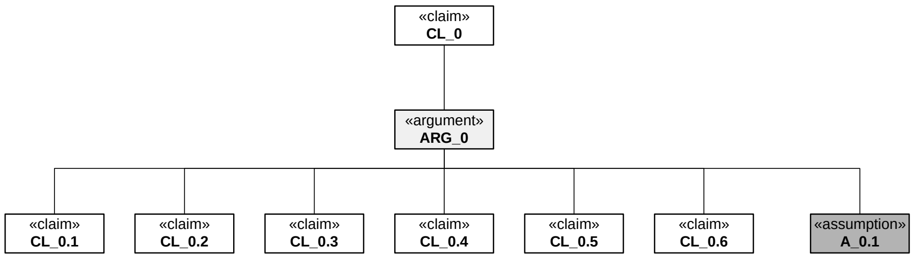

Fig. 8. Claim model for CL_0

The associated claim context model is shown in Fig. 9.

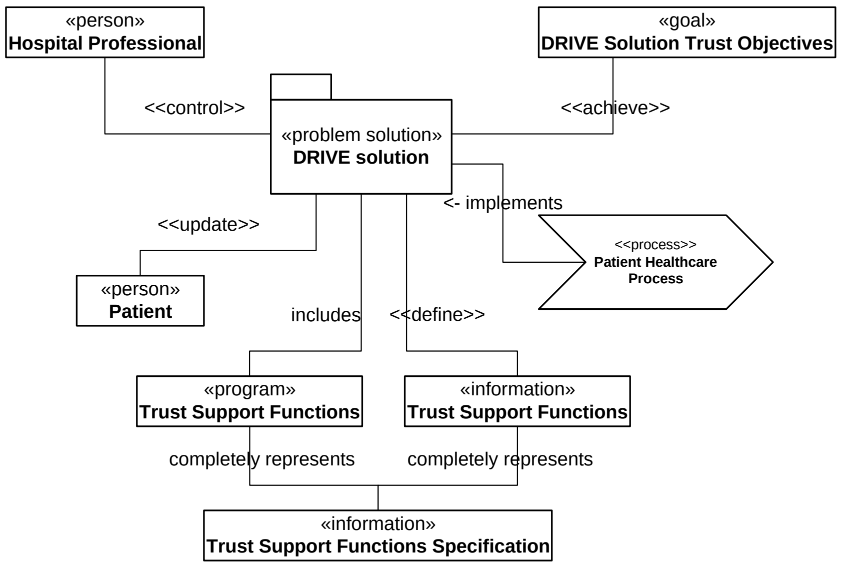

Fig. 9. Refined context model

The context introduces the following entities:

- *Hospital Professional* – Hospital Professional involved in DRIVE Solution,
- *Patient* – Patient subjected to the drug application resulting from DRIVE Solution,
- *Patient Healthcare Process* – a process implementing DRIVE Solution,
- *DRIVE Solution Trust Objectives* - a set of objectives of DRIVE Solution concerning safety and security as perceived by DRIVE Solution stakeholders,
- *Trust Support Functions* – set of DRIVE Solution functions to support safety and security of stakeholders,
- *Trust Support Functions Specification* – functional and design specification of Trust Support Functions.

The argument associated with CL_0 has the form:

ARG_0
Based on the A_0.1 if the claims CL_0.1 to CL_0.5 are justified and in addition are contradiction-free (CL_0.6), CL_0 is justified as well.

and is based on the following assumption:

A_0.1
The analysis of trust objectives in DRIVE is complete and correct.

Note: This assumption reflects the fact that the trust objectives for DRIVE solution were predefined by the DRIVE project and no further analyses were performed to identify their completeness and validity (and consequently there was no further evidence available). Nevertheless, A_0.1 reflects this explicitly in the trust case and could be later converted into a claim, provided further evidence were produced e.g. by performing additional analyses.

The claim model includes the following claims:

CL_0.1
DRIVE solution maintains Patient safety.

CL_0.2
DRIVE solution maintains privacy of Hospital Professional.

CL_0.3
DRIVE solution maintains Patient privacy.

CL_0.4
DRIVE solution ensures that all actions related to Patient health are accountable.

CL_0.5

Trust support functions used in DRIVE are reliable.

CL_0.6
The claims CL_0.1 – CL_0.5 are mutually consistent.

Note: This claim recognizes the need for additional argumentation that the remaining claims do not contradict each other. An example of such conflict could be if the provisions for ensuring patient privacy compromise his/her safety, e.g. by restricting access to patient health data in case of emergencies.

8. Architectural Trust Case for DRIVE

The trust targets were then further decomposed in the search for the supporting evidence. As the decomposition process progressed, we arrived at more specific claims referring to the architectural components, such as particular workstations with associated services and particular functionalities representing business logic rules. Some of the lowest level claims postulated a reliable implementation of a component of the architecture. Those claims were finally closed by converting them into assumptions (we did not have access to the evidence supporting them).

There were two main sources of the evidence referred to in the trust case: the DRIVE design documentation and the information derived from the DRIVE design team by means of interviews. Some additional evidence was obtained by applying UML-HAZOP analytical method [10] to selected design documents of DRIVE [11]. The results of those analyses were then included in the trust case as additional facts. We were also able to discover a number of (sometimes hidden) assumptions that had been made during the design process bringing them explicitly to the trust case.

All claims of the trust case are specified in a natural language and supplemented with their formalized specification. This proved to be very useful while reading and reviewing the trust case as the natural language specifications suffer the obvious limitations concerning their unambiguous interpretation.

The trust case graphical structure including claim models and claim context models are maintained in the tool, Microsoft Visio [9].

The data summarized in Table 1 gives the overview of the present DRIVE trust case size (the number of arguments differs from the number of claims because due to time limitations not all trust case targets were addressed in a complete way).

Table 1. Statistics of Trust Case for DRIVE

Trust case element's name	Number of elements used
Claim	97
Argument	92
Fact	234
Assumption	121

The relatively large number of assumptions present in the trust case reflects that in many cases we could not find enough facts supporting claims and instead had to make assumptions.

9. Conclusions

The paper presented an approach used while developing a trust case for a complex IT infrastructure supporting drug distribution and application processes. The primary decision in our approach was that we have chosen UML as the basic modeling framework for the trust case itself and for both, the subject domain and the solution domain under consideration. The concept of *claim context model* provided for controlling the scope and the language associated with a claim and made the trust case easy to communicate. UML with its mechanism of stereotypes proved to be a flexible and powerful tool to capture the contexts, especially those related to the higher level claims where the model had to cover a considerably broad scope, including people and physical objects involved as well as the pharmaceutical rules and knowledge. Those higher-level models did not exist before and their development was a part of our work on the trust case. The models were very useful in controlling the scope and providing the terminology for corresponding claims. What is even more important, they were very helpful in identifying (sometimes hidden) assumptions that conditioned the validity of some higher level claims. As an example take:

A_0.1.1.1
Hospital Professionals are fully qualified and their intentions are consistent with Pharmaceutical Knowledge and Patient's health state.

Without this assumption we could not argue that the DRIVE infrastructure supports patient's safety as the intentionally wrong drug prescription generally could not be prevented. To express this assumption, however, we needed to refer to the concept of Pharmaceutical Knowledge that was introduced in the corresponding context model.

We observed that when the trust case developed down into more specific claims we could use in the related context models the models that were already developed during the design process. This way the analytical work smoothly incorporated the results of the analyses already performed during system design.

Another important contribution of our approach was a step towards formalization of our claim definition language (CDL). Having all the trust case elements specified formally (in addition to the natural language specifications) was very useful during communication within the team (different parts of the trust case were developed in parallel and then the results were merged during the *composition meetings*) and was confirmed during an independent review of the whole trust case.

Having UML as the underlying framework, using claim context models and formalizing the Claim Definition Language are, in our opinion, the main contributions of our approach while comparing with the work of others [12,13,14].

Another aspect where our work differs from what was presented in the literature is the scope of the analytical work. In our work we concentrated on the notion of *trust* which covered both safety and security (privacy, accountability) aspects. The resulting trust case brings into the surface all the facts and assumptions that support the argumentation for the trustworthiness of the analyzed target. It forms a sort of a map that shows strong and weak points of this argumentation. In our future work we consider using different colors to distinguish the arguments of different strength to provide for easier analysis.

Our work was restricted in two aspects. Firstly, we concentrated our attention only on the DRIVE architecture and its usage context without taking into account other aspects that can adversely affect trustworthiness, like development process, maintenance, installation etc. The reason was that DRIVE was a R&D project with main focus on system architecture and design and did not generate enough evidence to cover the other aspects. Consequently, our trust case is conditional and based on the assumptions that the trustworthiness of DRIVE solution is not compromised by those additional aspects. The second restriction was that the work on the trust case started late in the project, after the product was already in the implementation phase. Due to this fact, our work was a sort of *a posteriori* analysis without much influence on the system design process.

We have also performed some additional analyses aiming at producing more evidence for our trust case. This was done with the help of the UML-HAZOP method [10] and a related tool [11].

Visio 2002 [9] proved to be a useful tool in maintaining the trust case and helping with performing some simple consistency checks. The configuration management related to our trust case comprised of: claim models and claim context models maintained in the tool [9], CDL specification, claim, argument, assumption and fact specifications and the references (derived from the DRIVE documentation) maintained in files.

We consider our approach promising and plan to continue this work in both, research and experimentation. Some of the issues to be addressed include: broadening the scope of the trust case, investigating possible feedback from the trust case to the development process, addressing internal consistency of the trust case in case of potentially conflicting trust targets, getting deeper insight of the differences in the strength of the arguments, depending on the "weight" of the supporting evidence and the nature of the argument itself. We also want to investigate the possibility of better tool support and possible merging of our approach with the already existing frameworks for safety cases, e.g. those embedded in [15].

References

[1] Defence Standard 00-56, http://wheelie.tees.ac.uk/hazop/html/56.htm
[2] EU EUREKA SHIP (Safety of Hazardous Industrial Processes) Project http://www.csr.city.ac.uk/csr_city/projects/ship/ship.html

[3] Safety Case Assessment Criteria http://www.hse.gov.uk/railway/criteria/
[4] Common Methodology for Information Technology Security Evaluation, version 1.0, 1999
[5] Common Criteria for Information Technology Security Evaluation version 2.1, 1999 (Parts 1,2,3)
[6] ISO/IEC Information Security Management, 2000
[7] Eriksson, H.-E., Penker, M.: Business Modeling with UML, J. Wiley, 2000
[8] DRIVE D11.1-3 –Trust Case for DRIVE, D11.1-3, version 1.1, January 2003
[9] Microsoft Visio 2002 Professional, 2002
[10] Górski J., Jarzębowicz A.: Detecting defects in object-oriented diagrams using UML-HAZOP, Found. of Comp. and Decision Sciences, vol. 27, no. 4, 2002
[11] DRIVE D11.4 – UML-HAZOP, D11.4, version 1.1, January 2003
[12] Wilson, S. P., Kelly T. P., McDermid J. A.: Safety Case Development: Current Practice, Future Prospects
[13] Adelard Safety Case Development Manual, Adelard, 1998
[14] Kelly, T.: Arguing Safety A Systematic Approach to Managing Safety Cases (1998). PhD Thesis, University of York, UK, YCST 99/05, 1998, available at http://www.cs.york.ac.uk/ftpdir/reports/YCST-99-05.ps.gz
[15] ASCE (Adelard Safety Case Editor) homepage http://www.adelard.com/software/asce