Tool-Supported Risk Modeling and Analysis of Evolving Critical Infrastructures

Fredrik Seehusen and Bjørnar Solhaug

SINTEF ICT
{fredrik.seehusen,bjornar.solhaug}@sintef.no

Abstract. Risk management is coordinated activities to direct and control an organization with regard to risk, and includes the identification, analysis and mitigation of unacceptable risks. For critical infrastructures consisting of interdependent systems, risk analysis and mitigation is challenging because the overall risk picture can be strongly affected by changes in only a few of the systems. In order to continuously manage risks and maintain an adequate level of protection, there is a need to continuously maintain the validity of risk models while systems change and evolve. This paper presents a risk analysis tool that supports the modeling and analysis of changing and evolving risks. The tool supports the traceability of system changes to risk models, as well as the explicit modeling of the impact on the risk picture. The tool, as well as the underlying risk analysis method, is exemplified and validated in the domain of air traffic management.

Keywords: Risk analysis, interdependencies, critical infrastructures, ATM.

1 Introduction

Critical infrastructures is a term that is commonly used to refer to assets and facilities that are highly essential for the functioning of a society. Such infrastructures include, for example, electricity and power generation and supply, gas and oil production and distribution, telecommunication, public health, and public security and emergency services. Clearly, the disruption of services that are provided by these infrastructures can be severe or even catastrophic. Critical infrastructures are therefore the subject of the strongest requirements to protection from hazards and risks. They are moreover often characterized by strong interdependencies, which means that the safety, security and reliability of some critical infrastructures strongly rely on such properties of several other.

Governments are both nationally and internationally taking measures to safeguard critical infrastructures. For example, at EU level the European Programme for Critical Infrastructure Protection (EPCIP) was created as a result of the European Commission's Directive EU COM(2006) [5]. The objective is to maintain a list of critical infrastructures across the EU, including the set of assets for which protection and incident preparedness is required. The directive stresses the strong interdependencies and the need to identify these in order to achieve

G. Quirchmayr et al. (Eds.): CD-ARES 2012, LNCS 7465, pp. 562–577, 2012.

the EPCIP objectives. It moreover prescribes risk management and risk assessment as necessary means to identify and document threats, vulnerabilities, risks and countermeasures. Clearly, such risk assessments need to systematically take into account the identified infrastructure interdependencies [3,9].

Risk management is coordinated activities to direct and control an organization or a system with regard to risk, and includes the identification, analysis and mitigation of unacceptable risks [8]. For critical infrastructures, risk analysis is particularly challenging because the overall risk picture can be strongly affected by changes in only a few of the systems; in order to continuously manage risks and maintain an adequate level of protection, there is a need to continuously maintain the validity of risk models while systems change and evolve. For this purpose the risk analysis process should be supported by methods and tools that cope with interdependencies in a systematic way, allowing traceability of changes from system to risk. However, established risk management guidelines, standards and methods [1,2,8,11,14,16] largely view systems in a monolithic way and provide little support for handling change [10,12].

This paper presents a risk analysis tool that supports the modeling and analysis of changing and evolving risks. The tool supports traceability of system changes to risk models, as well as the explicit modeling and assessment of the impact of the changes on the overall risk picture. The tool is developed to support the method and the risk modeling language presented in [12], and has been validated in the air traffic management (ATM) domain [17]. ATM systems are critical infrastructures with strong dependencies on other critical infrastructures such as communication, electricity and energy, positioning and satellite systems, transportation systems and emergency. Moreover, interdependencies in ATM are becoming even more relevant and critical with the ATM 2000+ Strategy [6] and the SESAR initiative (http://www.sesarju.eu/) as traditionally quite closed systems at national level are integrated at transnational, European level.

The structure of the paper is as follows. In Section 2 we give some background to risk analysis in general and to the risk analysis method and language that the tool presented in this paper is developed to support. In Section 3 we present the most important requirements to the tool, and in Section 4 we give an example-driven presentation of the most important functionalities. In Section 5 we discuss the tool with respect to the identified requirements. Finally, in Section 6, related work is discussed before we conclude.

2 Background

According to the ISO 31000 risk management standard [8] risk analysis should be regularly conducted in order to assess and mitigate risks. The standard defines risk analysis as an iterative process consisting of five consecutive steps. *Establish the context* is to define the external and internal parameters to be accounted for when managing risk, and to set the scope and risk criteria for the risk management policy. *Risk identification* is to find, recognize and describe risks. *Risk estimation* is to comprehend the nature of risk and to determine the risk level.

Risk evaluation is to compare the risk estimation results with the risk criteria to determine whether the risk and its magnitude are acceptable or tolerable. *Risk treatment* is the process of modifying the risk.

The purpose of the standard is to provide methodological advice on how to manage risk, and it offers no specific techniques for how to conduct the activities in practice. Nor does the standard prescribe any support or advice on how to identify and document risk, for example by means of risk modeling techniques and languages. Most of the established risk analysis methods have activities that generally follow the principles laid out by the standard, and commonly they also provide techniques and come with tools to support the activities. CORAS [11] is an approach to model-driven risk analysis that is closely based on ISO 31000. It is self-contained in the sense that it offers a method that comes with concrete, practical guidelines, it comes with a language with modeling support and analysis techniques, and it comes with a tool that supports all activities.

In order to handle change, interdependencies and traceability in a systematic and methodic way, each activity must be supported by specialized guidelines and techniques. For this purpose the ISO 31000 process is in [12] generalized to include such guidelines throughout the whole process, and this generalization is in turn instantiated in CORAS. At the same time the CORAS language is extended and generalized to offer risk modeling support for the traceability techniques and for explicitly modeling and assessing changes to risks. The reader is referred to [12] for the details about the generalized method and language. In the following we highlight the most important principles, focusing on risk identification.

A part of establishing the context is to make a description of the target of analysis. The target description includes the documentation and models of the target of analysis, and serves as the basis for the subsequent risk assessment. It is therefore important that all relevant aspects of the target of analysis are properly documented. Moreover, dependencies to other systems and infrastructures must be included in the description of the environment in order to capture how external factors may affect the risks. An important aspect of the generalized process is that possible changes and evolutions are explicitly taken into account during context establishment and documented as part of the target description.

The risk identification involves identifying and documenting unwanted incidents with respect to the target of analysis and the identified assets. In the generalized approach, a further objective is to identify and document the changing risks given the description of change in the target of analysis. A main principle is that to the extent that we have identified and documented the risks for the target of analysis before changes, we only address the parts of the target that are affected by the change. The methodological guidelines are summarized as follows. 1) Identify and document risks by using as input the target description before changes have been taken into account. 2) Establish and document the traceability between the target description before change and the risk documentation resulting from the previous step. 3) Based on the traceability and the description of the changed target, identify the parts of the risk documentation that are persistent under change. 4) Conduct the risk identification of the

Fig. 1. Elements of CORAS threat diagrams

changed target only with respect to the parts of the target and the risks that are affected by the change.

In conducting these activities we make active use of three model artifacts, namely the target model, the risk model and the trace model. The target model needs to capture the relevant aspects of the target of analysis and can be built using any suitable notation, such as the UML [15]. For risk modeling CORAS threat diagrams are used. Threat diagrams document risks by describing how threats exploit vulnerabilities to initiate threat scenarios and unwanted incidents. A risk is the likelihood of an unwanted incident and its consequence for a specific asset. Risks are estimated by annotating each identified unwanted incident with a likelihood and a consequence. The graphical language constructs are shown in Figure 1, and an example diagram is depicted in Figure 5.

The trace model is part of the generalized CORAS approach and is used to build links between elements of the target model and elements of the risk model. The links are depicted in CORAS diagrams by means of the target segment construct which is an extension of the standard CORAS language. A further extension is the support for the explicit modeling of risk changes. Each CORAS language element can be assigned one of the three modes *before*, *after* and *before-after*. The mode *before* captures elements of the risk picture that become obsolete after change and are depicted in gray. The mode *after* captures elements that emerge after change and are depicted in the standard way with colors. The mode *before-after* captures elements that are persistent under change and are depicted using a double layer. Moreover, because the likelihoods and consequences of the latter elements may change these values are annotated in pairs, where the former is the value before change and the latter is the value after change. Examples of the use of these modes are shown in Figure 7.

A full risk analysis will typically result in a large number of threat diagrams covering the various parts of the target description. Without any automated tool support the task of tracing changes from the target model to the risk model must be conducted manually. Moreover, keeping track of how changes percolate, and correctly and consistently assigning modes to the risk elements, can be challenging. Hence, making efficient use of the generalized risk analysis method and the traceability requires proper tool support.

3 Tool Requirements

The tool is mainly a diagram editor that is designed to support on-the-fly risk modeling during structured brainstorming. The brainstorming sessions involve

one or two risk analysts as well as four to seven people with expert knowledge about the target of analysis. The tool is operated by a risk analyst who draws diagrams that are displayed to the participants while the discussions proceed. There are of course many requirements that apply for such a tool to adequately support the whole risk analysis process. In this section we focus on the requirements that are relevant for handling dependencies, traceability and change.

First and foremost, for the tool to fulfill its purposes it should support the specification of all kinds of CORAS diagrams using the language generalized to capture the three different modes with respect to change. Next, in order to deal with traceability the tool must support the specification of traceability links from elements of the target model to elements of the risk model. This, in turn, requires the tool to import (a representation of) the target model. CORAS does not prescribe a specific notation to be used for the target modeling, as different languages may be suitable in different risk analyses, so the tool should be able to import models irrespective of the chosen language.

Whereas we have generalized the CORAS language to explicitly model change, we cannot assume the corresponding expressiveness in the language chosen for the target modeling. Instead we assume that the target model is modified to reflect the changes, such that we have one model before the changes and one model after the changes. The specific parts of the target of analysis that have changed must then be determined by comparing the two models and generating the diff. For detecting changes and supporting traceability, this should be conducted automatically by the tool. Based on the specified traceability links and the diff, the system changes can now be traced to the parts of the risk model that may be affected. This should also be supported by the tool by automatically flagging the affected diagrams of the risk model.

When reassessing the risks the threat diagrams are updated by assigning modes of change to the elements. While the generalized language is useful for the documentation of and the reasoning about risk changes, it may be challenging to keep track of how changes percolate through the threat diagrams and to maintain consistency with respect to change. For this purpose there should be automatic support for detecting and resolving inconsistencies in the diagrams.

In the following the requirements to the tool that are relevant for handling dependencies and change are summarized.

- The tool should support on-the-fly modeling of syntactically correct CORAS diagrams with change.
- The tool should support the importing of the target models irrespective of the chosen language for target modeling.
- The tool should support the specification of traceability links between elements of the target model and elements of the risk model.
- The tool should automatically generate the diff between the target model before changes and the target model after changes.
- The tool should automatically flag all elements of the risk model that are affected by the system changes and therefore have to be reassessed.
- The tool should provide support for detecting and resolving inconsistencies with respect to change.

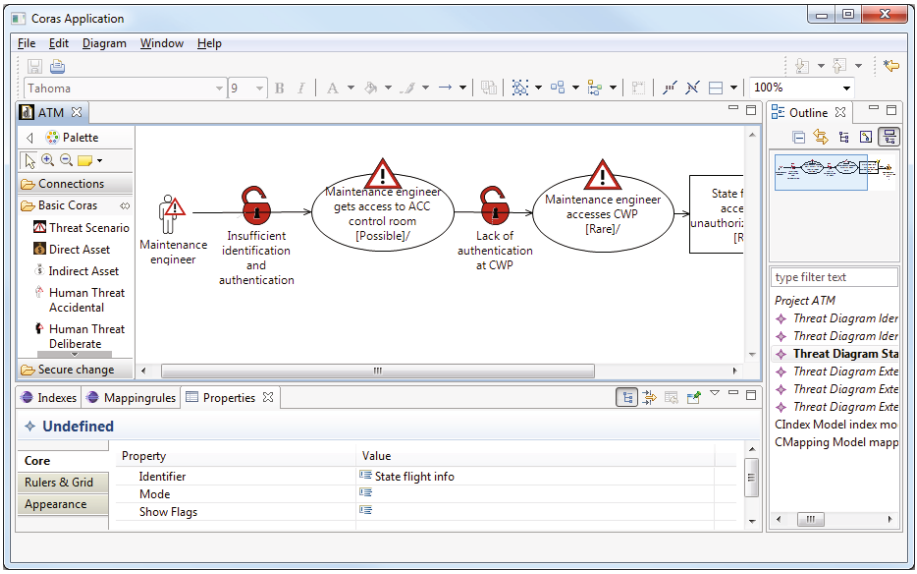

Fig. 2. Creating CORAS diagrams

4 Tool Functionality and Features

In this section we describe and exemplify the most important functionality and features of the tool regarding the support for handling traceability and change. The tool has been developed as a plugin to Eclipse such that it can be easily extended with new features and easily integrated with other Eclipse-based tools. To ensure that the models that are created using the tool are stored in a standard format, it is based on Eclipse Modeling Framework (EMF). The examples that we use are from the ATM domain and have been extracted from validation activities involving experts on ATM and on secure system engineering [17].

The main use of the tool is as a diagram editor for creating CORAS diagrams. The interface is shown in Figure 2 with the drawing canvas in the middle. The palette to the left contains all language elements and relations, and diagrams are created easily by drag-and-drop. For each diagram that is created, the user must first choose the kind of CORAS diagram. The syntactically correct diagram is defined by the meta-model of the language, and the tool automatically prevents the user from making diagrams that are grammatically incorrect.

As mentioned above, the risk identification and the creation of the threat diagrams are conducted by systematically going through the target model trying to determine where things can go wrong and how. The ATM validation activities addressed the initial phases of a system engineering process for the development of ATM services provided by an Air Navigation Service Provider (ANSP) in Area Control Centers (ACCs). The requirements were captured in collaboration

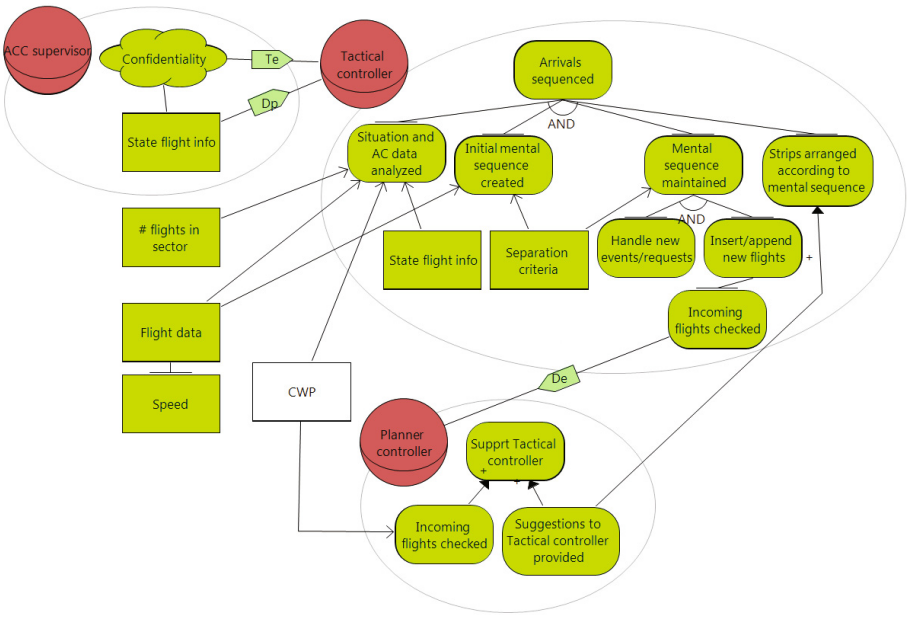

Fig. 3. Modeling the target of analysis

with the ANSP and modeled as SI* goal models [13]. A small fragment of the requirement model, which served as the target model, is shown in Figure 3.

SI* models show the goals (rounded rectangles) that actors/roles (circles) seek to achieve, as well as the resources (rectangles) that are needed for achieving the goals. Non-functional requirements, such as security goals, are captured by soft goals (clouds). A delegation marks a passage of responsibility for achieving a goal (delegation execution, De) or of authority with respect to a resource (delegation permission, Dp). Trust is a relation between actors/roles representing the expectations of the trustor about the capabilities of the trustee (trust execution, Te) or about the behavior of the trustee with respect to a permission (trust permission, Tp). Figure 3 shows three air traffic controllers (ATCOs) and some of their goals in the sequencing of flights during arrival management. The ACC supervisor is supervising the ATM activities, and is also responsible for protecting the confidentiality of State Flight info. A State Flight is any flight involving military, customs, police or other law enforcement services of a state, or any flight declared as such by state authorities. ATCOs need access to State Flight info since these flights are sequenced differently than commercial flights, but at the same time the information must be kept confidential to prevent misuse by potential adversaries. The Tactical controller is responsible for sequencing the arriving flights, and is supported by the Planner controller in achieving some of the sub-goals. The latter two ATCOs rely on several resources, such as the controller working position (CWP), which is their workstation.

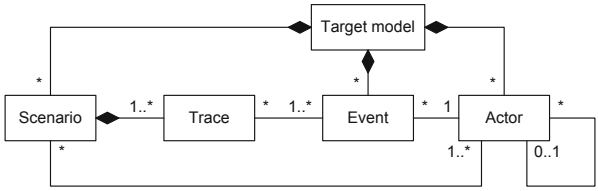

Fig. 4. Meta-model for generic target model

The risk analysis tool is designed to import the target model independent of the language that is chosen for making the target description. To this end, all target models must conform to a meta-model, *TM*, for the representation of the target model in the tool. This meta-model is shown in Figure 4 (somewhat simplified to focus on the conceptual aspects), and is defined to capture the parts of the target of analysis that are relevant for reasoning about risk, namely actors (which can be human and non-human), events that may occur, and scenarios that may unfold. Because the tool can take arbitrary target models as input, the user needs to define a mapping from the meta-model, *MM*, of the target model to the meta-model, *TM*, of the tool. The only requirement is that *MM* is defined according to an Ecore meta-model, which is the EMF specification of meta-models. The mapping is defined by a set of transformation rules that apply to elements, element attributes and element references. In the ATM case study with the use of SI*, we have, for example, mapped actor/role to actor and goal to scenario, including the names and the goals that are attributed to the actor/role. The specification of the transformation is a one-off task for the selected target modeling language; it is stored in a text file that is loaded by the tool together with the meta-model of the target model (here the SI* meta-model).

The tool is now ready to import the target model that is used. The purpose of importing the target model to the tool is to be able to specify traceability links, not to view or edit the target models, which instead is conducted separately in a designated tool. In order to be able to uniquely refer to the separate elements of the target model the tool creates an index of all elements, where each index is represented by a tuple (*ID*, *Name*, *Category*, *Description*, *Mode*). *ID* is a unique automatically generated identifier, *Name* is the name of the element as specified in the target model, *Category* is the kind of target model element (*Actor*, *Event* or *Scenario*), and *Mode* specifies the mode of the target model element with respect to change. *Description* is an initially empty field that can be filled in by the user if further explanation is desired or needed. The set of index tuples are represented in the tool in the table format shown at the bottom of Figure 5. Note that the mode of all target elements are currently *before* since no changes have yet been specified for the target system.

Given the threat diagrams and the index of the target model, traceability between them can be specified by defining mapping rules. Basically, a mapping rule is a pair of a target model element and a risk model element. Additionally, each mapping rule is given a tag with a chosen name for the mapping rule. In the visualization of the mapping rule in the threat diagrams, the name of the tag

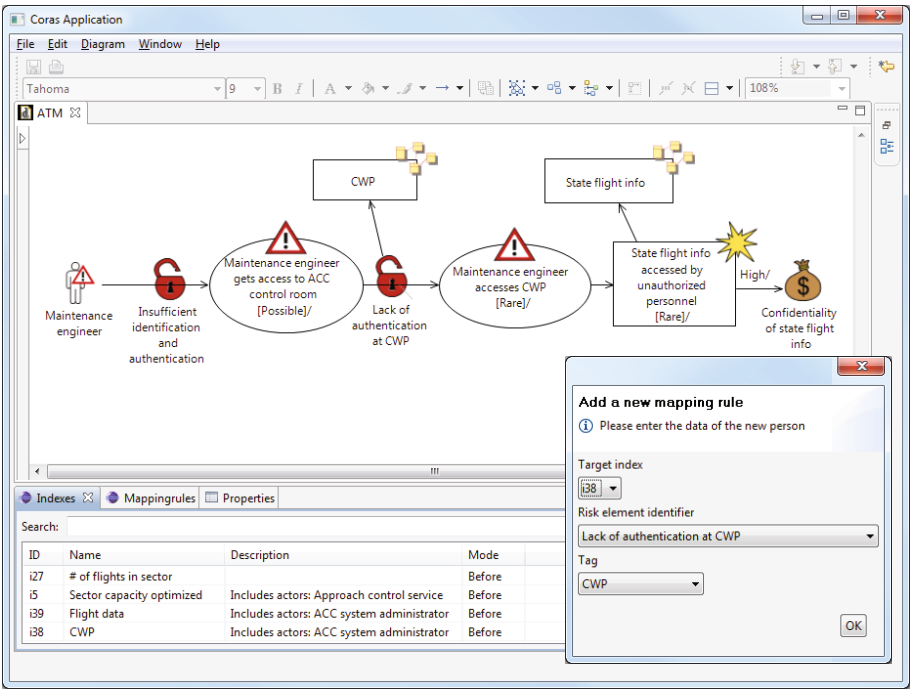

Fig. 5. Defining mapping rules

is automatically inserted. For example, in Figure 5 a mapping rule pairs target element with index *i38* (*CWP*) with the vulnerability *Lack of authentication at CWP*. A further purpose of using the tag is that it allows several mapping rules to be clustered. For example, all three actors in Figure 3 are ATCOs, and for elements of the risk model to which all three are related we can create three mapping rules with the same tag, *ATCO*.

Given the target model, the finalized risk model, and the trace model, the process is ready to proceed with handling changes that may occur. Two of the changes that were addressed in the ATM case study were the introduction of the AMAN and the SWIM. AMAN (Arrival Manager) is a decision support tool that automates the sequencing of flights in arrival management. The AMAN therefore needs to be fed with all relevant flight data, including State Flight info. SWIM (System Wide Information Management) is an information network that will integrate information systems and facilitate the flow of information between entities such as aircrafts, airports, ACCs, ATCOs, ANSPs, etc. Previously, State Flight info did not have to be digitally stored and shared, but could be kept separate from flight data (cf. Figure 3). With the AMAN introduction, however, this information needs to be part of the general flight data and shared over the SWIM network. This raises new security threats, for example with respect to the confidentiality of State Flight info, which is an asset in our case study.

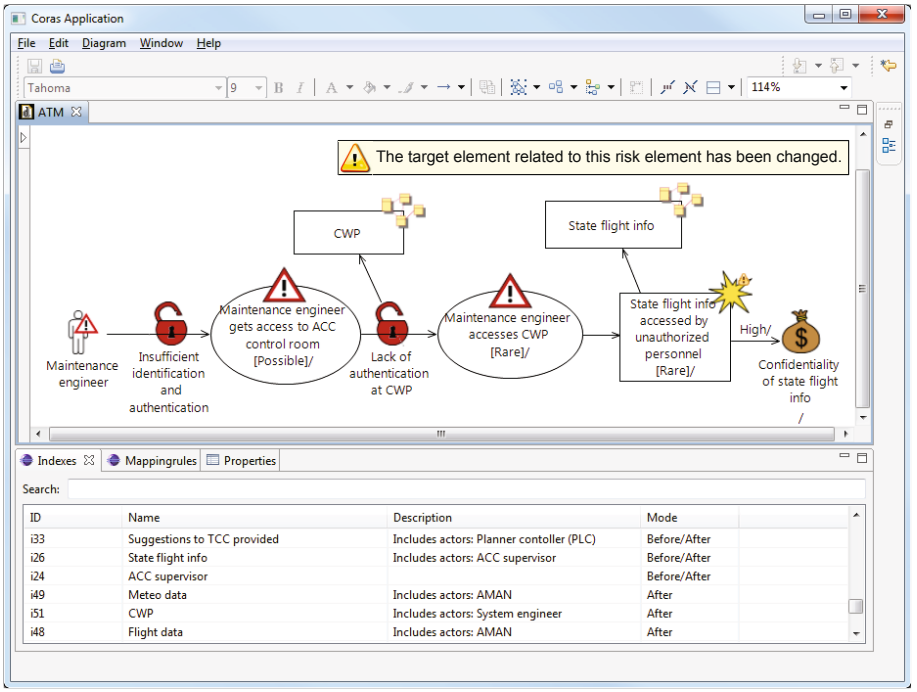

Fig. 6. Automatic flagging of changes

The first step in addressing the changes is to update the target model. As a result we have two target models, one for the situation before changes and one for the situation after changes. To determine what has changed we need to compare the two models. This is supported by the tool by the automatic generation of the diff, which is done by comparing two target model files as specified by the user. In practice, the tool first transforms the two models to its own target model representation and generates the diff based on this. It thereafter updates the index to include the changes and setting the correct change mode to each target model element. Due to space constrains we do not show the updated SI* model after change, but at the bottom of Figure 6 we see some of the new indexes with their mode. For example, System engineer is an actor that was included after the changes, whereas ACC supervisor is in the target model both before and after.

Due to the changes in the target of analysis, some parts of the risk analysis may have to be conducted anew. The tool supports the identification of the affected threat diagrams by flagging all risk model elements that are related to changes in target model elements. This is shown in Figure 6 with the warning sign on the unwanted incident *State flight info accessed by unauthorized personnel* and the pop-up message on the screen.

In reassessing the risks that are documented in the flagged threat diagrams, we need to determine which parts become obsolete and which parts are persistent, and we also need to identify new risks that may arise. The explicit modeling

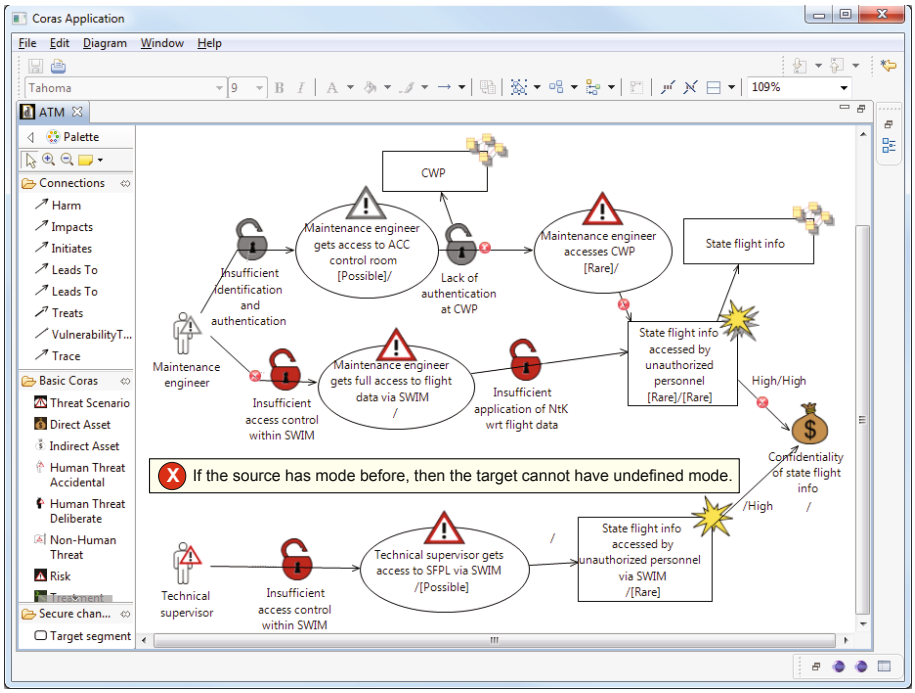

Fig. 7. Modeling change and resolving inconsistencies

and assessment of such changes are supported by the generalized language and by the tool by assigning one of the modes *before*, *after* and *before-after* to each element. This is exemplified in Figure 7, where, for example, the threat scenario *Maintenance engineer gets access to ACC control room* has mode *before*, the threat scenario *Technical supervisor gets access to SFPL via SWIM* has mode *after*, and the unwanted incident *State flight info accessed by unauthorized personnel* has mode *before-after*. Note that the default mode is *undefined*, i.e. when a new element is inserted there is no mode assigned to it.

While the generalized language supports the assessment and documentation of changing risks, the increased complexity makes it more challenging to keep diagrams consistent. For example, a threat diagram element of mode *before* cannot be related to an element of mode *after* as they do not coexist. Moreover, a threat diagram element should not have an undefined mode when it is related to elements where the mode has been specified. To support the user in modeling risk changes the tool does automatic consistency checking, and warnings pop up whenever an inconsistency in introduced. Each inconsistency is flagged with a red circle with a white cross, for example on the relation between the threat *Maintenance engineer* and the threat scenario *Maintenance engineer gets full access to flight data via SWIM*. This is, as the shown pop-up explains, because

the former is assigned mode *before* and the latter has undefined mode. A further example is on the relation between the unwanted incident (in mode *before-after*) and the asset (in undefined mode); the latter should also be in mode *before-after*.

In addition to detecting errors, the automatic consistency checking is very useful for systematically tracing changes through the threat diagrams. For example, when starting from the left and assigning a mode to the threat, warnings will immediately be flagged on the relations to the adjacent elements. When resolving this inconsistency these warnings disappear, and new ones emerge one step further to the right. The user can in this way update the diagram step by step while maintaining the consistency.

Finally, when all threat diagrams are updated, finalized and consistent, new mapping rules should be defined to establish the traceability with respect to the updated target model. This ensures that all documentation is ready for any further future changes that need to be dealt with in the same way.

5 Discussion

In this section we discuss the tool with respect to the requirements identified in Section 3. As mentioned before, these requirements focus on the need for tool support for handing dependencies, traceability and change. For a more general evaluation that also takes into account the methodological support and the language support, the reader is referred to [17].

The tool should support on-the-fly modeling of syntactically correct CORAS diagrams with change. The tool is a diagram editor with all language constructs immediately available by drag-and-drop from the palette. Relations between the elements can moreover easily be attached by click-and-drag, where the tool automatically selects the syntactically correct relation. Moreover, as the tool implements the meta-model of the CORAS language, the user is prevented from making diagrams that are grammatically incorrect. An exception from the latter is the use of the extended notation to enable the modeling of risk changes; instead of preventing the user from making inconsistencies with respect to change, the tool automatically flags all such errors. However, this is intended as diagrams are updated in a stepwise manner when change is brought into the picture. If inconsistencies with respect to change was prevented, the users would have to build all the diagrams from scratch instead of updating them.

The tool should support the importing of the target models irrespective of the chosen language for target modeling. The tool is designed to import target models of arbitrary languages, since different notations are suitable for different kinds of target systems and since different communities can have different modeling preferences. The tool therefore uses one common and generic meta-model for all target models. This is at the cost of users having to specify the transformation from the meta-model of the chosen language to the meta-model used by the tool, which requires some expertise in language design. However, this is a one-off task for a given target modeling language, and the benefit is of course that the use of

the risk analysis method and tool is not restricted to only one target modeling language. Still, one restriction with respect to the choice of target modeling language is that the its meta-model must be an Ecore meta-model.

The tool should support the specification of traceability links between elements of the target model and elements of the risk model. This is supported by the functionality of defining mapping rules. In order to be able to specify all possible mapping rules, it is necessary that all elements of the target model are indexed by the tool. This in turn requires that the transformation rules mapping the meta-model of the target modeling language to the meta-model in the tool is complete, i.e. that it covers all elements of the target model. For some languages, it may be that the meta-model of the tool is not rich enough, resulting in dependencies that cannot be captured automatically.

The tool should automatically generate the diff between the target model before changes and the target model after changes. This is supported by the tool by comparing the two models and generating the diff between them, identifying elements that are deleted, added or modified. The diff is generated by comparing the representation of the target models in the tool, i.e. according to the target meta-model of the tool. As for the traceability, this means that if there are target elements that are not captured by this meta-model, changes to these are not captured. As an alternative to using a generic target meta-model in the tool, the tool could be customized for one specific target modeling language, such as the UML or SI*. The tool would then use the meta-model for this specific language, and all elements and changes would be captured. Moreover, the user would not have to specify the transformation rules for the meta-model. When designing the tool we aimed for flexibility and general applicability, which explains our design choice. However, customized versions could be developed by replacing the generic target meta-model with any specific one.

The tool should automatically flag all elements of the risk model that are affected by the system changes and therefore have to be reassessed. This is supported by the tool, but the extent to which all affected threat diagrams are flagged depends on two things. First, the generated diff must be able to capture all system changes as discussed above. Second, the users of the tool of course need to specify mapping rules that cover all system changes that occur.

The tool should provide support for detecting and resolving inconsistencies with respect to change. This is supported by the flagging of inconsistencies as discussed above. Resolving the inconsistencies must be conducted manually, but the user can immediately see when consistency is restored.

6 Related Work and Conclusion

Model Versioning and Evolution (MoVE) (http://move.q-e.at/) [4] is an approach to build an infrastructure to maintain the validity, mutual consistency and interdependencies between models as they evolve over time within model engineering. The approach does not target security and risk in particular, but

rather builds a tool-supported infrastructure for versioning of several interdependent models, for example for software architecture and design, business processes, services, security and risk. Similar to our approach, the underlying idea is to provide support for tracing changes from one model to another so as to ensure that they are globally up-to-date and mutually consistent. However, although the infrastructure can support the handling of dependencies and change in risk analysis, as exemplified in a case study, there is no specific modeling or methodological support for this. Instead the user chooses the models and notations, such as CORAS or UML, to be managed by MoVE.

ProSecO [7] is a model-based approach to risk analysis with support for dependency identification and modeling. The approach relates risk elements to elements of a functional model of the target of analysis. Moreover, the model elements are related to security objectives and security requirements, and risks are related to threats and security controls. Although risk assessment is supported, there is no risk modeling support other than a description of incidents and their likelihood and consequence.

Dependent CORAS [3,11] is an extension of the CORAS language to support modular risk analysis, as well as the modeling and reasoning about interdependencies. However, the purpose is not to capture and model dependencies of the risk models on the target system, but rather interdependencies between different risk models. This is in particular relevant for mutually dependent critical infrastructures, where the risks and risk level of one can strongly depend on the risks and risk level of the other. Tool support is provided for creating and editing Dependent CORAS diagrams, but there is no automation of dependency analysis. In [9] a method is proposed to capture and monitor the impact of service dependencies in interdependent systems of systems. The dependencies between system services are explicitly modeled, and subsequently taken into account during risk identification and modeling. However, the problem of tool-supported traceability between system elements and risk elements is outside the scope.

In [17] an approach to integrate risk assessment and modeling with system development and modeling to aid the handling of dependencies and change between the two domains is presented. The former is supported by the Rinforzando tool and the latter by the SOA Modelling Suite (SMS). The tool supports tight integration between the system model and the risk model in several ways. For example, in the risk models assets and supporting equipment, technologies, work processes, etc. can be selected directly from the SMS target models. Moreover, dependencies are continuously maintained such that modifications in one model is either automatically propagated to the other model, or warning flags appear as pop-ups. This integration of system and risk model is tighter than in our approach, but it requires the use of SMS for system modeling. Furthermore, explicit modeling of risk changes is not supported as the objective is rather to maintain the mutual consistency between models.

In this paper we have presented a tool to support the method and language presented in [12], enabling the traceability from the target of analysis and/or its

environment to the risks. The tool gives automated support for tracing changes and identifying risks that need to be reassessed in order to maintain the validity of risk models, as well as automated support for maintaining consistency of risk models. The tool has been validated in the ATM domain, which is a critical infrastructure with strong dependencies on other infrastructures.

Further work is required for evaluating and validating the tool with the use of different modeling languages for capturing the target of analysis, although the results have been promising in case studies so far. A further topic for future work is to develop a tighter integration between the target model and the risk model to support mutual consistency and automated updates of model elements in case of changes.

Acknowledgments. This work has been partially funded by the European Commission via the NESSoS (256980) network of excellence.

References

1. Alberts, C.J., Davey, J.: OCTAVE criteria version 2.0. Technical report CMU/SEI-2001-TR-016, Carnegie Mellon University (2004)
2. Barber, B., Davey, J.: The use of the CCTA risk analysis and management methodology CRAMM in health information systems. In: 7th International Congress on Medical Informatics (MEDINFO 1992), pp. 1589–1593. North-Holland (1992)
3. Brændeland, G., Refsdal, A., Stølen, K.: Modular analysis and modelling of risk scenarios with dependencies. Journal of Systems and Software 83(10), 1995–2013 (2010)
4. Breu, M., Breu, R., Löw, S.: MoVEing forward: Towards an architecture and processes for a Living Models infrastructure. International Journal On Advances in Life Sciences 3(1-2), 12–22 (2011)
5. Communication from the Commission on a European programme for critical infrastructure protection. In: The European Commission, COM, 786 final (2006)
6. EUROCONTROL: Air traffic management strategy for the years 2000+ (2003)
7. Innerhofer-Oberperfler, F., Breu, R.: Using an enterprise architecture for IT risk management. In: Information Security South Africa Conference, ISSA 2006 (2006)
8. International Organization for Standardization: ISO 31000 Risk management – Principles and guidelines (2009)
9. Ligaarden, O.S., Refsdal, A., Stølen, K.: Using indicators to monitor security risk in systems of systems: How to capture and measure the impact of service dependencies on the security of provided services. In: IT Security Governance Innovations: Theory and Research. IGI Global (to appear, 2012)
10. Lund, M.S., Solhaug, B., Stølen, K.: Evolution in relation to risk and trust management. Computer 43(5), 49–55 (2010)
11. Lund, M.S., Solhaug, B., Stølen, K.: Model-Driven Risk Analysis – The CORAS Approach. Springer (2011)
12. Lund, M.S., Solhaug, B., Stølen, K.: Risk Analysis of Changing and Evolving Systems Using CORAS. In: Aldini, A., Gorrieri, R. (eds.) FOSAD VI. LNCS, vol. 6858, pp. 231–274. Springer, Heidelberg (2011)

13. Massacci, F., Mylopoulos, J., Zannone, N.: Security Requirements Engineering: The SI* Modeling Language and the Secure Tropos Methodology. In: Ras, Z.W., Tsay, L.-S. (eds.) Advances in Intelligent Information Systems. SCI, vol. 265, pp. 147–174. Springer, Heidelberg (2010)
14. Microsoft Solutions for Security and Compliance and Microsoft Security Center of Excellence: The Security Risk Management Guide (2006)
15. Object Management Group: OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.2, OMG Document: formal/2009-02-02 (2009)
16. Peltier, T.R.: Information Security Risk Analysis, 2nd edn. Auerbach Publications (2005)
17. Report on the industrial validation of SecureChange solutions. SecureChange project deliverable D1.3 (2012)