CHESS Dependability Guide

14 April 2020

1 Table of Contents

List of Figures

2 List of Tables

3 Document history

Date	Changes
14 April 2020	First version

4 Introduction

This guide provides information about the CHESS Dependability and analysis.

5 CHESS Dependability profile

See CHESSML profile guide.

See https://github.com/montex/CHESS-SBA/wiki/CHESS-SBA-Extensions.

6 Dependability Analysis

6.1 State Based Analysis

See https://github.com/montex/CHESS-SBA/wiki/CHESS-SBA-Extensions >-

See Module 05 at http://www.chess-project.org/page/training

6.2 FLA

See Module 7A at http://www.chess-project.org/page/training)

ConcertoFLA rules are logical expressions, which specify the component's behaviour by describing the input/output relationship. ConcertoFLA rule is a combination of the port (input/output) and the guide word referring to the failure mode; supporting standard failure modes i.e., timing, value and provision. Each of these failure modes has two specializations, which are early & late, ValueSubtle & ValueCoarse, and Omission & Commission corresponding to timing, value and provision respectively.

ConcertoFLA allows users to calculate the behaviour at system-level, based on the specification of the behaviour of individual components. During the analysis, ConcertoFLA calculates the failure propagation paths and produces their representation according to the specifications of FlaMM meta model[1]. These failure propagation paths are utilized to generate a fault tree. In fault tree terminology, the failure at system level is referred to as *top event* and the contributing failures are classified as intermediate and basic events. In safety context, the top event refers to a safety hazard which may cause accidents. In the security context, the top event is a breach of security properties i.e., confidentiality, integrity and availability. The intermediate and basic events contributing to the top event could also be due to the compromise of any of the concerns. For example, a cyber-security attack, which makes a component to halt its services, may cause a safety hazard, which, in turn, may cause an accident. Therefore, a fault tree integrating the security threat events contributing to the safety hazard could enable security-informed safety. To support the generation of such rich fault tree, a further elaboration is introduced to the input/output failure behaviour

[1] CONCERTO Deliverable D3.3 November 2015 Design and implementation of analysis methods for non-functional properties – Final version.

of the components. Before discussing modelling of the elaborated failure behaviour of components, first, security threat process and security related terms are introduced in what follows.

The causality-chain that leads to the violation of the security-related properties is illustrated in Figure 1. A threat event, initiated by a threat source agent, able to exploit a vulnerability of an asset (e.g. a component/system) may result in a loss to the confidentiality, integrity and/or availability (often, together referred to as CIA) of the asset. *Threat* refers to the event or capability to breach security and cause harm. Where the *vulnerability* is a weakness or internal flaw in the design, architecture or implementation of a service/application. A threat source could be a malicious cyber-security attack, non-malicious human errors, natural/human made disasters, etc. It is also worth noting that the accidents caused by safety hazards can also be a threat source enabling a situation, where a threat may exploit a vulnerability and cause a security breach. However, to this end, the focus is on cyber-security attacks as threat source; hence attacker as a threat source agent. The loss to CIA of a component or system, which is a consequence of an attack, could be as following:

- Unauthorized access of the system (loss of confidentiality)
- Halting services of the system (loss of availability)
- Corrupting the services of the system (loss of integrity)

For instance, a cyber-security attack on a component, where a *data corruption* threat exploits the *missing data integrity schemes* vulnerability, may result in corrupting the services of the component.

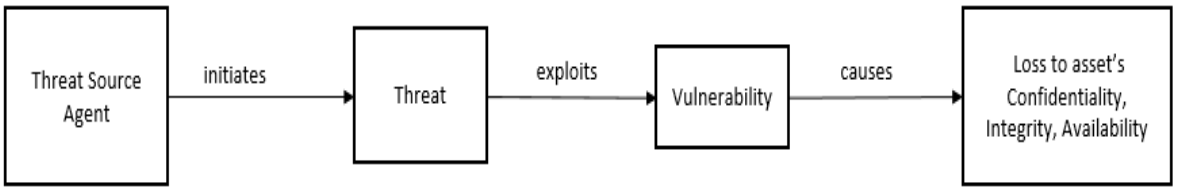


Figure 1. Process of Security breach **Error! Reference source not found. Error! Reference source not found.**

The further elaboration of the input/output failure behaviour of components could be modelled using state machine diagram. Safeconcert, which is the CHESS dependability profile, allows to tag the state machine with <<ErrorModel>> stereotype. The error model provides support for modelling state transitions, erroneous state and the effect of this on a property of the component and its nominal behaviour. The state transition in the error model are specialized and could be tagged with <<Failure>> and <<InternalFault>> stereotype.

To model the security, using *an* <<ErrorModel>>-tagged state machine, the failure, internal fault and effect are extended to include security threats, vulnerability and consequences respectively. The security threats could be represented by a pre-loadable vocabulary (through exploitation of the connection with EPF Composer, where the requirements mandated by the standards are modeled), which refers to the specific threats used within a specific domain/standard. The inclusion of different types of threat could be collected from the catalogues of the domain and standard. For example, in the space domain, when engineering a space mission, the vocabulary provides a pre-defined enumeration of common/discovered security threats collected from CCSDS 350.1-G-2 **Error! Reference source not found.** and NIST 800-30 **Error! Reference source not found.** as well as by the personal competences (e.g., respective system engineer, security analyst, etc.). In a similar fashion, the vulnerabilities could be represented as a pre-defined enumeration collected through different sources (for example personal competence, standards and results of previous threat analysis, etc.). Finally, the consequences could also be modelled using pre-defined effects, which refers to the loss of CIA. Figure 2 illustrates the error model, where a cyber-security attack initiates *data corruption threat* and exploiting the *value check function is false* vulnerability thus causing a transition to erroneous state.

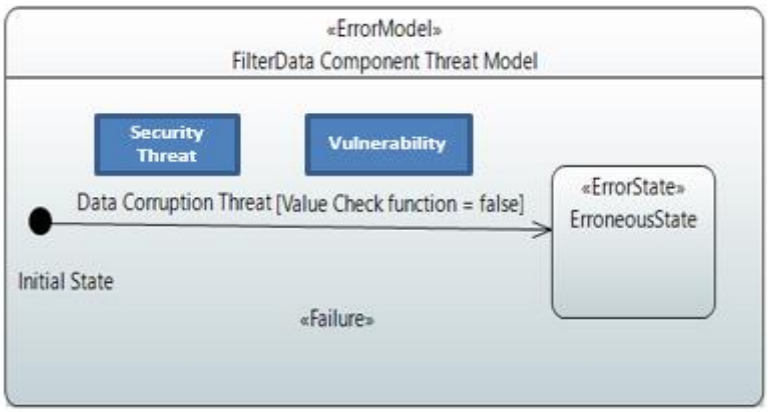


Figure 2. Error Model showing erroneous state transition due to security threat event and vulnerability

A component could have multiple instances of <<ErrorModel>>-tagged state machines, attached to it. Each instance would provide the elaboration of input/output failure behaviour addressing a specific concern.

6.2.1 Specify Failure Behaviour of CHESS System Components

The failure behaviour of a component can be defined by applying “FLABehavior” stereotype to that specific component. This failure behaviour is specified via Failure Propagation Transform Calculus (FPTC) (ConcertoFLA) rules. The FPTC rules provide input/output failure behaviour of a component, where a failure on input port is mapped to the failure on the output port. Figure below shows a component with “FLABehavior” stereotype and FPTC rules specifications.

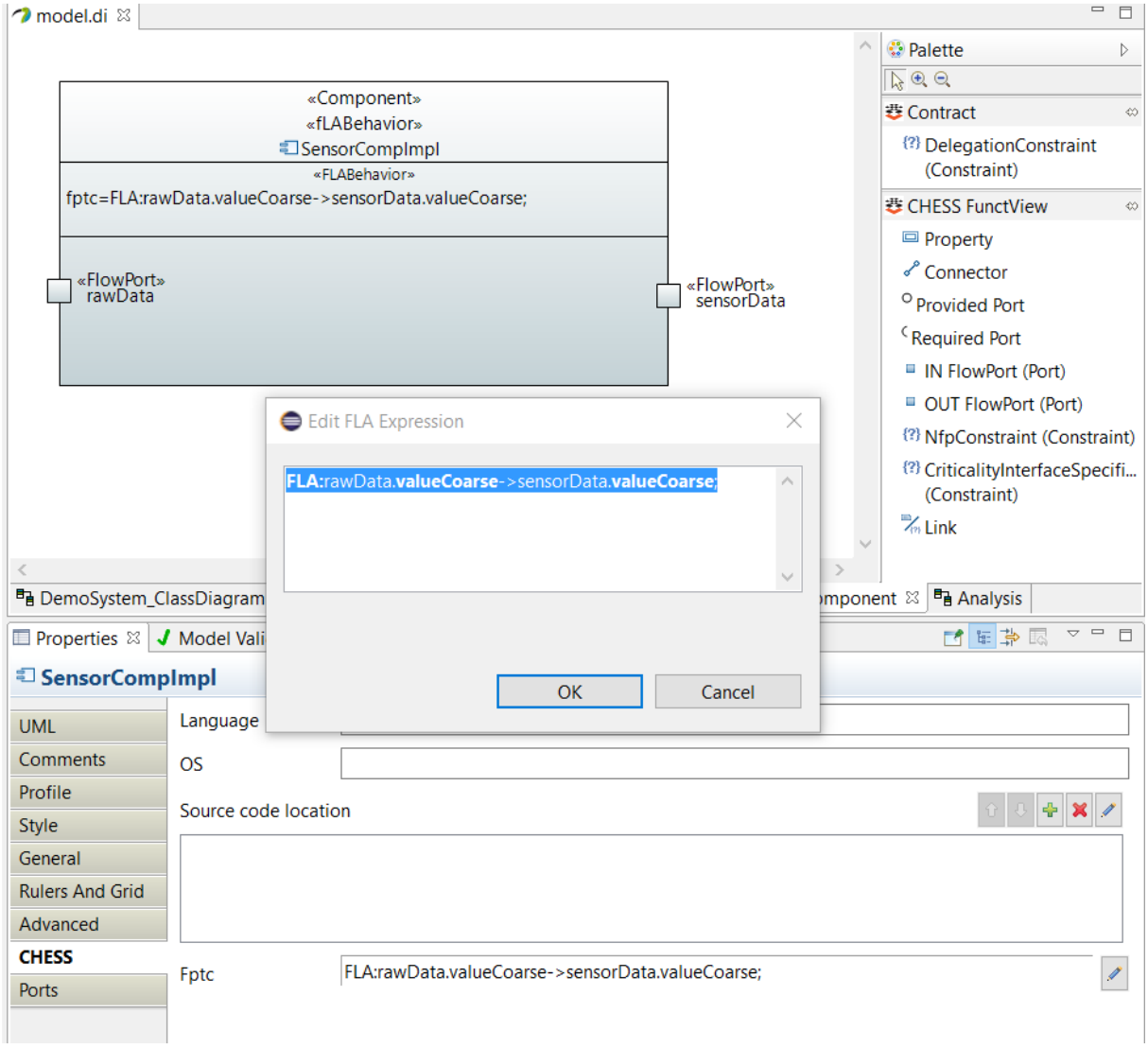
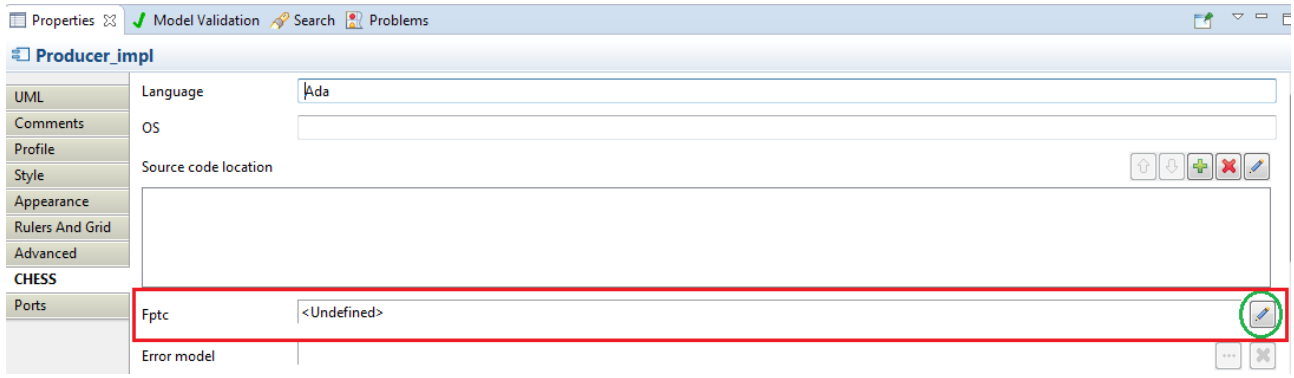


Figure 3 Failure Behaviour specification of a component

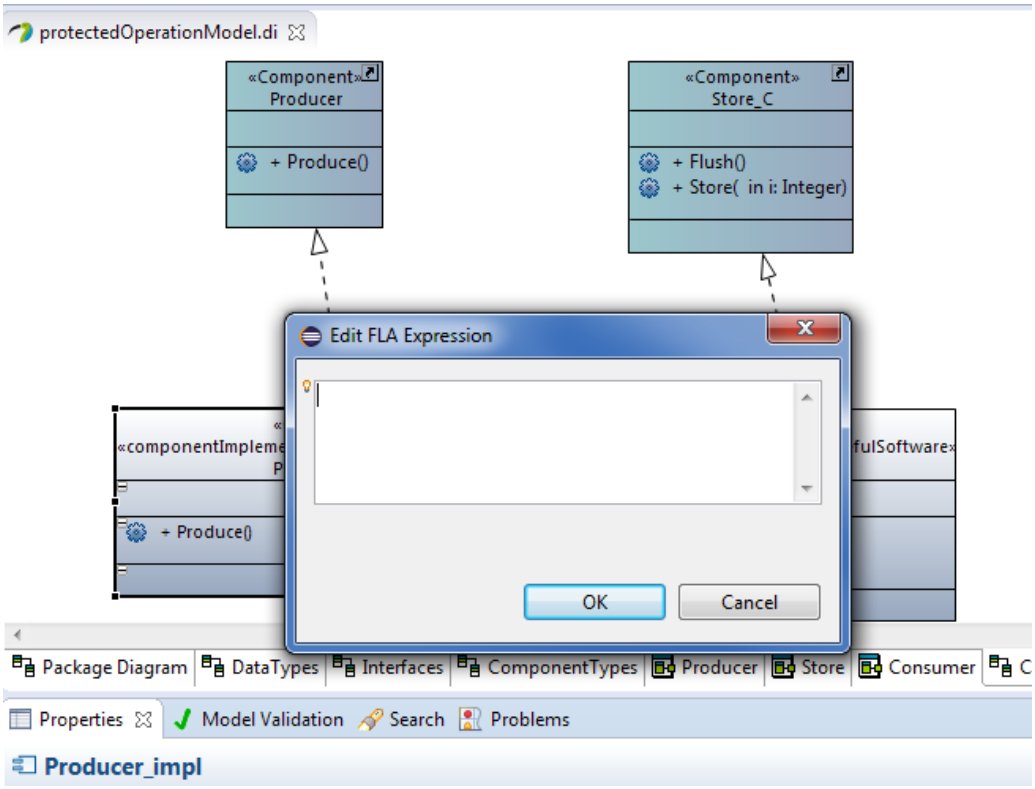
6.2.2 Editing Failure Propagation rules with XText (content assistance)

Open the CHESS model and select a Component stereotyped as FLABehavior.

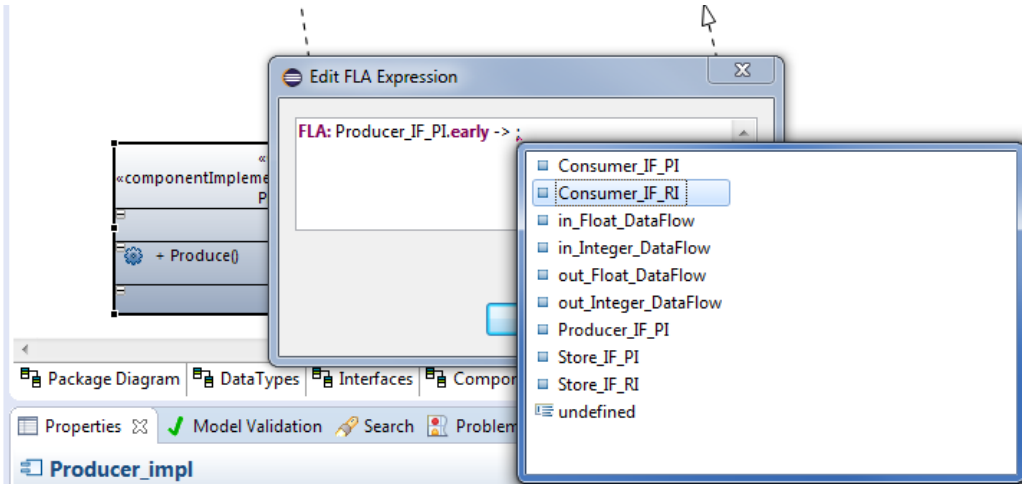
Open the CHESS Property tab:



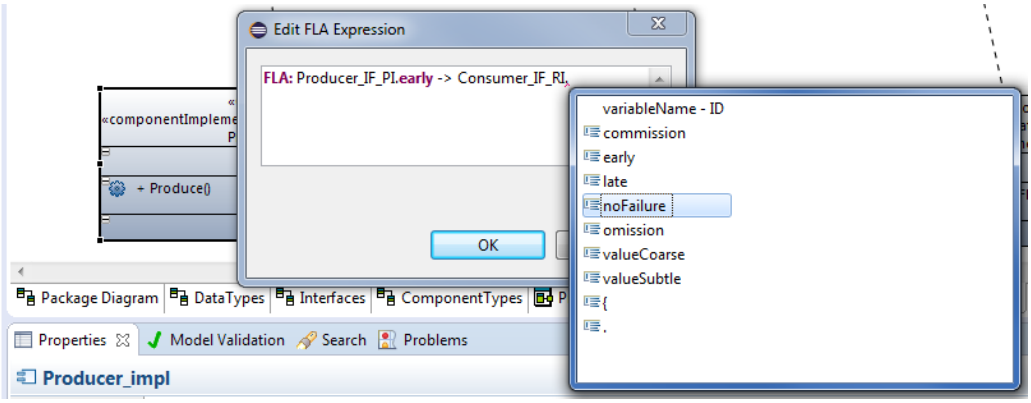
The FPTC property (in red) can be edited clicking the highlighted (in green) Edit button:



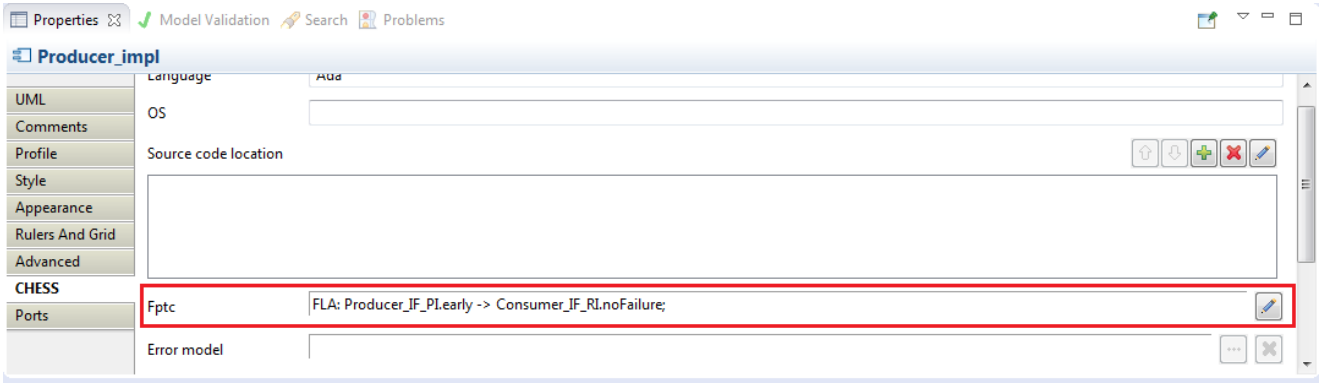
The editor widget allows the editing of the property with the Xtext support. Press CTRL+SPACE to activate context assist:



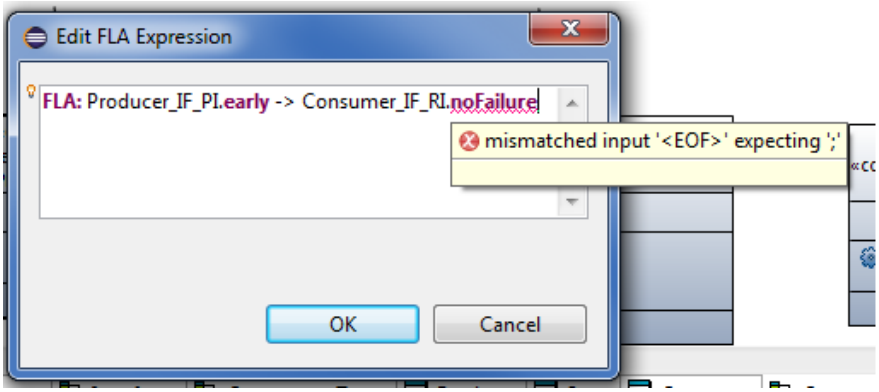
Context assist works for both referring CHESS model elements (Ports in the example) and for selecting enumeration literals:



When “OK” is pressed, the expression is written in the model as shown in the CHESS property tab:



The Xtext-enriched Editor can also highlight errors in the expression (according the defined grammar), for example, the following expression is missing the closing semicolon:



6.2.3 FLA rules with MARTE::ClientServerPorts and SysML::FLowPorts

FLA adopts the following rules to determine if a ClientServerPort has to be considered as output or input port wrt failure propagation:

- a ClientServerPort is considered as input port if:
  - - ClientServerPort.kind=provided and *all* the parameters defined for the interface operations have direction kind = 'in', or

        - ClientServerPort.kind=required and *all* the parameters defined for the interface operations have direction kind = 'out'
    - a ClientServerPort is considered as output port if:
        - ClientServerPort.kind=provided and exists *at least one* parameter defined for an interface operation which has direction kind = 'out', or
        - ClientServerPort.kind=required and exists *at least one* parameter defined for an interface operation which has direction kind = 'in'

In case of FlowPort FLA uses the FlowPort.direction attribute ('in' or 'out').

6.2.4 Dependability Co-Analysis with FLA

This section describes the usage of papyrus CHESS to model the architectural specifications and perform dependability co-analysis along with the generation of multi-concern fault tree.

6.2.4.1 Specialize Failure Behaviour of Component for Security Concern

To further specialize the failure behaviour “ErrorModelBehavior” stereotype is assigned to the components. The specialization is performed through defining a state machine diagram for the component stereotyped as “ErrorModel”. Using this state machine, the error states and the transitions are modelled. These transitions are stereotyped as “Attack” and “Vulnerability” to specify the security attack exploiting a vulnerability of the component. Figure 200 shows the “ErrorModel” stereotyped state machine with “Attack”, “Vulnerability” and “Failure” stereotyped transitions. The kind of attack and vulnerability are provided under the attack and vulnerability stereotype respectively.

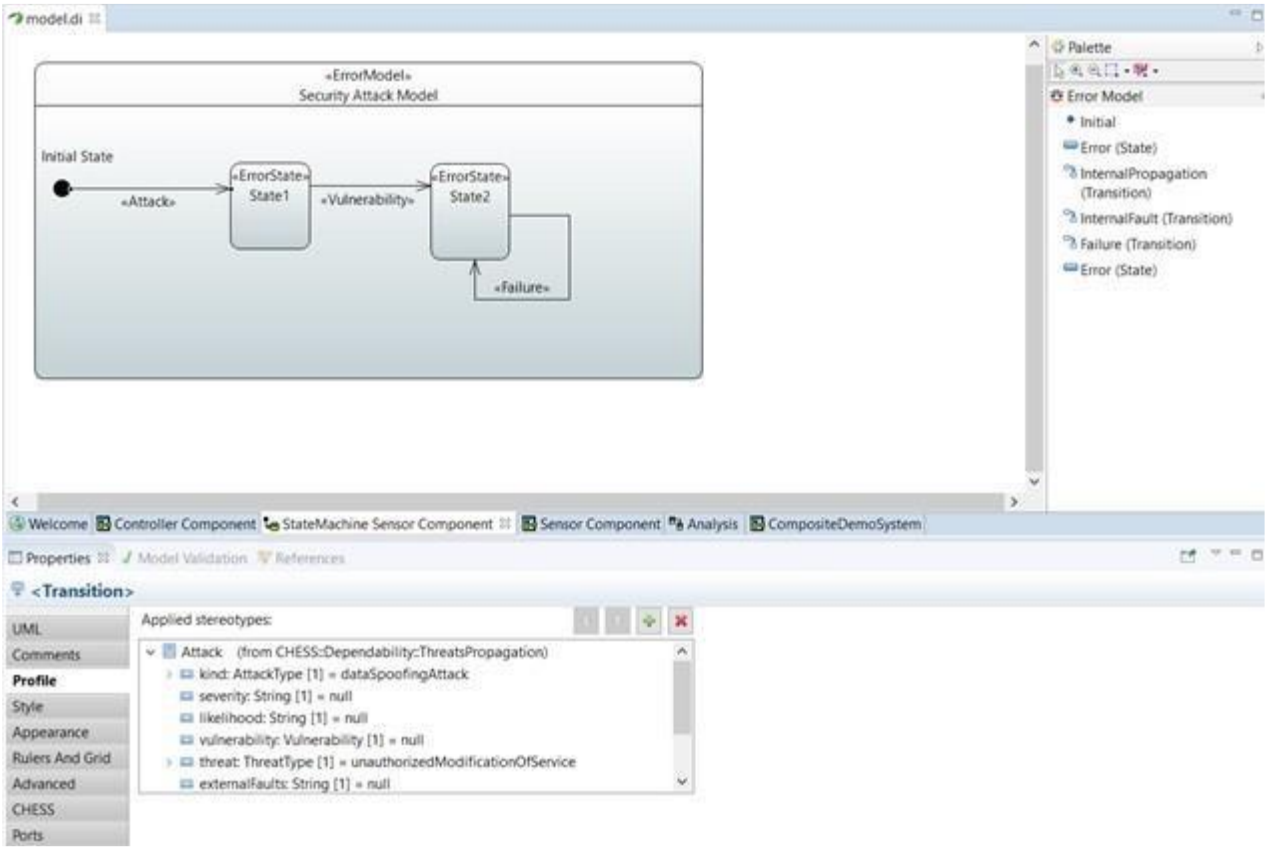


Figure 4. Specialization of Failure Behaviour for Security Concern

6.2.5 Invoke ConcertoFLA and Generate Fault Tree

ConcertoFLA analysis is invoked from the CHESS menu entry titled as “Failure Logic Analysis (Concerto-FLA)”, as shown in Figure 7, to calculate the failure propagation paths.

Figure 5 shows how failure types occurring on top level-input ports (on the left of the bottom figure), i.e. the fault injection, can be provided for *ComponentImplementations* at type and instance level, to allow FPTC analysis.

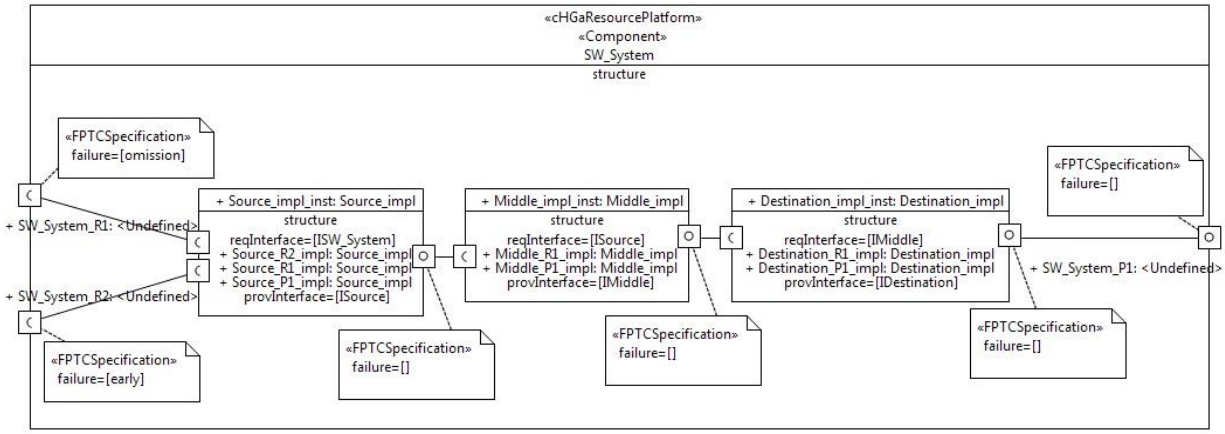


Figure 5: FPTC fault injectionspecification

Figure 6 shows how the results of the FPTC analysis about failure types occurring at output-port level are back propagated in the model.

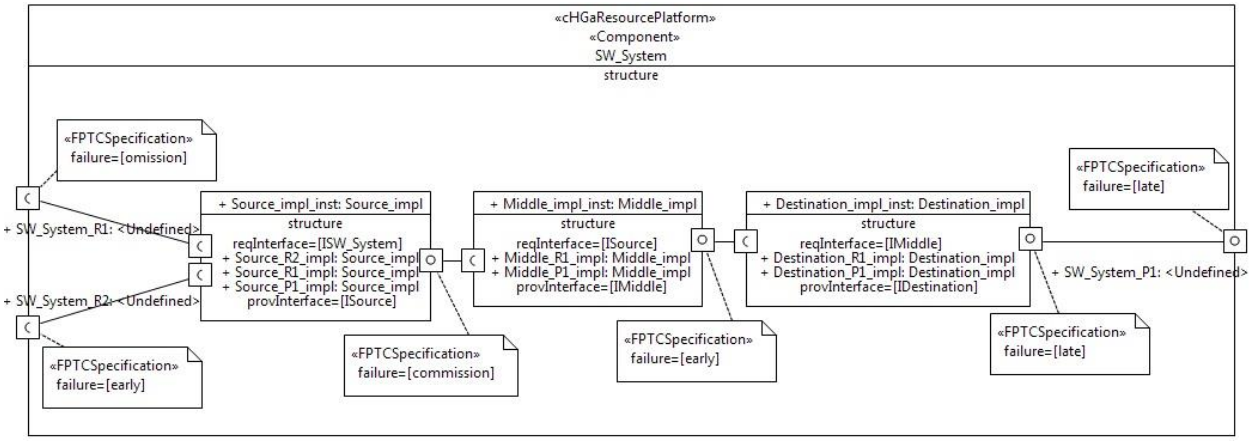


Figure 6: Results of the FPTC analysis

The multi-concern fault tree is generated from the CHESS menu entry titled as shown in Figure 7.

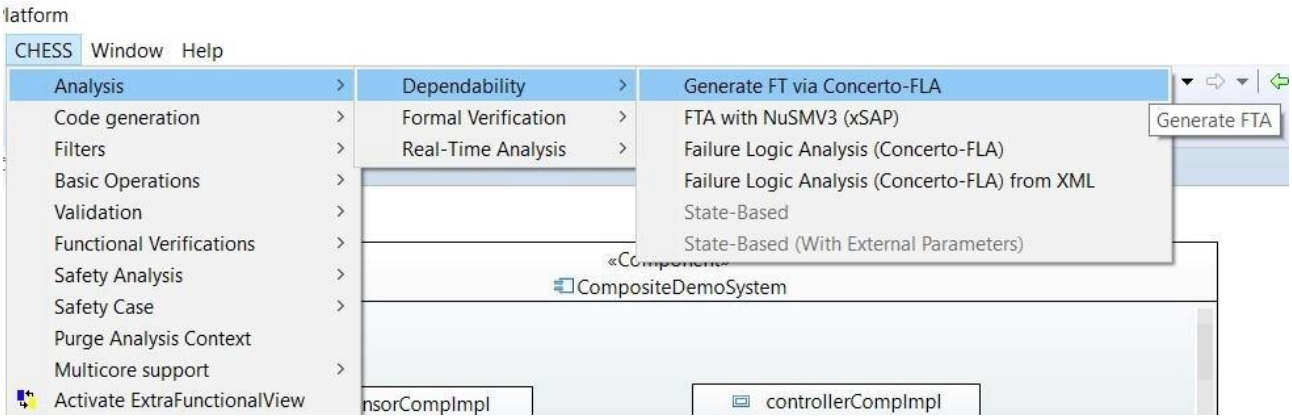


Figure 7. Invoking Failure Logic Analysis and Fault Tree Generation

6.2.6 FI4FA

- FI4FA - Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures (see Module 7B at http://www.chess-project.org/page/training)

6.3 Mobius

See the CHESS-Mobius Integration Guidelines.

6.4 Contract-based Analysis and Model Checking Safety Analysis

See the Contract-based Analysis and Model Checking Safety Analysis guide.