CHESS – Mobius Integration Guidelines

30 May 2019

1 Table of Contents

2 Introduction

This document describes the modelling of cyber security elements in CHESS. For this aim, the CHESS profile was extended. Model transformations were added to the CHESS Toolset to generate a Mobius Project (composed of SAN and REP/JOIN models) from the CHESS model with cyber security entities.

A detailed description of modelling in the CHESS Toolset is out of the scope of this document. Please refer to [1]

3 CHESS modelling language Extensions

3.1 SysML Block Modelling

A Vulnerability is modelled as a UML Port owned by a SysML Block: apply the stereotype <<**Vulnerability**>> on a Port. A block can have multiple vulnerabilities. A block with one or more vulnerabilities should be stereotyped as <<**ErrorModelBehavior**>>

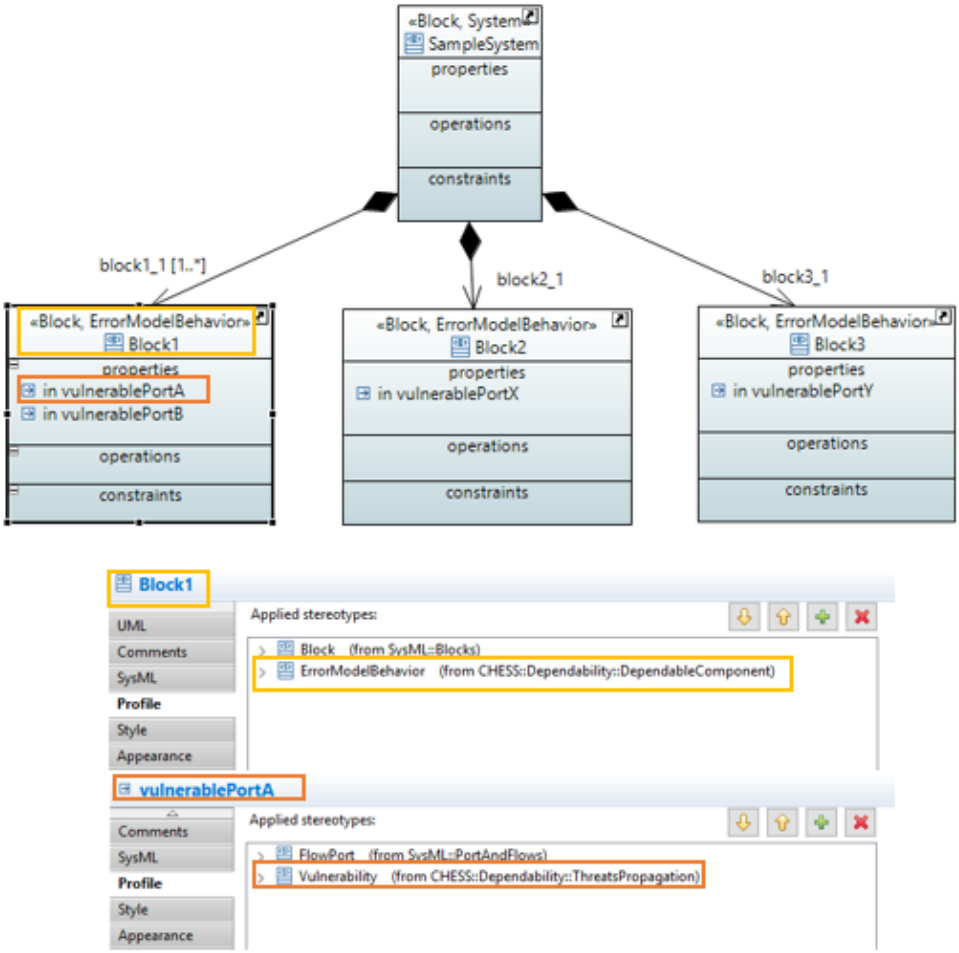

Figure 1: Blocks with Vulnerability Ports

The property “errorModel” (of an ErrorModelBehavior stereotype) should be set to a state machine stereotyped as <<**ErrorModel**>> and described in the following section. Only the first Error Model is taken

into account when generating the Mobius Project (despite the fact that the “errorModel” property has multiplicity *).

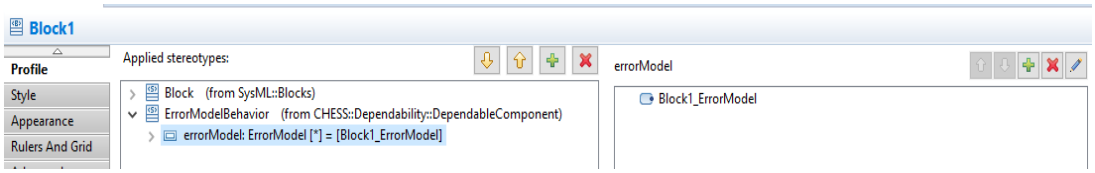

Figure 2: Set Error Model

In a decomposition it makes sense to model vulnerabilities and error models only for blocks which are “leaves” in the decomposition (no further decomposed) as other blocks are just containers.

3.2 Error Model State Machine

1. Create a new state machine diagram in the CHESS model (creates a new UML StateMachine element) as owned by a SysML block stereotyped as <<ErrorModelBehavior>> (see previous section).
2. States of an Error Model state machine should be stereotyped with one of the following:
    a. <<**NormalState**>> only one state should have this stereotype applied and it should be connected with the initial pseudo state. It represents the normal operational conditions of the Block, unaffected by external Attacks (i.e. Healthy)
    b. <<**DegradedState**>> multiple states can have this stereotype applied. They represent the operational conditions of the block affected by a successful external Attack (i.e. Compromised)
    c. <<**ErrorState**>> A state of the component that is considered erroneous, as in [2].
3. Transition of an Error Model state machine should be stereotyped with one of the following:
    a. <<**Attack**>> is used to connect the Healthy state with a Compromised state. It is mandatory to specify the value of the “vulnerability” property. Only a Port stereotyped as <<Vulnerability>> and owned by the enclosing Block is a valid value for this property.
    b. <<**InternalFault**>> is used to connect a Compromised or Healthy state to an Error state. The value of property “occurrence” can be set to model the Rate of the transition to the Error state, if left unspecified it will be represented by a global variable in the corresponding SAN model.

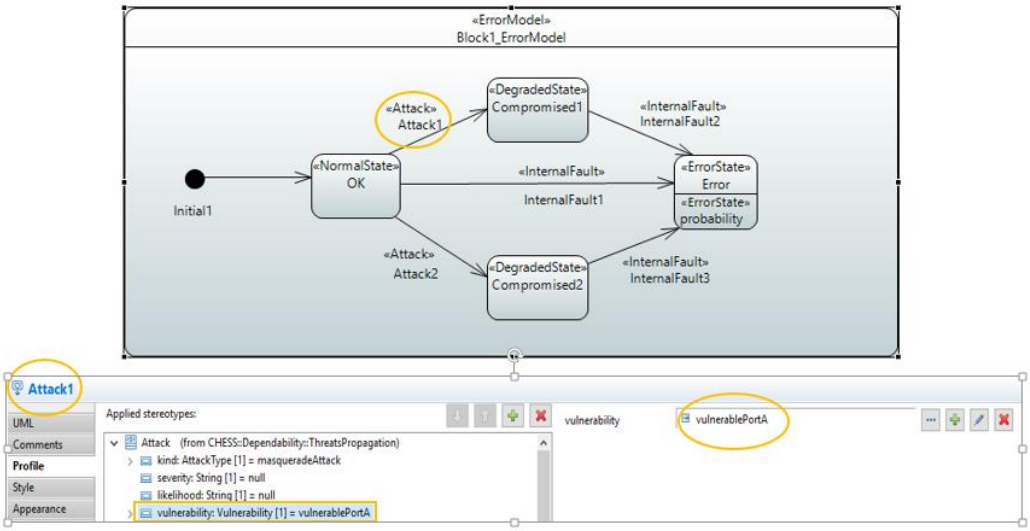

Figure 3: Error Model with Attacks

3.3 Attack Scenarios Modelling

Before modelling attack scenarios it is needed to add an adversary of the system in the model. In this context, an adversary is a UML Actor stereotyped as <<**Adversary**>>.

To model an attack scenario:

1. Create a new sequence diagram in the CHESS model (creates a new Interaction element)
2. Apply stereotype <<**AttackScenario**>> on the Interaction
    a. Optionally set “frequency” and “probSuccess” properties. If left unspecified -> global variables.
3. Create an anonymous UML Lifeline for the Adversary, and additional Lifelines for each block instance involved in the scenario where for instance here we mean a SysML Part of a composed Block (set the “Represent” property to the corresponding instance in the model)

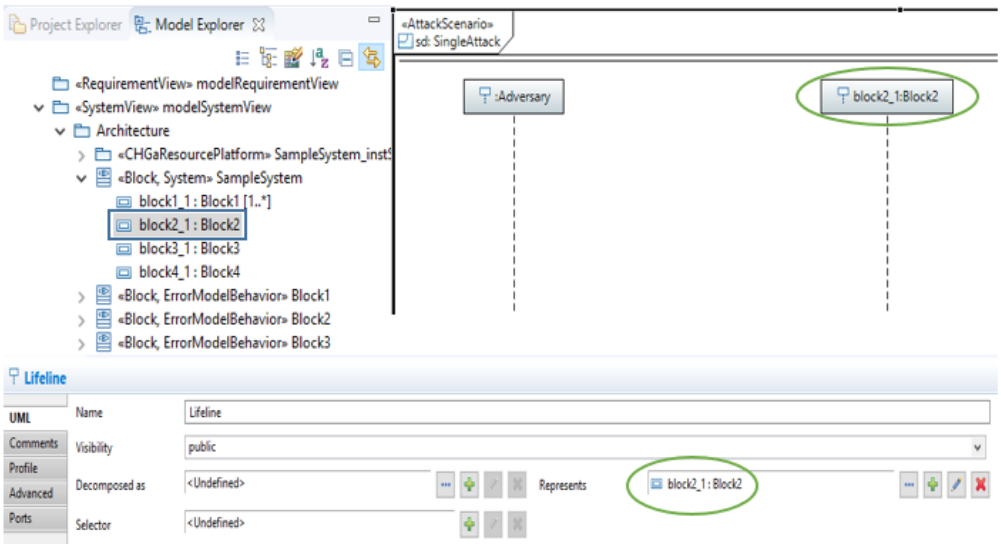

Figure 4: Attack Scenario: Lifelines

4. Add attack fragments to the scenario. A fragment can be one of:
    a. **Single Attack**: a UML Message stereotyped as <<**Attack**>> from the lifeline representing the Adversary to a lifeline of a vulnerable block. It is mandatory to specify the value of the "vulnerability" property. Only a Port stereotyped as <<Vulnerability>> and owned by the representing block is a valid value for this property.

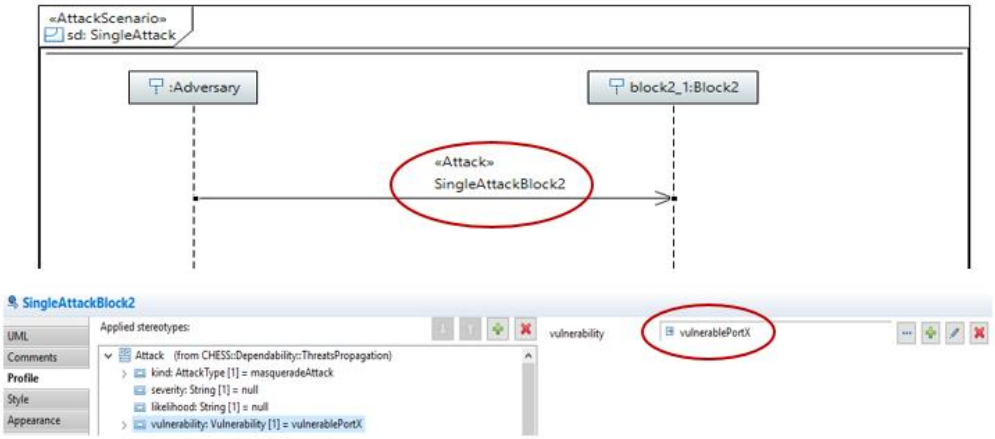

Figure 5: Attack Fragment: Single Attack

    b. **Alternative Attacks**: modelled as a UML CombinedFragment with operator equal to "alt". This CombinedFragment can contain two or more UML elements of type InteractionOperand:
        i. For each InteractionOperand specify its Guard (UML InteractionConstraint): create a UML LiteralString as "specification" of the InteractionConstraint Guard, the value of this LiteralString will become a global variable in the SAN model corresponding to the attack scenario.
        ii. A particular keyword can be used for the LiteralString value: "else". It can be used only once for each Alternative Attacks fragment and it must be used only for the last InteractionOperand.
        iii. Each InteractionOperand will contain exactly one Single Attack fragment (see case a.).

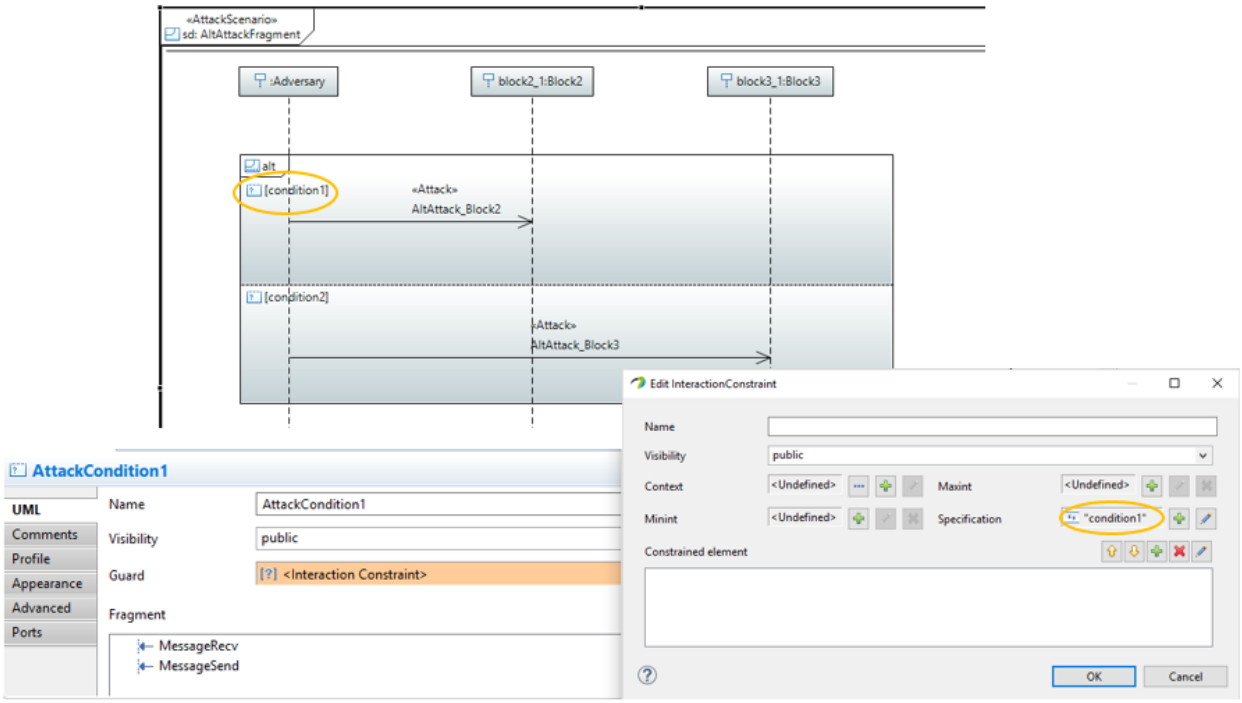

Figure 6: Attack Fragment: Alternative Attacks

c. **Optional Attack**: modelled as a UML CombinedFragment with operator equal to "opt". This CombinedFragment contains one UML element of type InteractionOperand:
   i. Specify the Guard of the InteractionOperand as for case b.
   ii. "else" keyword cannot be used.
   iii. The InteractionOperand will contain a Single Attack fragment.

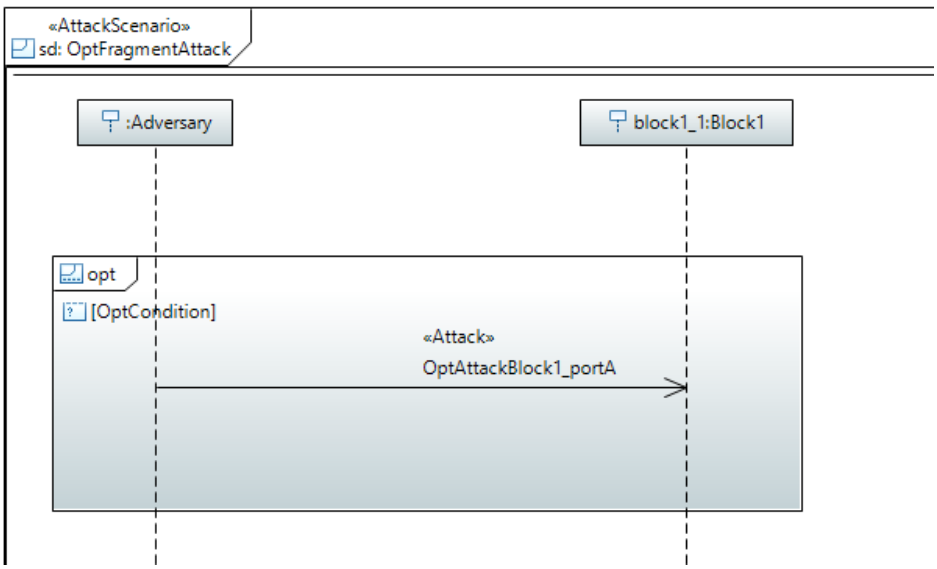

Figure 7: Attack Fragment: Optional Attack

d. **Loop Attack**: the semantics of a Loop Attack is to model a Single Attack repeated for each instance of a vulnerable block. The lifeline target of these attack messages must represent an instance of a block with multiplicity "*" or "1..*". This fragment is modelled as a UML CombinedFragment with operator equal to "loop". This CombinedFragment contains one UML element of type InteractionOperand:
    i. The Guard needs no Specification, instead set its property "maxInt" as UML LiteralInt or UML LiteralString and set their value. If using a LiteralString its value will become a global variable in the SAN model corresponding to the attack scenario.
    ii. The InteractionOperand will contain a Single Attack fragment.

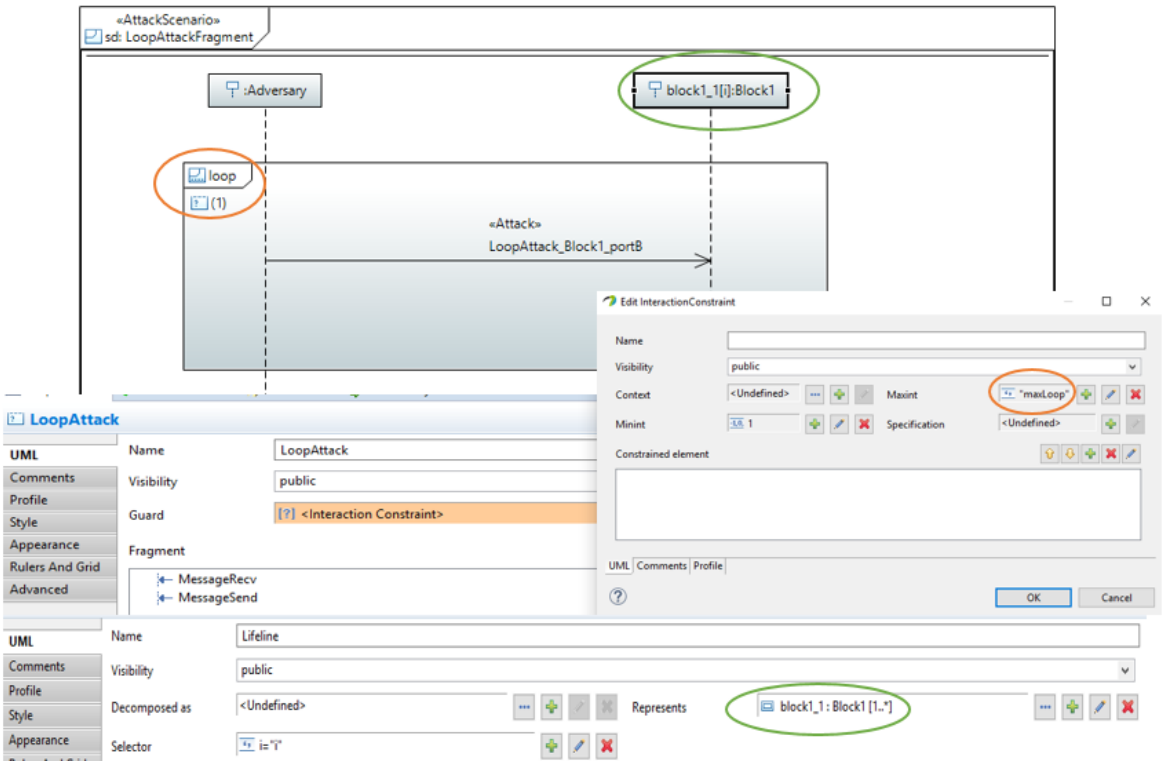

Figure 8: Attack Fragment: Loop Attack

e. Additionally it is possible to model **delays** between attacks. A delay is a UML DurationConstraint modelled as follows:
    i. Create a UML DurationConstraint in the sequence diagram.
    ii. Specify an attack fragment (described in bullets a. to d.) as constrained element of this DurationConstraint: semantically it is interpreted by the CHESS extended Toolset as a delay that occurs before the constrained element.
    iii. Specify the duration of the delay: check the "specification" of the DurationConstraint: it's a DurationInterval, it's "min" and "max" specify the range of the interval. Set them to the same value using UML LiteralString as expression ("expr") of "min" and "max". This value is again a global variable in the SAN model corresponding to the attack scenario.

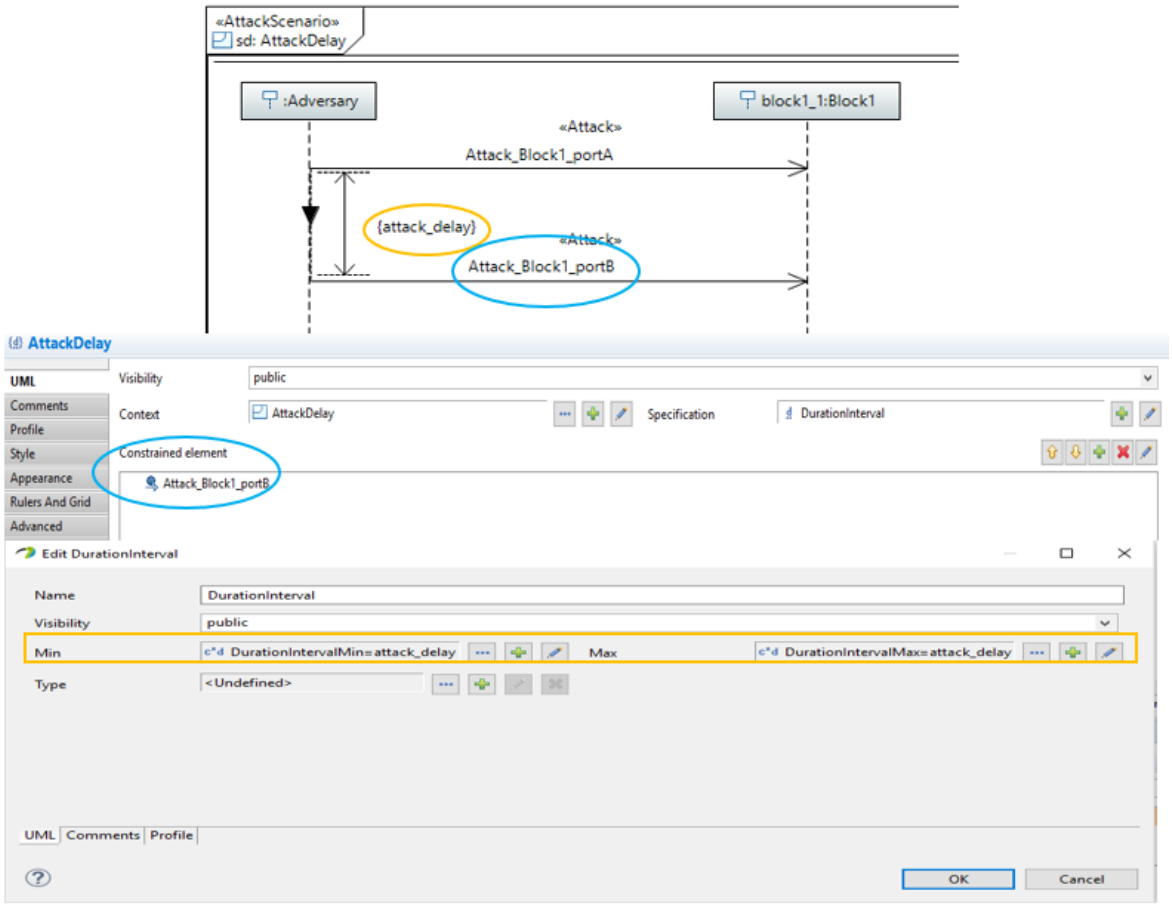

Figure 9: Attack Fragment: Delay

An Attack Scenario is composed of one or more fragments, to keep the order of the attack messages is highly recommended to use UML GeneralOrdering by connecting each start message (UML MessageSend) with the next one in sequence.

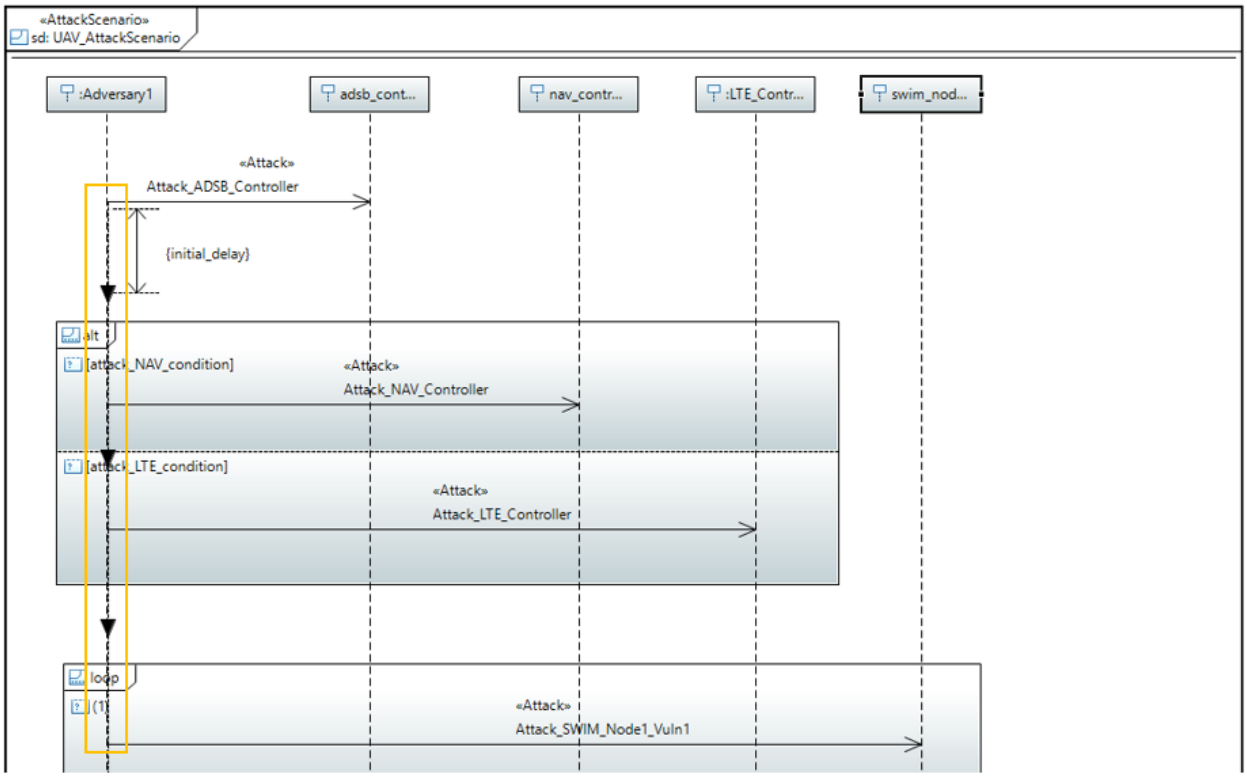

Figure 10: Attack Scenario and GeneralOrdering of Attack Messages

4 Generate the Mobius Project

It is possible to generate Mobius models for the Cyber Security elements modelled in CHESS. Make sure to set Mobius Projects directory in the preferences page (menu: Window -> Preferences -> CHESS -> Mobius Preferences)

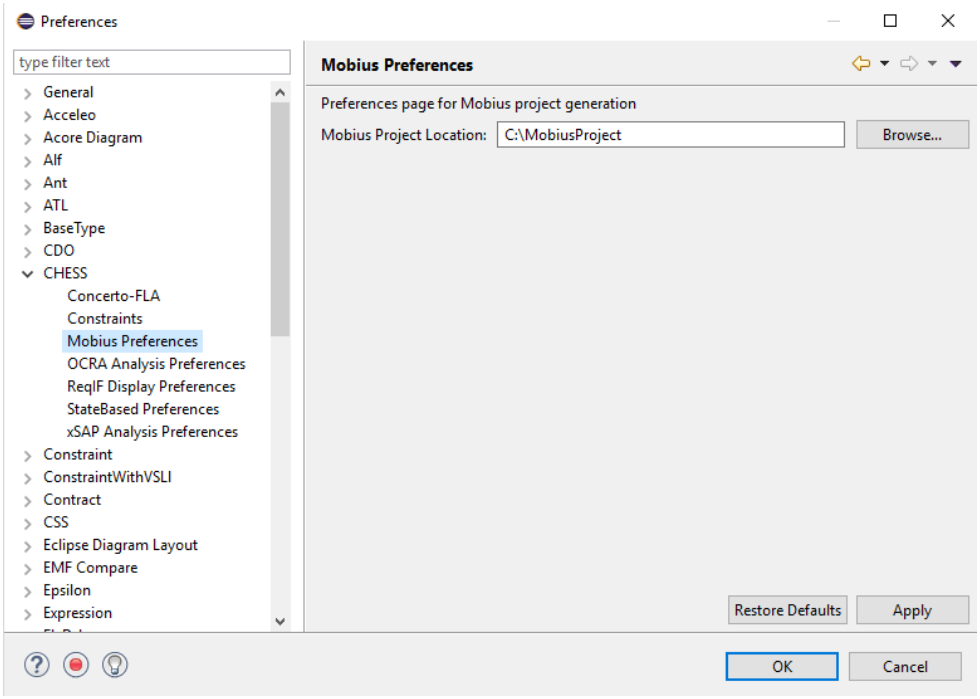

Figure 11: Preference Page

Then follow these steps:

1. Model analysis context for Cyber Security Analysis:
    a. Build model instances as in [1] section 9.3.1.2.
    b. Create a UML Class stereotyped as <<CyberSecurityAnalysis>> in the DependabilityAnalysisView Package. Set the value of "platform" property as the root of the instances (UML InstanceSpecification) generated in step a.

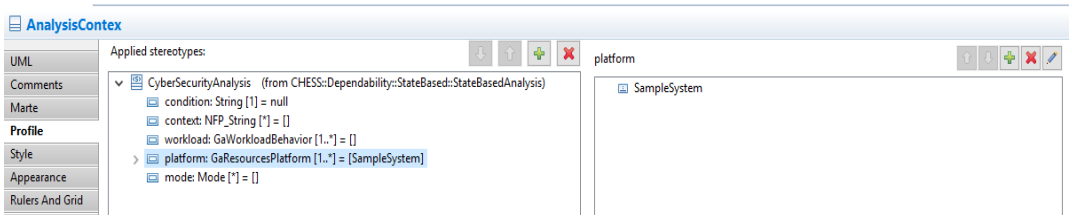

Figure 12: Analysis Context

2. Launch "Generate Mobius Project" from CHESS->Analysis->Dependability Menu and select the Analysis Context:

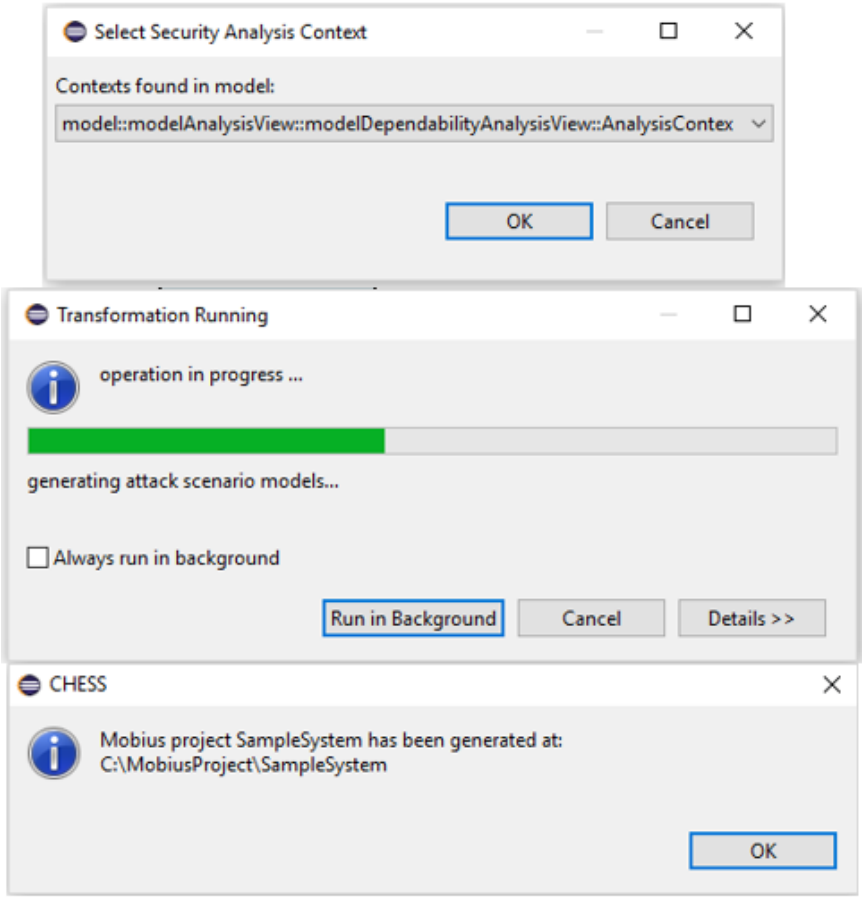

Figure 13: Launch Mobius Project Generation

5 CHESS to Mobius Transformation Overview

In order to generate a valid Mobius project a set of transformation in Acceleo (https://www.eclipse.org/acceleo/) has been designed and developed to transform entities from the CHESS model to SAN or REP/JOIN models in Mobius. The following list sums up this transformation process:

1. **Atomic Component** (SAN Model): SysML Block no further decomposed and stereotyped as <<ErrorModelBehavior>> with one or more Port stereotyped as “Vulnerability” and StateMachine stereotyped as <<ErrorModel>> and modelled as in section 3.2
2. **Composite Component** (REP/JOIN Model): SysML Block with further decomposition.
3. **Attack Scenario** (SAN Model): Interaction stereotyped as <<AttackScenario>> and modelled following the guidelines of section 3.3
4. **Vulnerable Component** (REP/JOIN Model) SysML Block transformed as Atomic Component and involved in one or more Attack Scenario

Additionally the transformation generates a Mobius Project File (.prj) to ensure that the project is recognized and read by Mobius correctly.

6 References

[1] https://www.polarsys.org/chess/publis/CHESSToolset_UserGuide.pdf.

[2] https://www.polarsys.org/chess/publis/CHESSMLprofile.pdf.