Modular model construction approaches for complex and interconnected systems

Leonardo Montecchi
University of Florence
leonardo.montecchi@unifi.it

SDCI 2012 Winter School: Hot Topics in Secure and Dependable Computing for Critical Infrastructures

January 15^{th}-19^{th}, Cortina D'Ampezzo, Italy

Model-based dependability analysis

- A <u>model</u> of the system is constructed
  - Abstraction of the system that highlights features that are relevant for the analysis and neglects the other details
- Advantages of model-based analysis
  - It does not require to exercise a real instance of the system
  - Allows "what-if analysis" and sensitivity analysis
  - Allows to assess the system in extreme conditions
- Common models that are used in dependability analysis
  - Combinatorial models
    - Fault-Trees (FT), Reliability Block Diagrams (RBD)…
  - State-based models
    - Continuous Time Markov Chains (CTMC)
    - Stochastic Petri Nets (SPN)
    - …

Challenges in LCCI modeling

- Challenges
  - Very high number of components
  - Components are interconnected, leading to complex dependencies and interdependencies
  - Dependencies evolve during time (also because of external events)
    - The failure of a router in a communication network
    - A tree touching a power transmission line…
- Challenges are getting also harder by the increasing adoptions of
  - Decentralized architectures
  - Wireless connectivity
  - Mobile application scenarios
  - COTS components
- Moreover: infrastructures are also interdependent on each other
  - The power distribution infrastructure relies on network communication for monitoring and control
  - Network communication needs electric power

Two complementary approaches

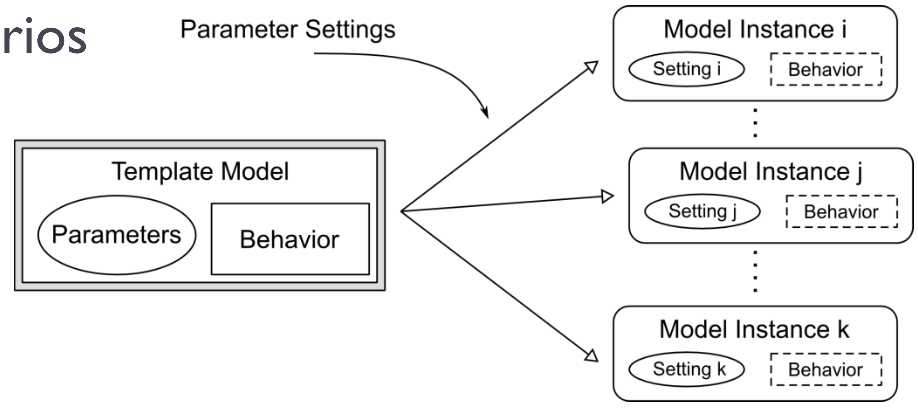
- The following properties of models are very welcome in facing these challenges:
  - Scalability – To address the high number of components and connections
  - Reusability – LCCIs are often characterized by groups of components having similar behavior
  - Modularity – Facilitates handling the interdependencies
  - Maintainability – Changes are easier to address: LCCIs evolve over time and their lifetime is typically several years
- In the following, two complementary approaches to achieve these properties are presented
  - System decomposition and modular model construction
  - Automated model construction by transformations

System decomposition /1

- Main "building blocks" of the system are identified
- Usually, the system is decomposed considering
  - Different components (System-level decomposition)
  - Different layers and functionalities within components (Component-level decomposition)
    - User layer: Models the interaction of the user (if any) with the component
    - Application layer: Models the main functionalities of the component
    - Architecture layer: Models the dependability behavior of the component
- <u>Interfaces</u> between submodels should be carefully identified
  - Submodels should communicate only through those interfaces
  - The behavior of submodels should be independent of each other

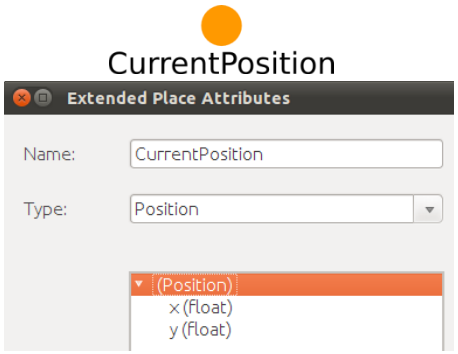
Template models

- <u>Parameters</u>
  - Often system components have similar functionalities, but different operating conditions and setup parameters
  - The behavior of submodels should not depend on those parameters
- "Template" submodels are built, based on a set of parameters
  - These templates are then instantiated multiple times with different parameters
  - And composed to obtain the overall model
- Advantages:
  - Easier to evaluate different scenarios
  - Possible mistakes in model construction are reduced
  - Easier to modify and maintain the model (changes to template are propagated)

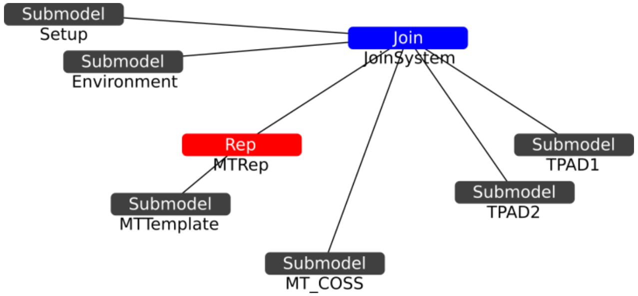


SAN implementation

- Stochastic Activity Networks (SAN)
  - Graphical formalism that extends Stochastic Petri Nets
  - Some useful features that can be exploited for this approach
- Extended places
  - Not limited to hold an integer number of tokens
  - Can hold C++ basic types, structures, arrays, or a combination of those (e.g., an array of structures)
  - Very useful to hold submodel parameters



- Join/Replicate composition formalism
  - Allows to compose different instances of submodels at multiple levels
- A special submodel "Setup" sets the actual parameters of instances

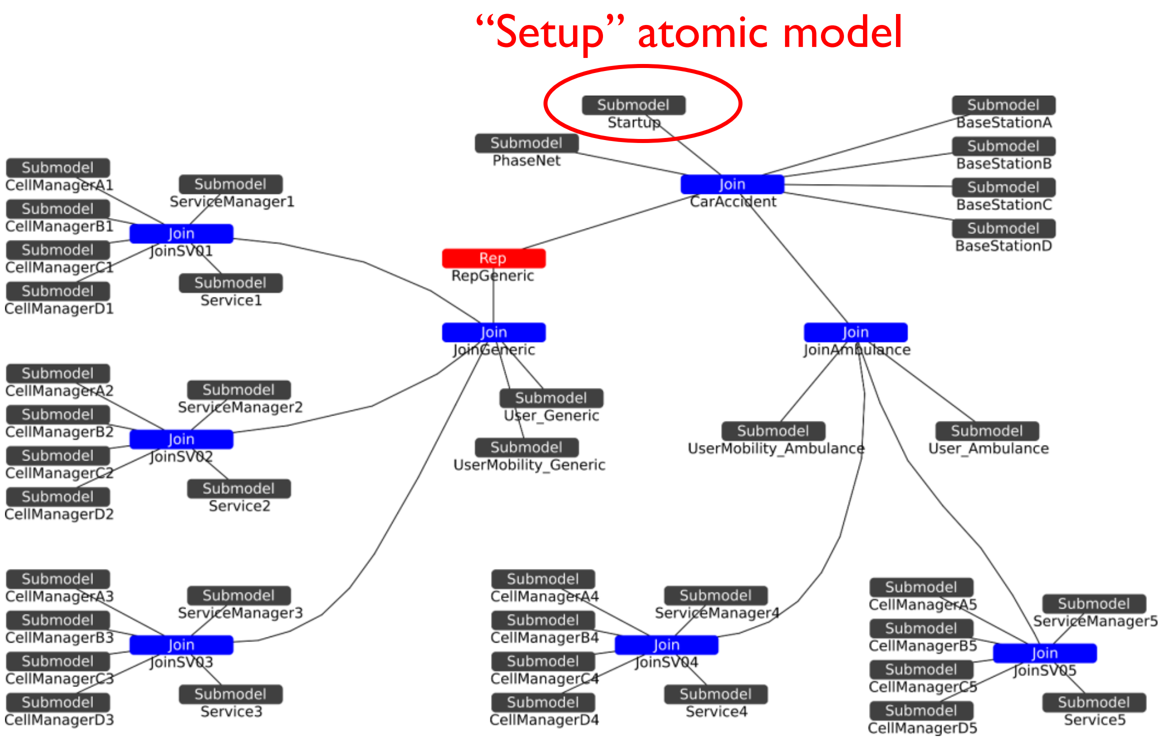


Application to LCCIs

- This kind of approach is tailored to LCCIs, and has already been used to model kind of systems
- HIDENETS "Car-accident" use-case

- <u>40</u> model instances
- Based on <u>10 templates</u>

- Services:
  - Different load factors
  - Possibly also different based on the cell
- Base stations:
  - Different locations
  - One is subject to outages while the other are not



CBD and MDE methodologies

- The second approaches takes advantage of two popular software development methodologies:
- Component-Based Development (CBD)
  - The system is obtained by composition of a predefined set of components, having well-defined interfaces
- Model-Driven Engineering (MDE)
  - Models are considered the main entities in the development process
  - The system is described using an high-level engineering language (UML, AADL, SysML…)
  - Then, artifact are generated by <u>automatic transformations</u>
    - Code
    - Representation in other modeling languages
    - <u>Analysis models</u>

Dependability analysis in MDE and CBD

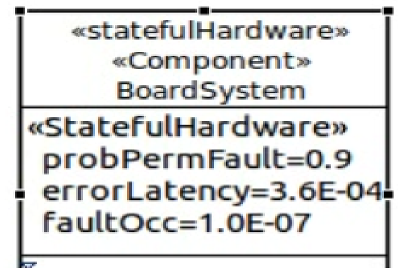
- Workflow:
  - The functional model of the system is built by composition of several components and interfaces, using UML-like languages
  - The model is then enriched with dependability attributes
  - Automated transformations generate the analysis model
  - The model is solved using the analysis tool, and results are back-annotated in the original model
- Incremental process: the resulting model can be used as input to subsequent analyses
- Which dependability properties should be represented in the high-level modeling language?
  - No agreed answer or standardized solution yet
  - Standards exist in other domains: real-time and schedulability ("MARTE" profile), quality of service ("QoS&FT" profile)

CHESS Dependability Profile

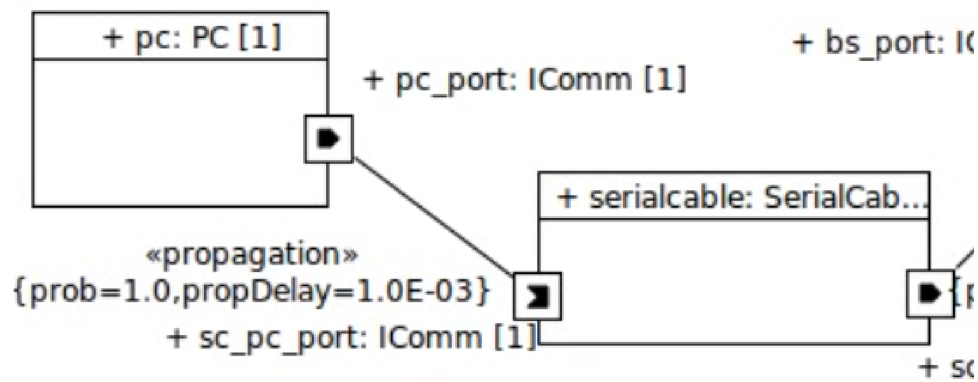
- CHESS (Composition with guarantees for High-integrity Embedded Software components aSsembly)
  - Objective: Define and develop a methodology based on CBD and MDE that allows to <u>specify</u> and <u>evaluate</u> non-functional properties
  - Real-time and embedded system domain
- CHESS Modeling Language (CHESS ML)
  - Language based on selected portions of UML, MARTE, and SysML
- CHESS Dependability Profile
  - Allows to specify dependability properties on CHESS ML models
  - Supports different dependability analysis techniques:
    - Fault-Tree Analysis (FTA)
    - Failure Modes, Effects [and Criticality] Analysis (FMEA and FMECA)
    - State-based analysis (using Stochastic Petri Nets)
    - Failure Transformation and Propagation Calculus (FTPC)

CHESS Dependability Profile: Key elements / 1

- Template stereotypes
  - Define generic components,
  - Represents a rapid way for users to provide dependability information on components



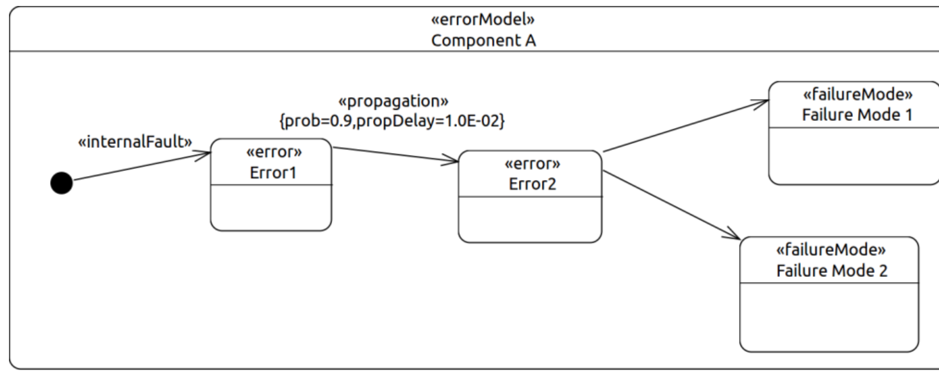
- Non-functional properties on connections and allocations
  - Specify how propagation takes place between connected components
  - Also propagation from hardware to software components



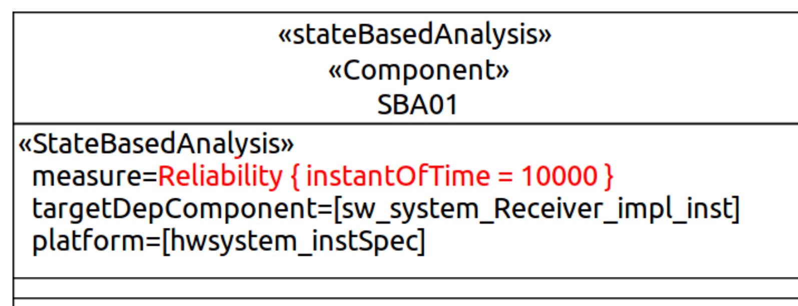
- Maintenance
  - Means to model more complex maintenance policies

CHESS Dependability Profile: Key elements /2

- CHESS Error Model
  - Defined as a particular kind of StateMachine diagram
  - Allows to define more complex behavior of components with respect to dependability
    - Internal faults
    - External faults
    - Errors
    - Failure modes

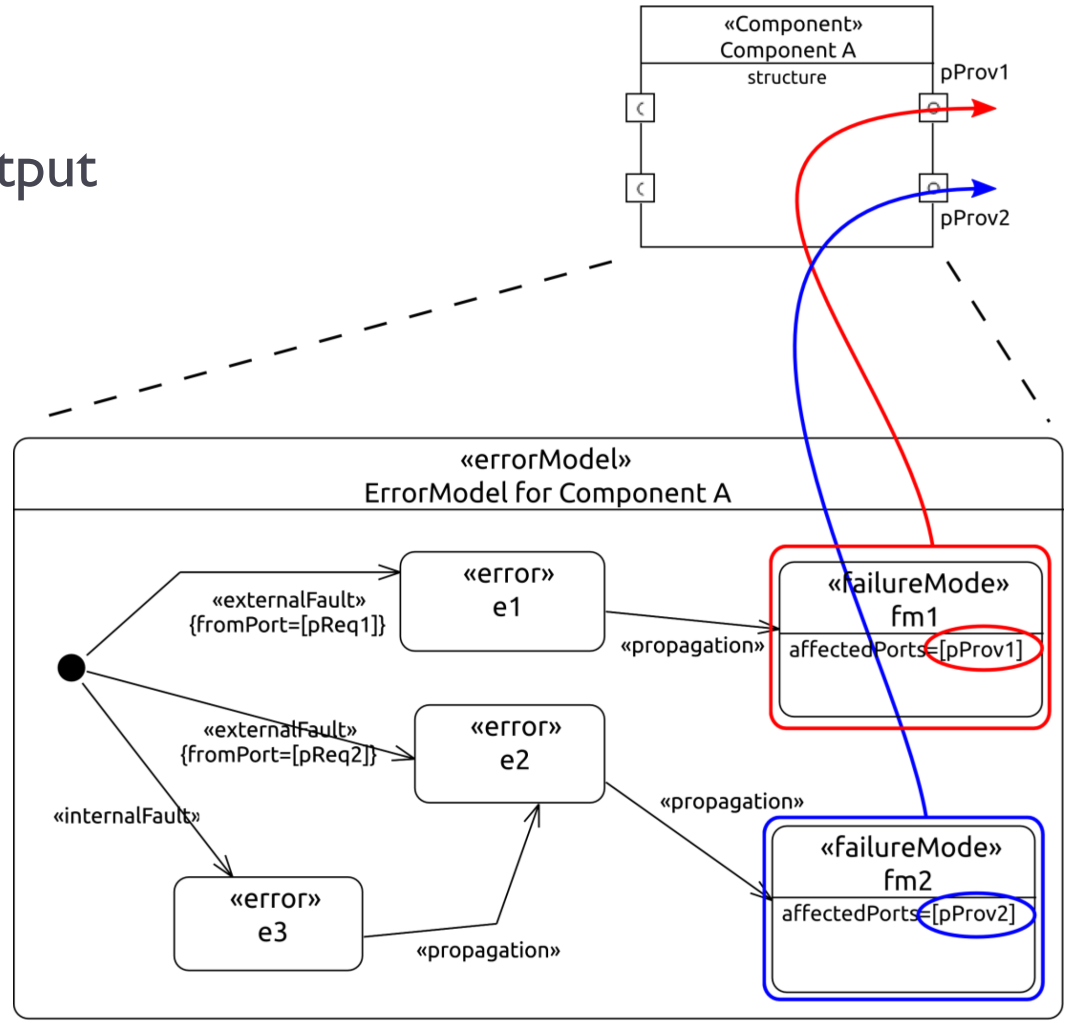


- Measures of interest
  - Supports the definition of different measures of interest
  - Currently implemented in transformations:
    - Instant of time reliability
    - Instant of time availability
    - Interval of time availability
  - W.r.t. to either a single failure mode or a whole component

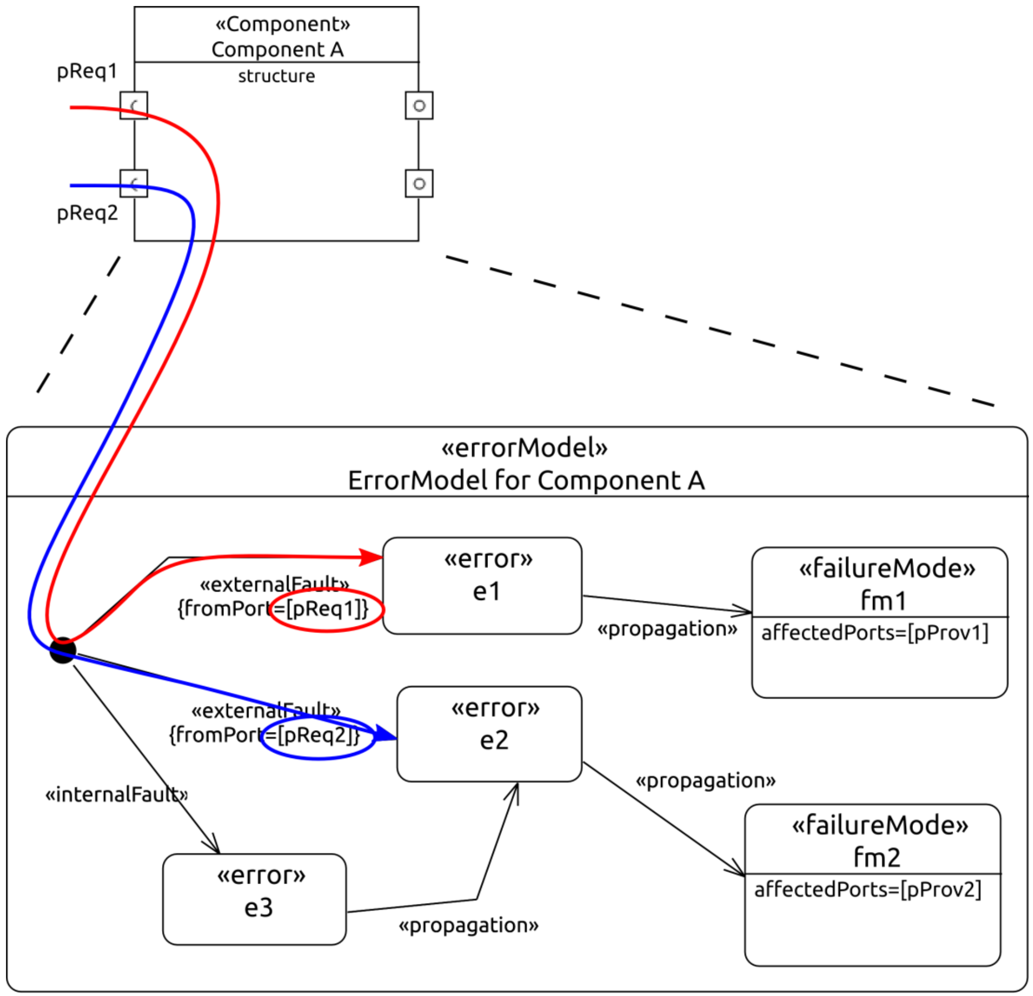


Linking the error model and component interfaces / 1

- Failure modes
  - Associated with output (provided) ports of components
- Maps failure modes to services provided by the component



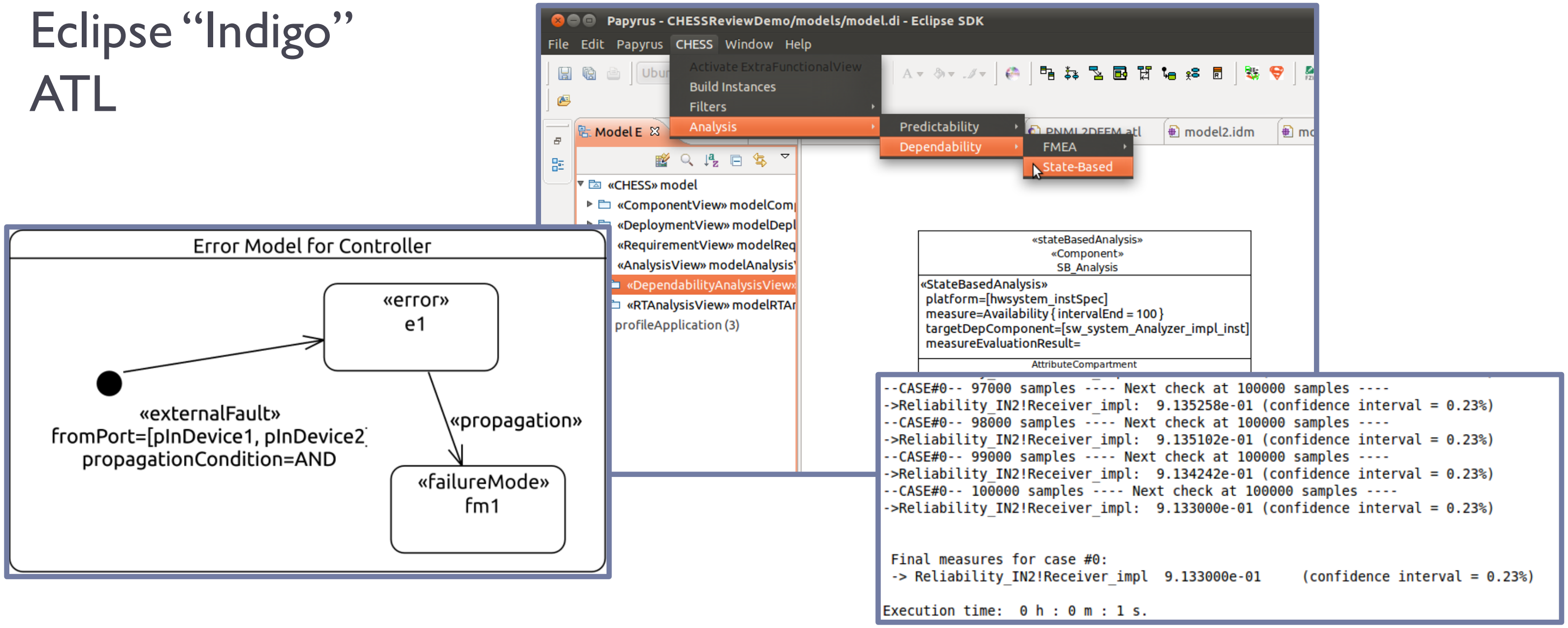
Linking the error model and component interfaces /2



- External faults
  - Associated with input (required) ports of components
- Maps external faults to services required by the component

CHESS State-based analysis plugin

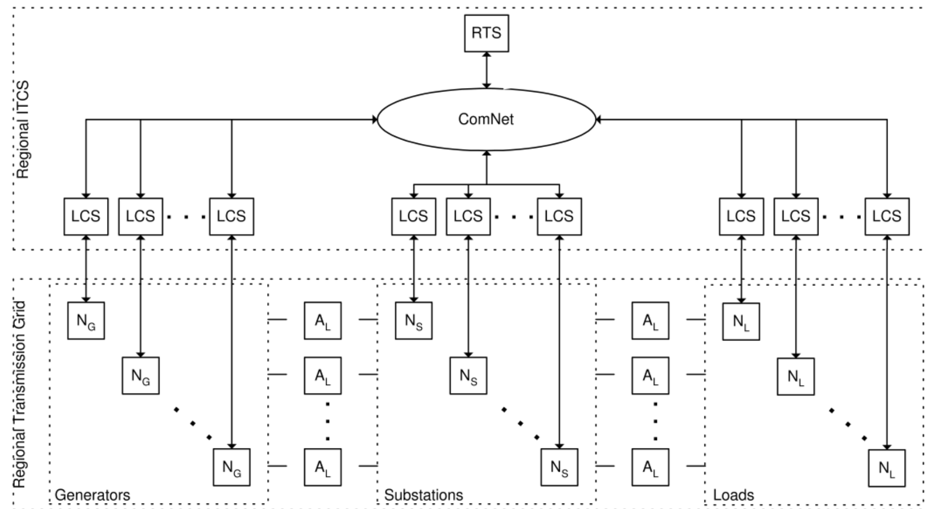
- Based on some of the latest technologies:
  - MDT/Papyrus
  - Eclipse “Indigo”
  - ATL



- Demonstration video is available online
  - http://chess-project.ning.com/page/videos-1
  - Set 2, Clip 2

Application to LCCIs

- System composed of many similar components and many interconnections
  - Example: Electric Power Infrastructure
- Model the components and interconnections using a CBD approach



- Add non-functional properties
- <u>Automatically derive propagation paths</u>
  - Facilitates in spotting "cascading" and "escalating" failures
- UML profile for Critical Infrastructures modeling?

Concluding remarks

- Presented two approaches to cope with system complexity in model construction
  - System decomposition and template models
  - Automated model generation by transformations

- Future work
  - Understand how these two approaches can be profitably combined
  - Extend the presented approaches to other contexts
    - LCCI (mainly for MDE-based approach)
    - Take into account aspects related to security
    - …

Thank you!

Questions?