Reliability Assessment of Safety-Relevant Automotive Systems in a Model-Based Design Flow

Sebastian Reiter, Michael Pressler, Alexander Viehl,
Oliver Bringmann, Wolfgang Rosenstiel

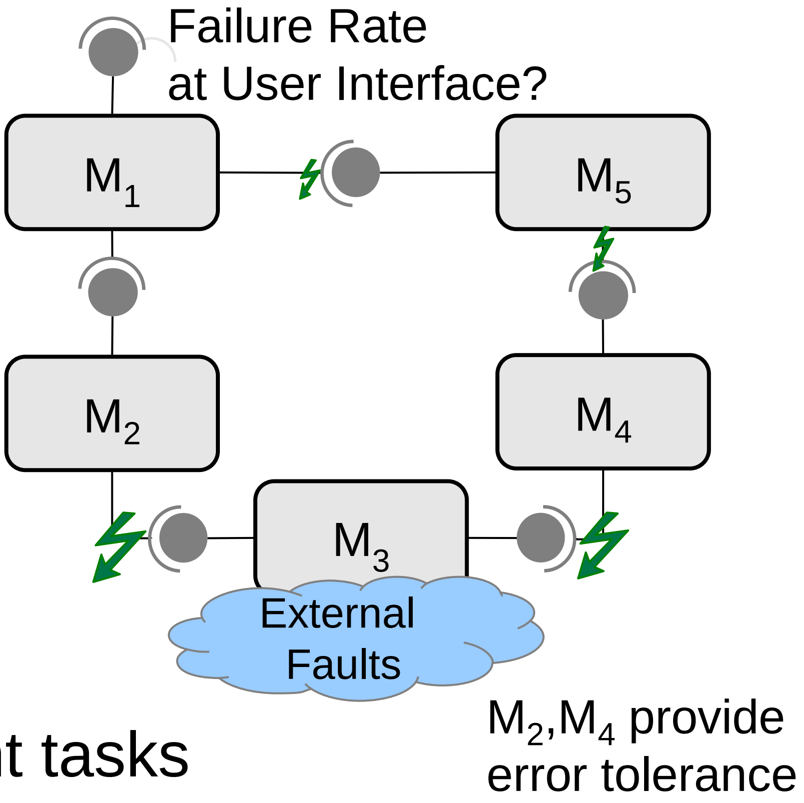
ASP-DAC 2013

24.01.2013

FZI FORSCHUNGSZENTRUM INFORMATIK

Motivation

- Software within vehicles has increased exponentially
  - More than 70 embedded platforms
  - System complexity and distribution increased
  - Synergistically interconnected
- Probability of operational errors increased similar
- Software covers safety-relevant tasks
- Erroneous delivered service could result in disastrous accidents



State-Of-The-Art - FMEA

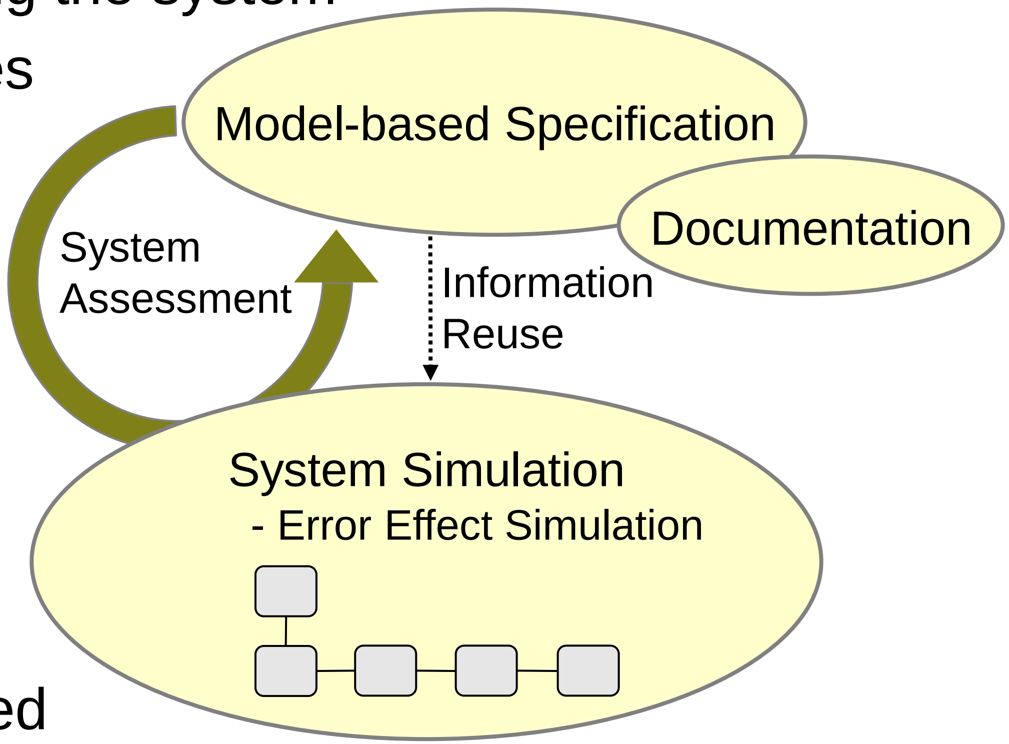
- Failure Mode and Effect Analysis (FMEA)
  - Recommended by the IEC 61508 [1] and the ISO 26262 [2]
- Basic idea
  - Identification of potential faults
  - Determination of the resulting error effects
  - Rating according to the severity and occurrence rate
- Drawbacks
  - Based on subjective estimations
    - Assessment of error tolerance
    - Hard to detected internal system dependencies and correlations
  - System knowledge is provided by the involved people
    - Challenging to share with component suppliers
    - Mostly not possible to reuse system knowledge
    - Cooperation of involved people is very important

Goal - Enhanced FMEA

- Reduce the subjective assessment
  - Improve the accuracy
    - Failure rates assessment
  - Assess influence of error tolerance mechanism
  - Detection of internal system interdependency
- Share information between component suppliers
  - Facilitate the collection of expert knowledge
  - Reuse of existing system knowledge
- Automated system analysis
  - Accelerate analysis in re-design loops
- Close integration into existing design flows
  - Reduce overhead of the analysis
  - Support the complete design process

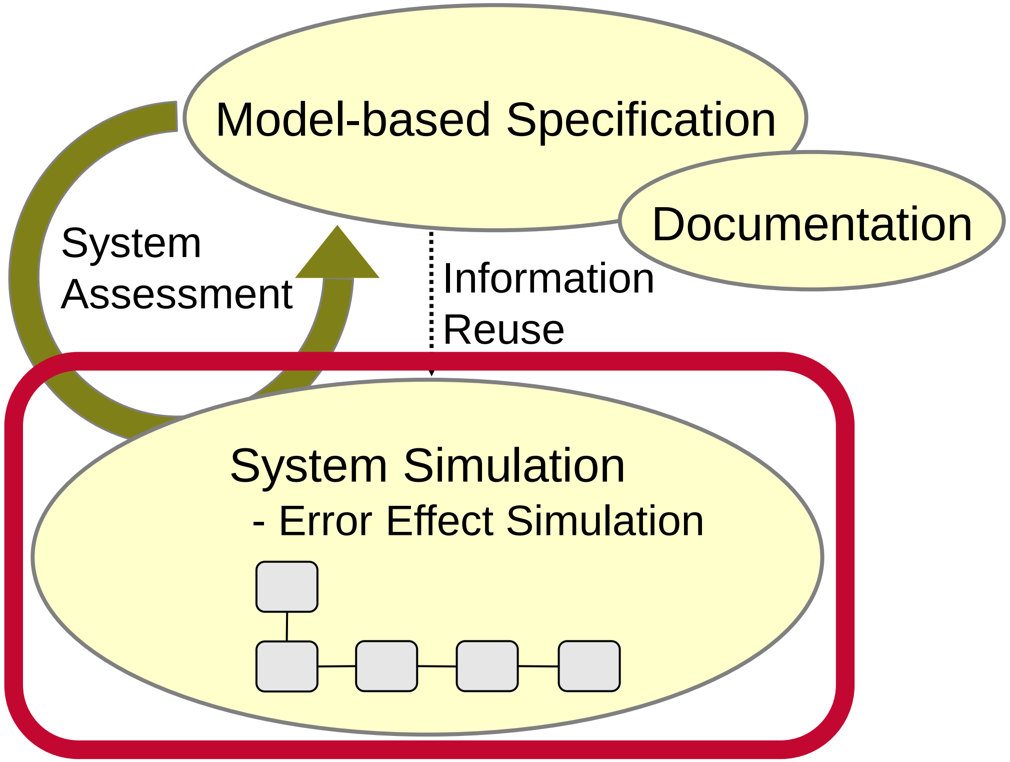
Approach

- FMEA support by system simulations
  - Simulate errors affecting the system
  - Use of virtual prototypes
  - Quantitative reliability assessment
  - Modular simulation framework
- Integration into existing model-driven design flows
  - Access already specified information
  - Link analysis results with system specification



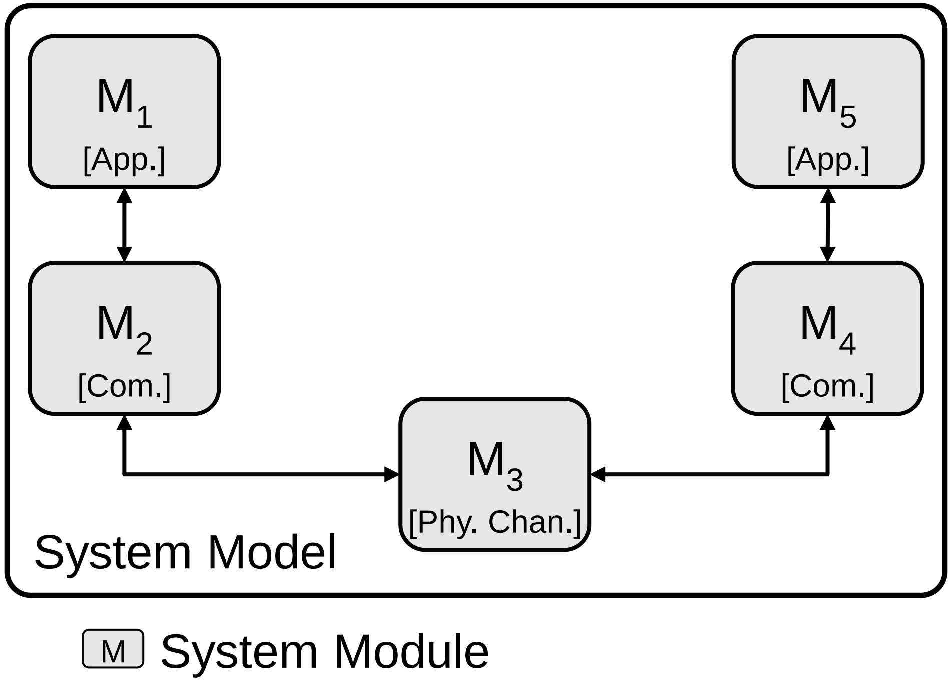
Error Effect Simulation

- Virtual Prototyping
- Error Stimulation
- Effect Monitoring
- Evaluation Platform
- Analysis Flow



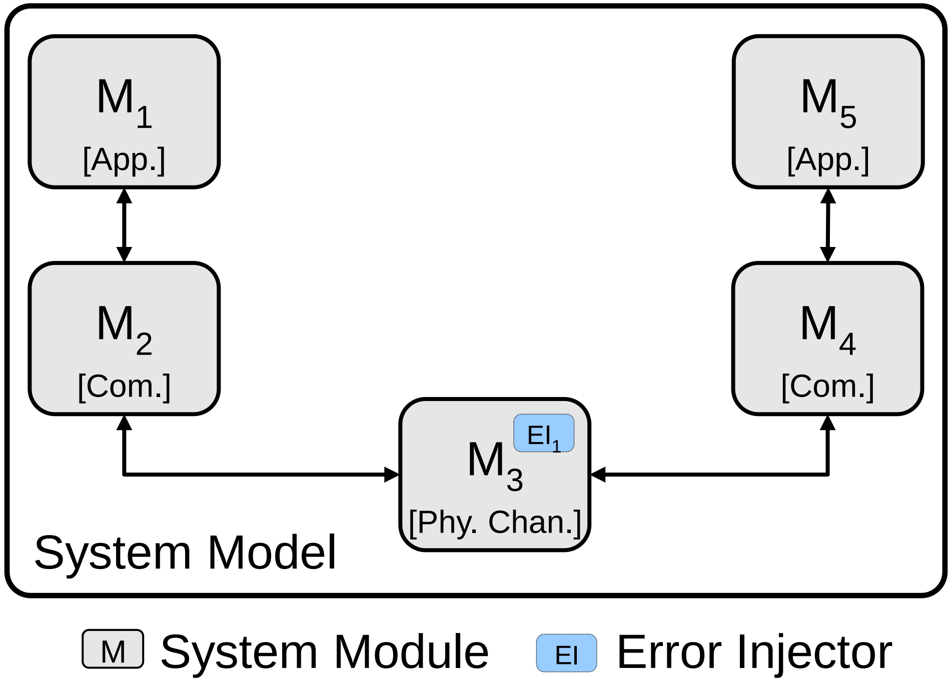
Virtual Prototyping

- A software based simulation kernel
  - Simulation of required system modules [M]
    - Functional and timing behavior
  - Event-driven simulation language SystemC [3]
  - Evaluation of hardware/software systems
  - No physical prototypes required
  - Support of different level of abstraction
    - Loosely timed
    - Cycle accurate
  - Applicable along the design process



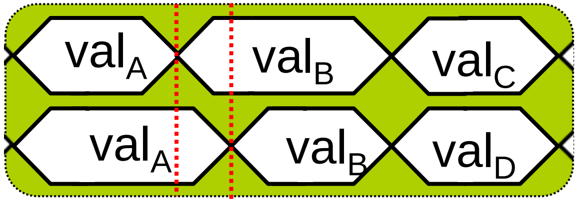
Error Stimulation

- Error injector module [EI]
  - Stimulates error within the virtual prototype
    - E.g. Bit-Flips, Cross-Talk or a Stuck-At errors
  - Supports four basic error modes
    - Modify content information
    - Modify timing behavior
    - Halt error mode
    - Complete loss of a signal
  - Combination for more complex errors
    - Erratic errors by content and timing corruption

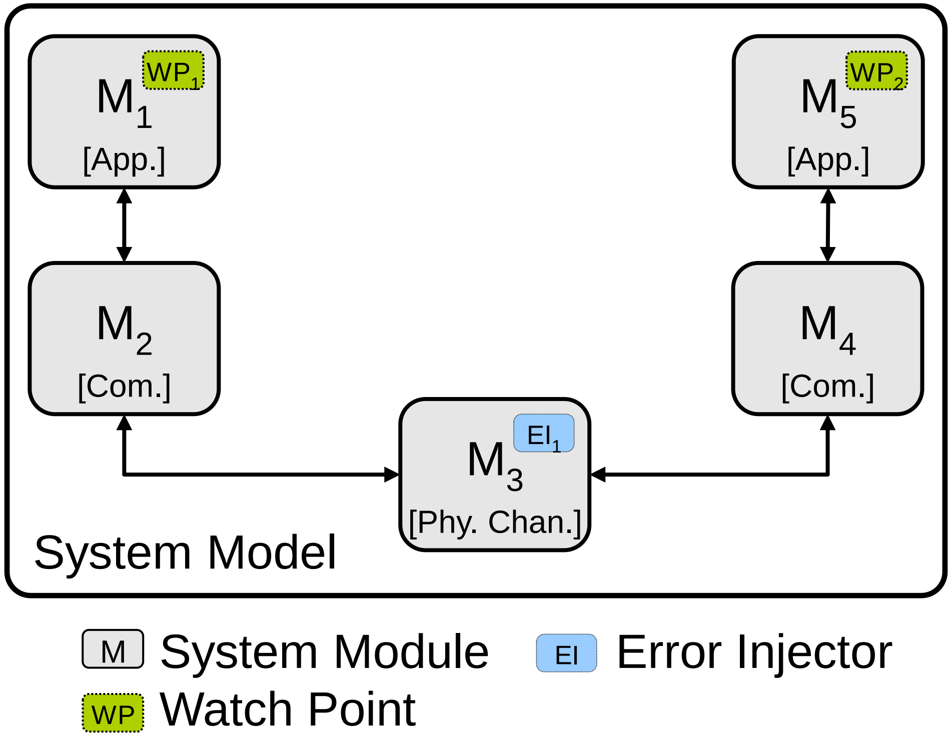


Effect Monitoring

- Watch points [WP]
  - Monitoring functional and timing behavior
  - Error prone system parts are neglected
  - Five failure modes are monitored

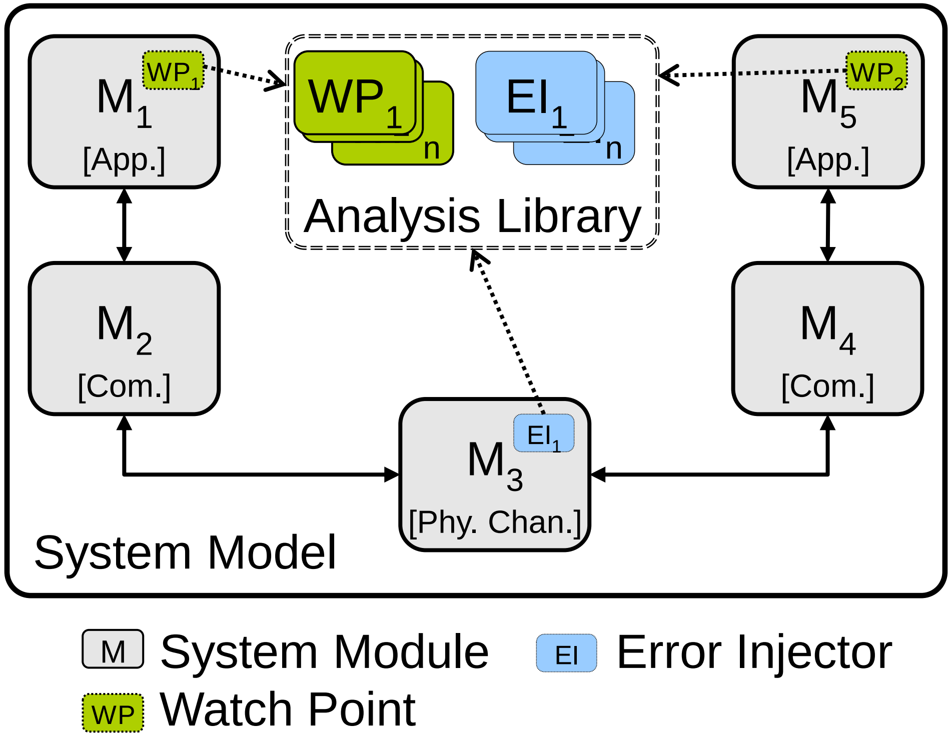


    - **Content failure**: Signal changes with correct timing but to an incorrect value
    - **Early / late failure**: Signal changes to the correct value but too early or too late
    - **Signal loss**: A signal change is missing
    - **Additional signal**: An additional signal change happens



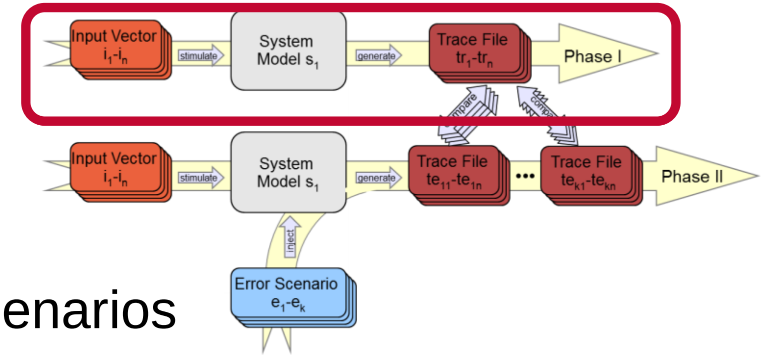
Evaluation Platform

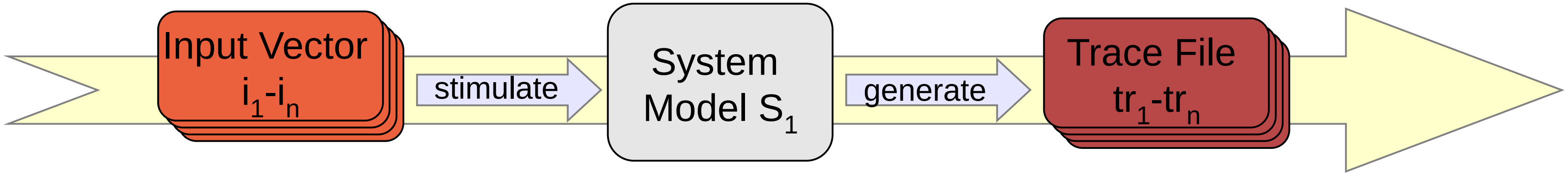
- Integration of the error injectors and watch points
  - Analysis library
    - Configured error injector instances
    - Watch point instances
    - Automatic generation by the model-driven tool chain
  - User intervention
    - Manual specification of references to the analysis library
    - Limited to a few lines of code



Analysis Flow

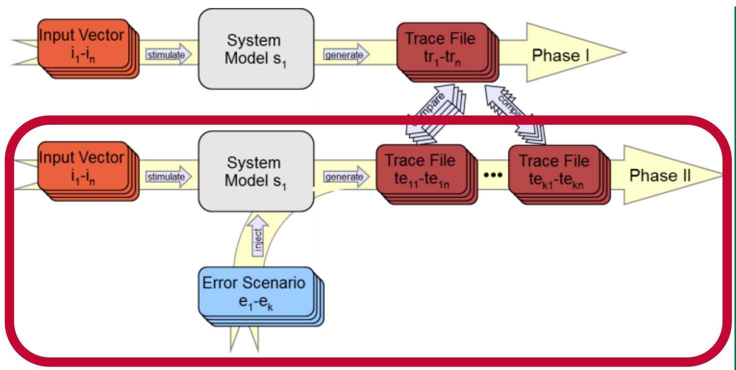
1. Reference trace generation
   - Error free system simulation
   - Usage of different input vectors
     - Evaluate system with different scenarios
     - Coverage directed test pattern generation

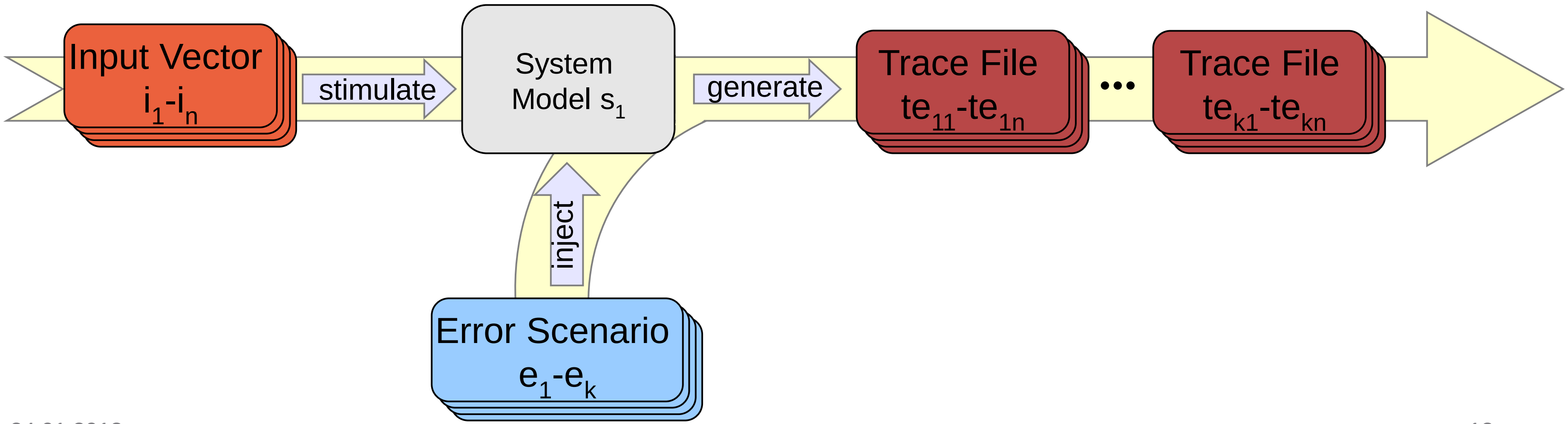




Analysis Flow

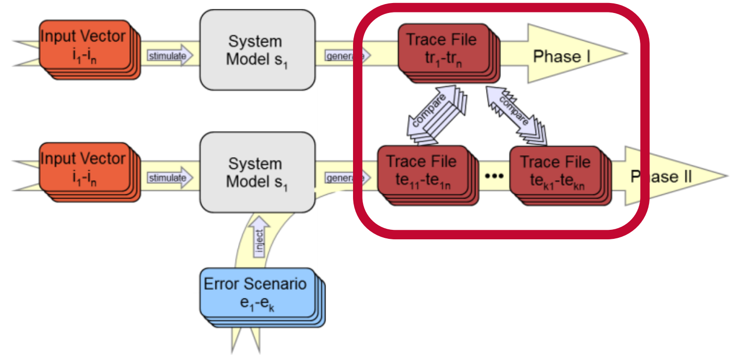
1. Reference trace generation
2. Repeated execution with different error stimulations
   - Error prone system simulation
   - Each error scenario is simulated with the complete set of input vectors



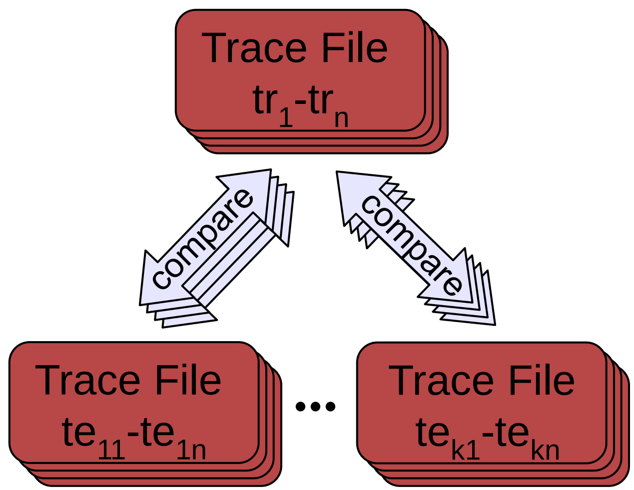


Analysis Flow

1. Reference trace generation
2. Repeated execution with different error stimulations

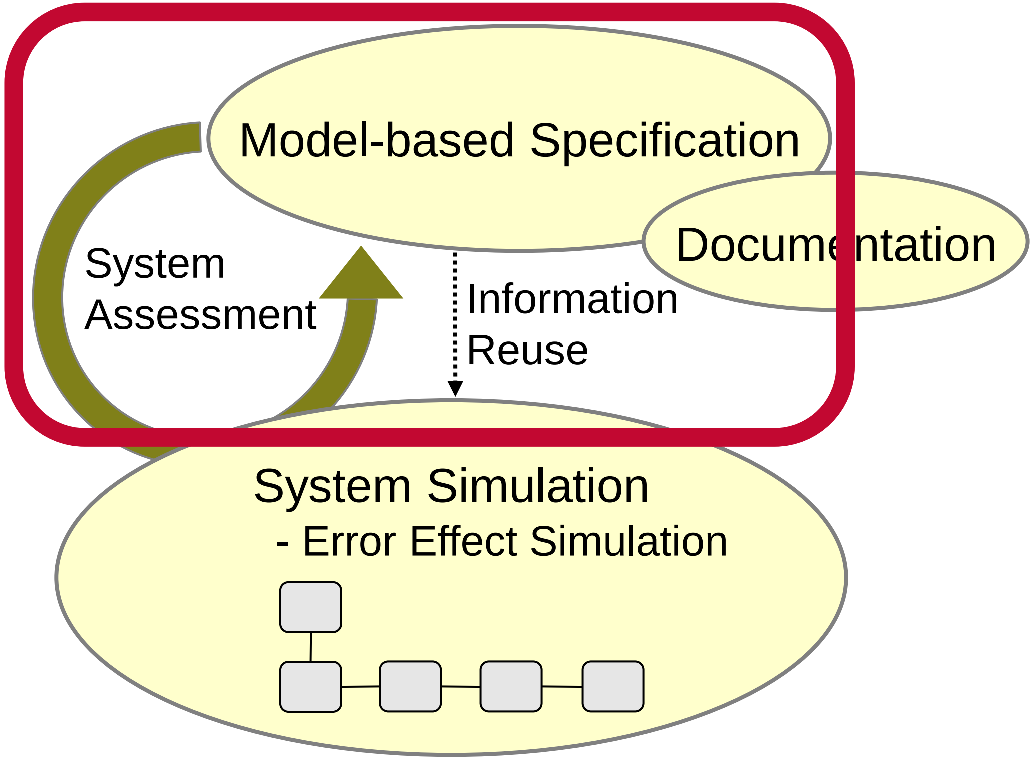


3. Comparison of reference trace file with error prone simulation results
   - Deviation will indicate potential failures
   - Comparison of trace files with the same input vector

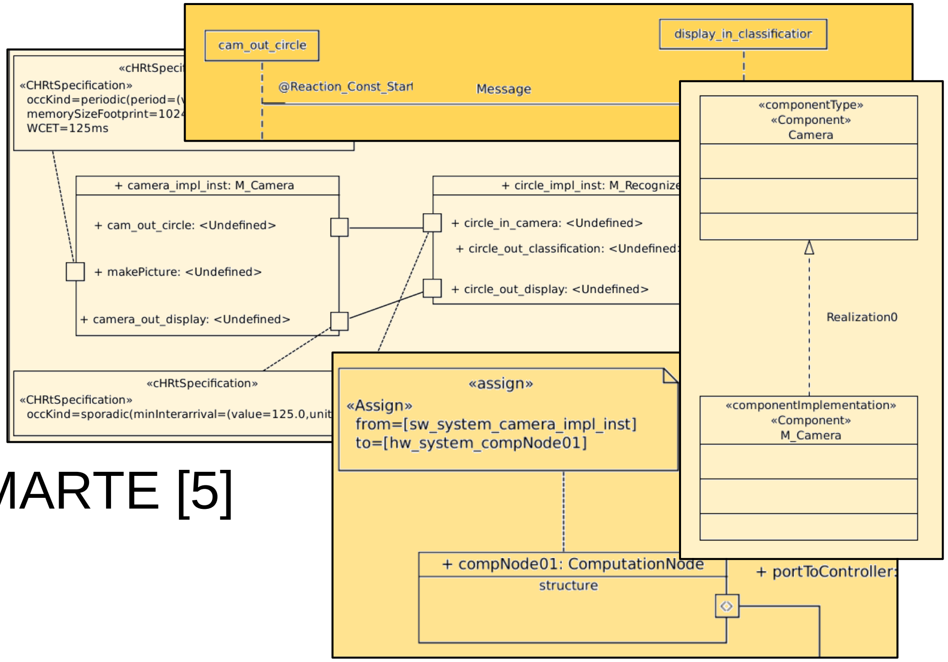


Design Flow Integration

- Model-Driven Tool Chain - A Survey
- Model-Centric Development
- Integration of the Analysis
- Modeling Framework Extensions



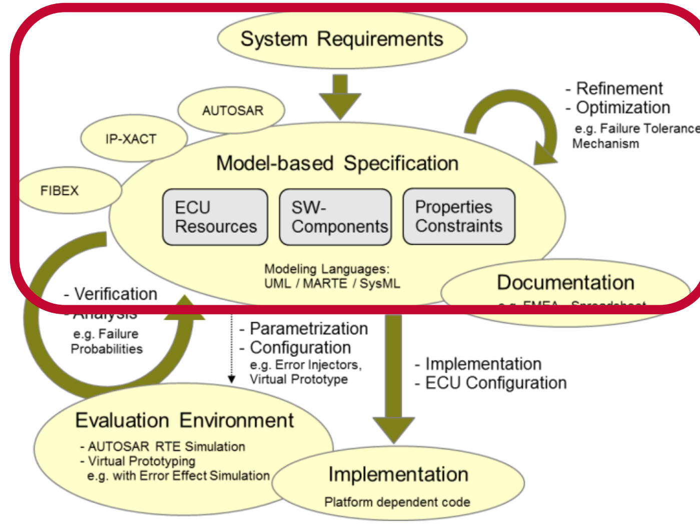
Model-Driven Tool Chain - A Survey

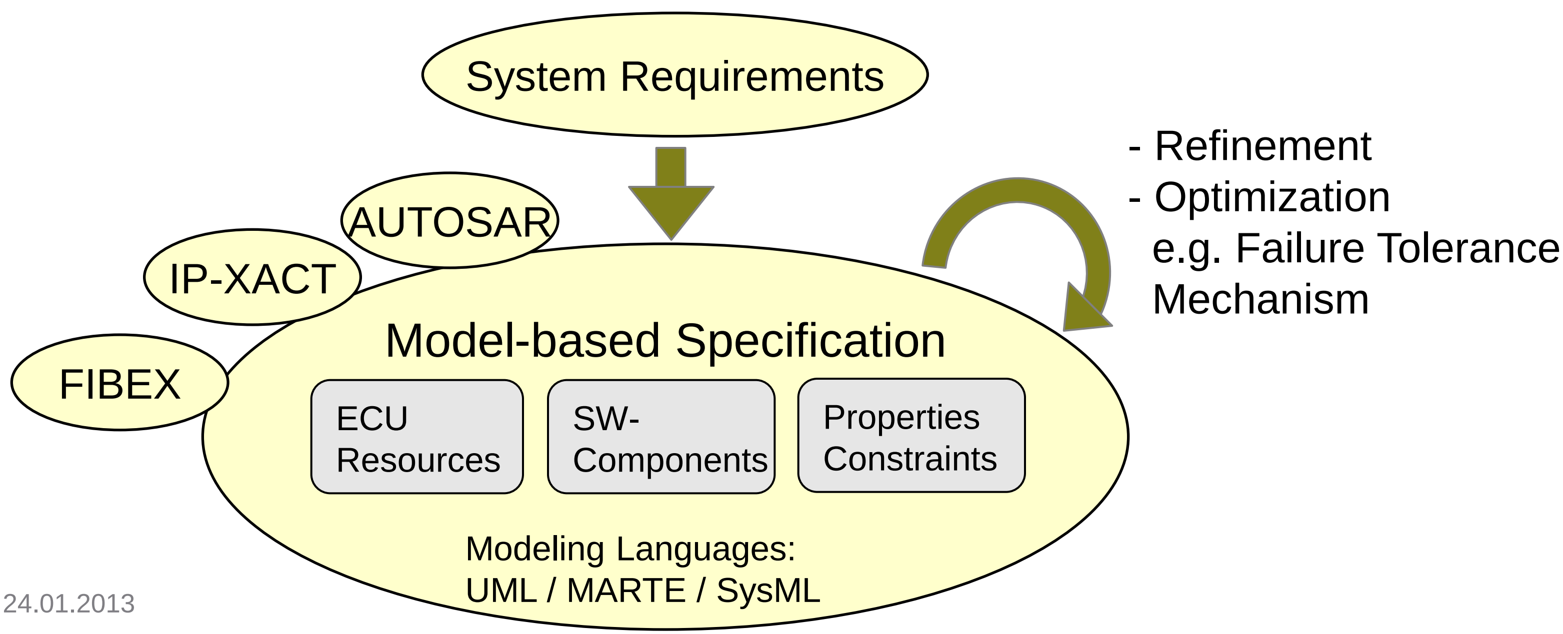


- Existing environment [4]
  - Software components
    - Types / Interfaces
    - Instantiation
    - Assembly
  - Hardware resources
    - Parameterization with MARTE [5] stereotypes
  - Deployment
    - Assignment of SW instances to HW resources
- Needed extensions
  - Reliability specification
  - Analysis specification
  - Information exchange with the error effect simulation
  - Parameterization of HW/SW instances

Integration of the Analysis

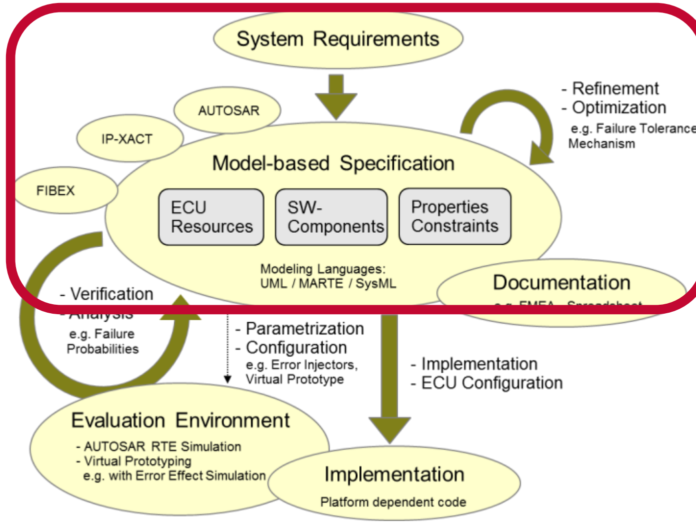
- Information base creation
  - Modeling tool chain
  - System configuration
    - Structural information
    - System parameters

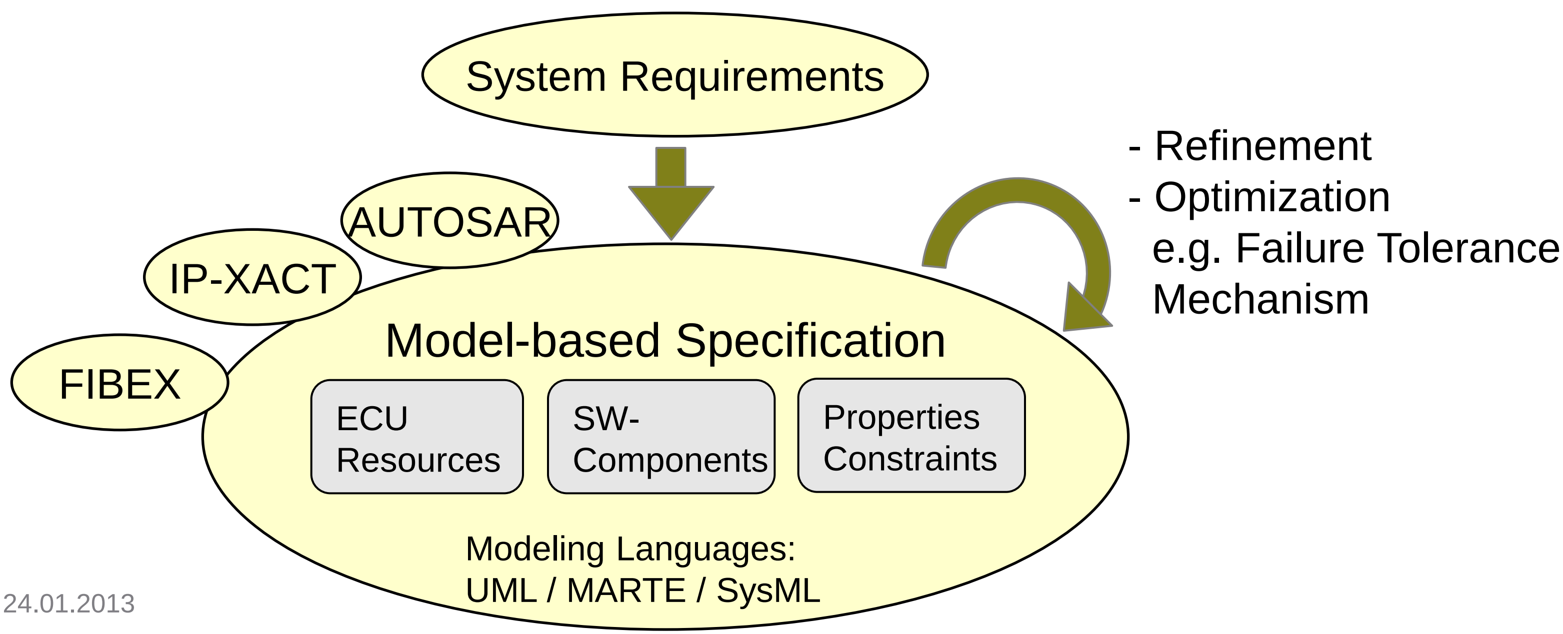




Integration of the Analysis

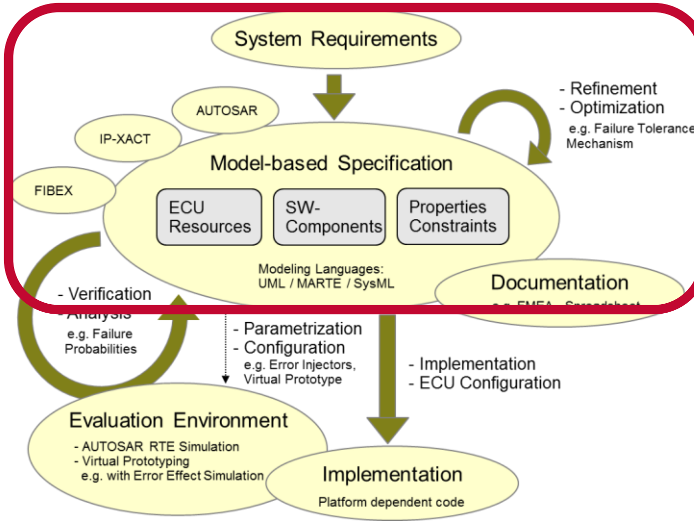
- Information base creation
  - Modeling tool chain
  - System configuration
  - Reliability information
    - Fault, Error, Failure specification
    - Analysis Configuration

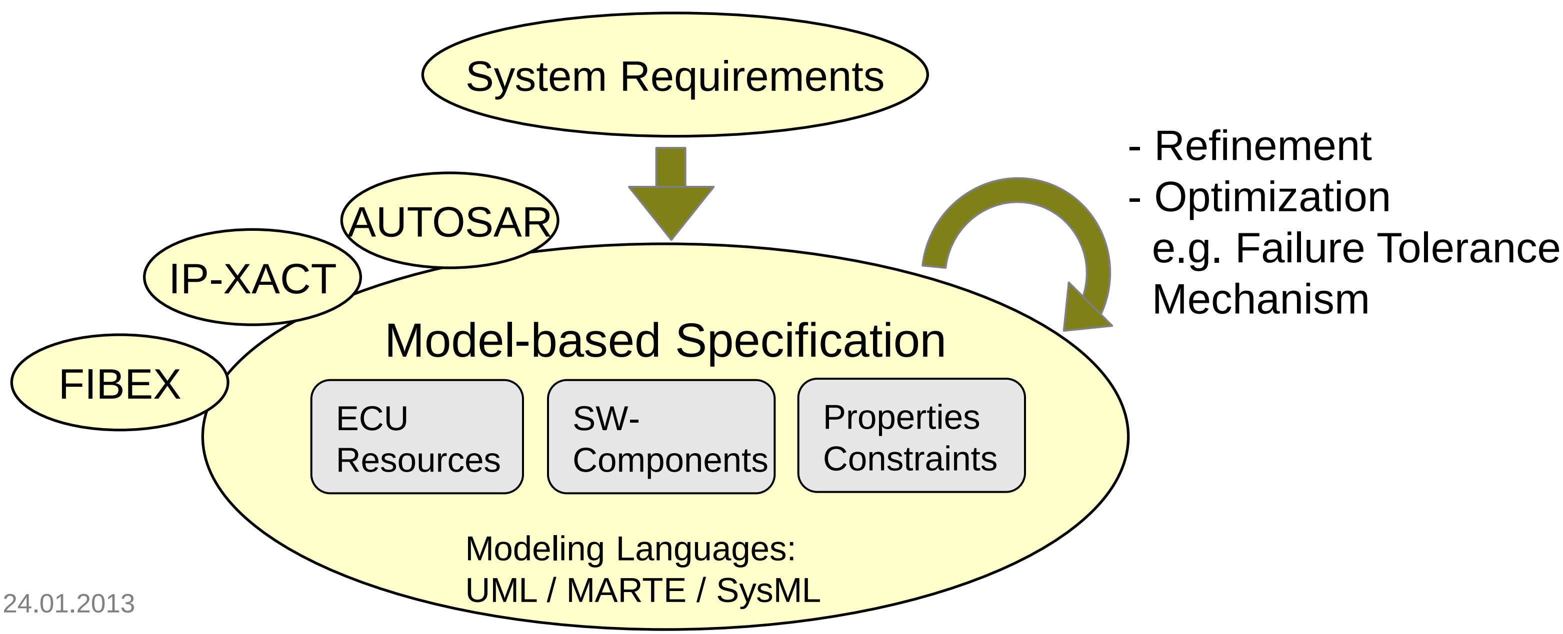




Integration of the Analysis

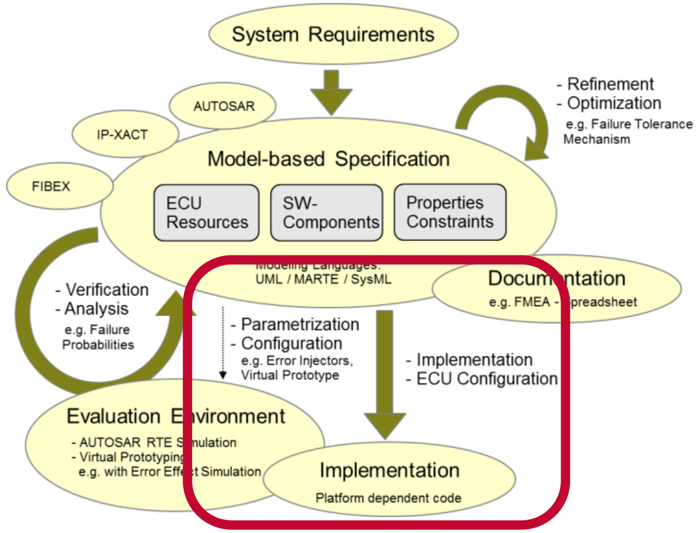
- Information base creation
  - Modeling tool chain
  - System configuration
  - Reliability information
  - Support system refinements and optimizations

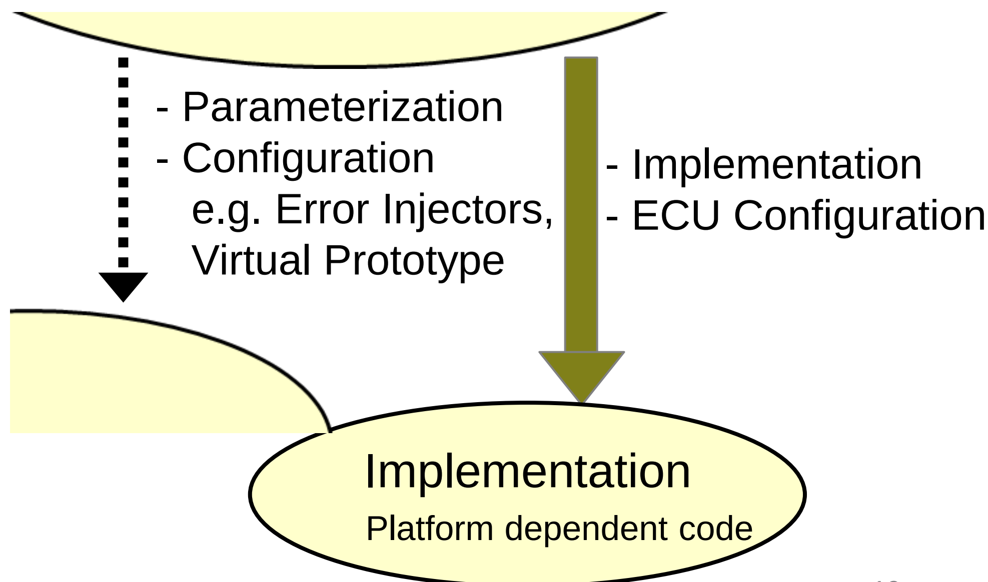




Integration of the Analysis

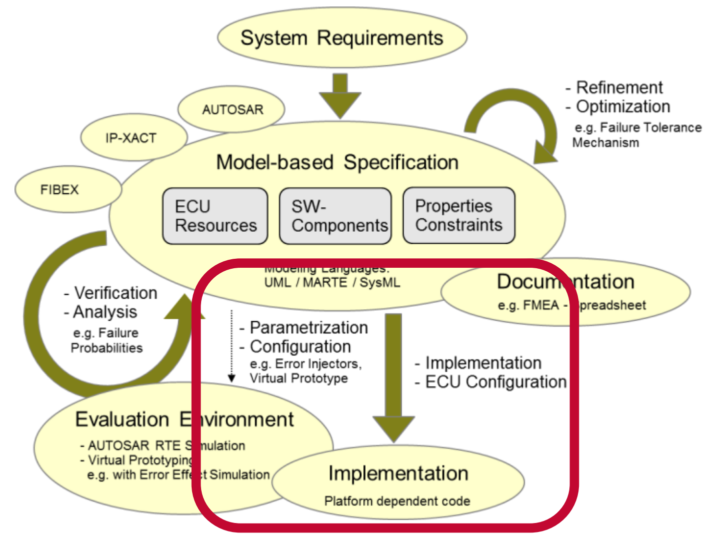
- Information base creation
- Information extraction
  - .xml configuration files
  - Model-to-model transformation
    - Transformation between
      - Editor meta models
      - XML - schemas
    - Query/Views/ Transformation (QVT) language

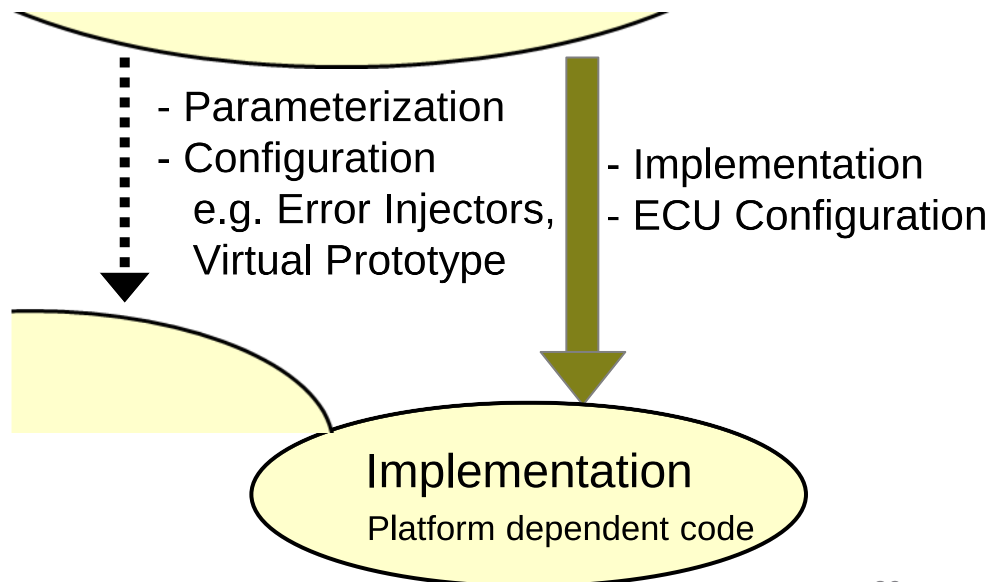




Integration of the Analysis

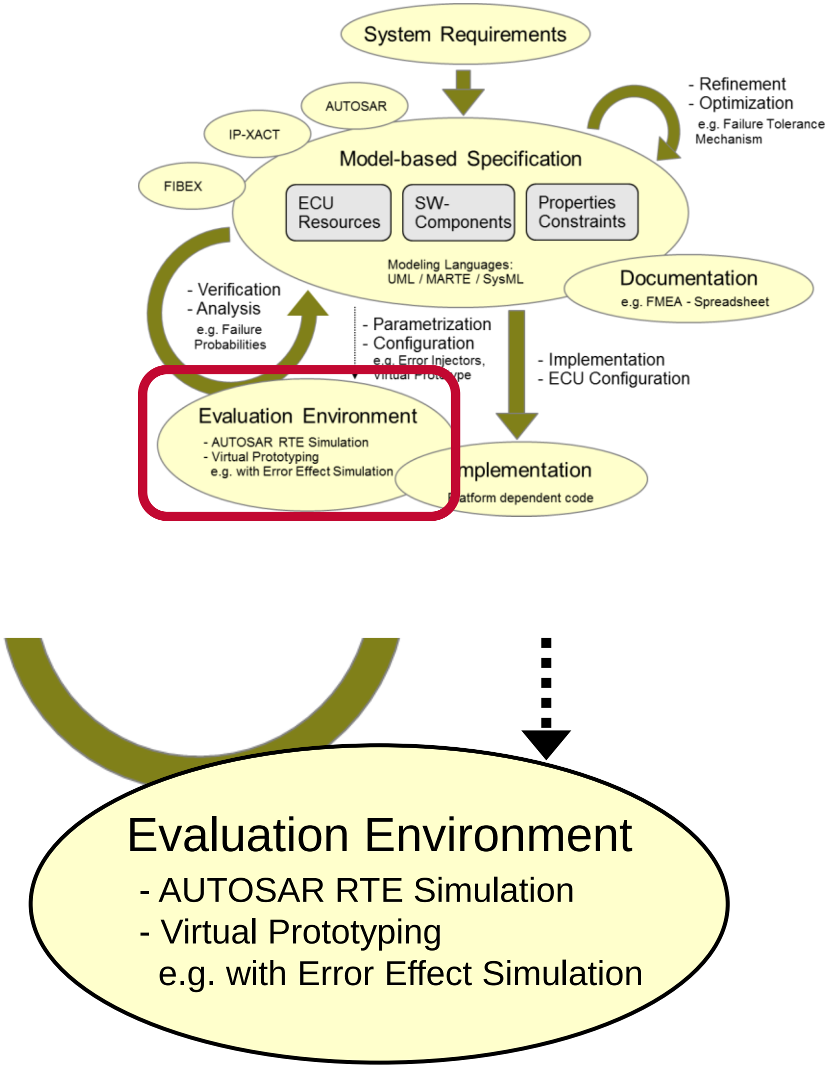
- Information base creation
- Information extraction
- Analysis configuration
  - Analysis library created
    - A configured error injector for each error state
    - A watch point instance for each failure state
  - Virtual prototype configuration
    - Instances creation
    - Parameterization





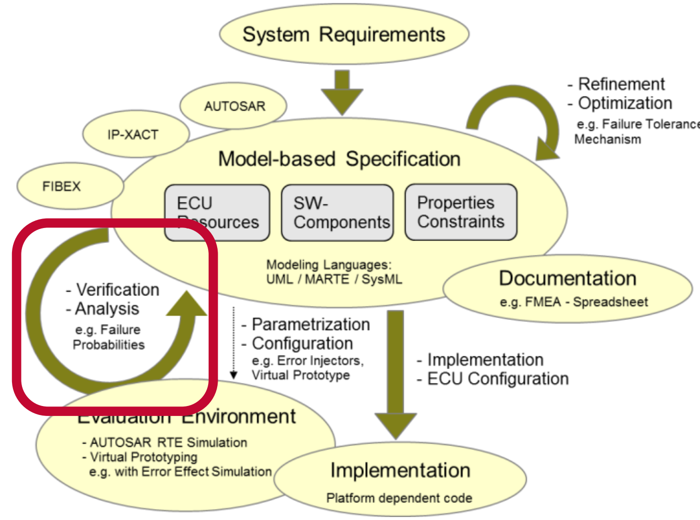
Integration of the Analysis

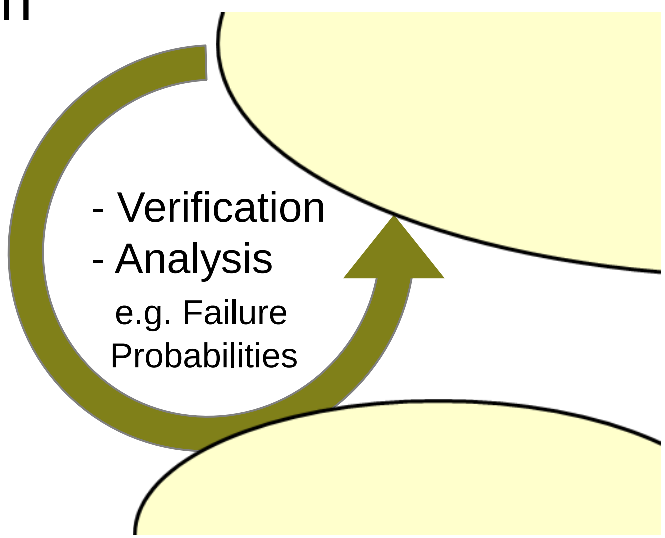
- Information base creation
- Information extraction
- Analysis configuration
- Analysis execution
  - For each error injector multiple simulation are executed
    - Mean failure probability
  - Associated watch points are monitoring



Integration of the Analysis

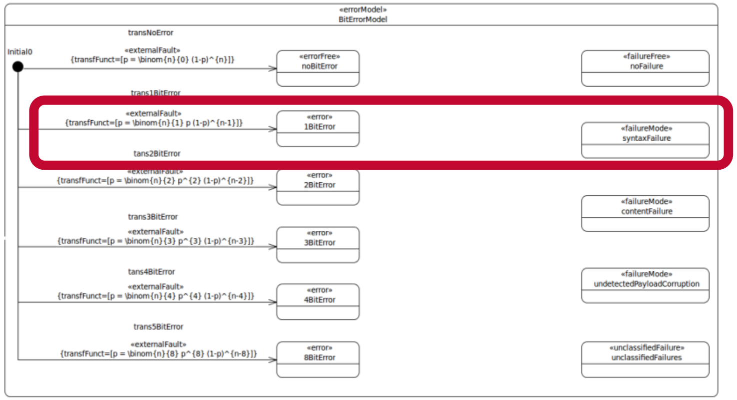
- Information base creation
- Information extraction
- Analysis configuration
- Analysis execution
- Analysis results are back-annotated
  - Model-to-model transformation
  - Failure probabilities

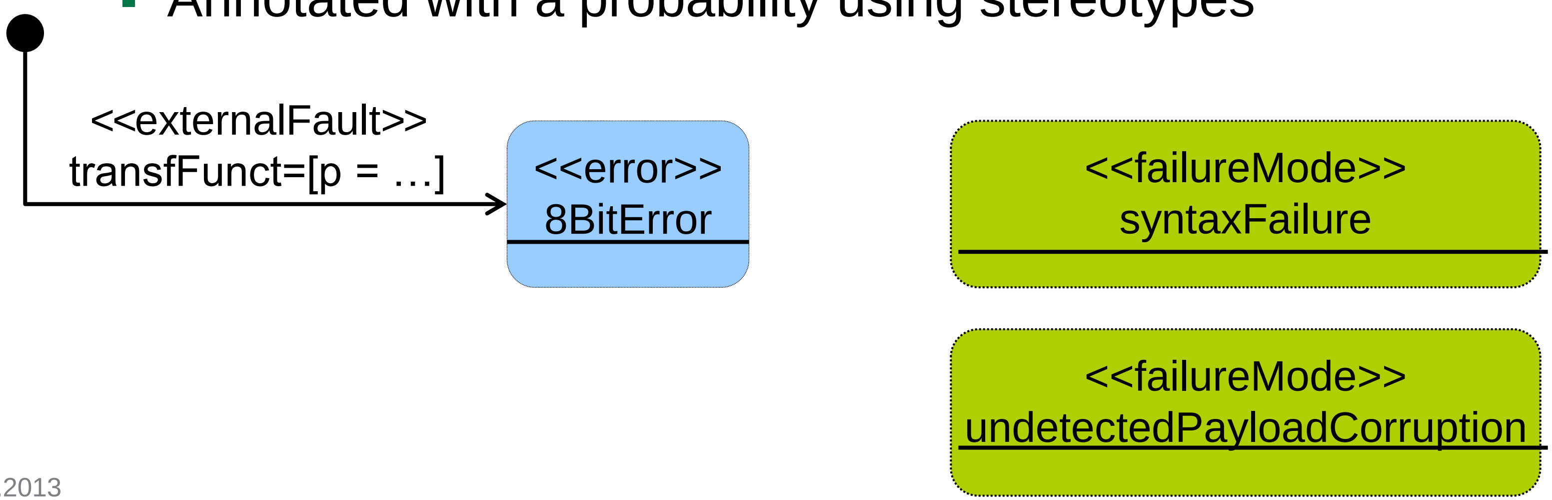




Modeling Framework Extension

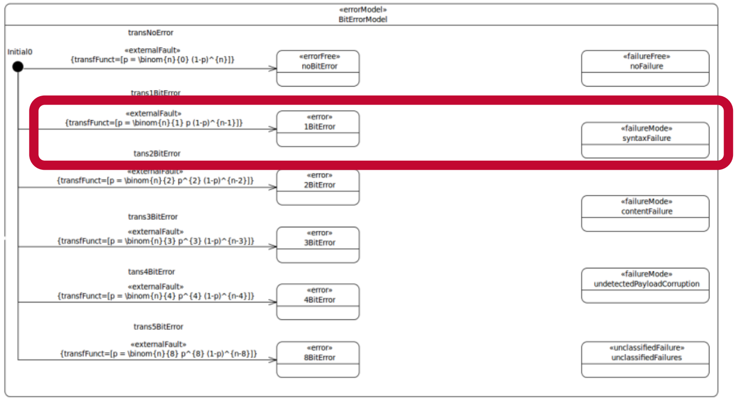
- Fault, Error and Failure specification
  - State diagram with states for
    - Potential errors
    - Resulting failures
  - Transitions between error and failure states
    - Automatically inserted by the analysis
    - Specification of relations between error and failures
    - Annotated with a probability using stereotypes

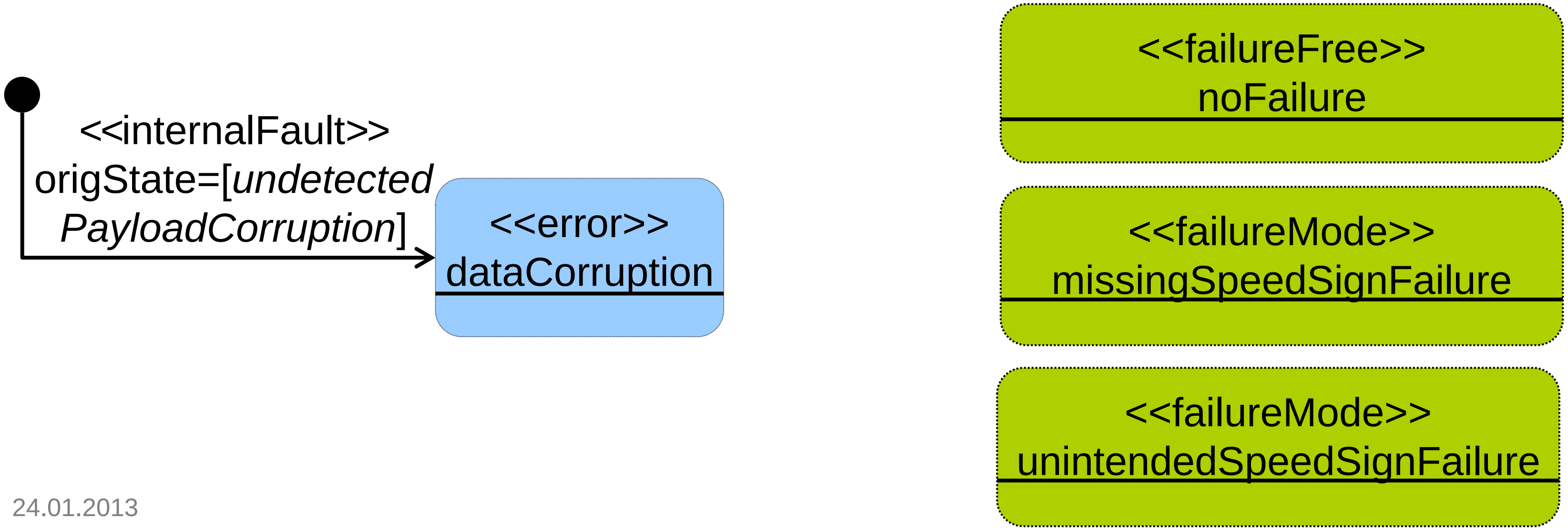




Modeling Framework Extension

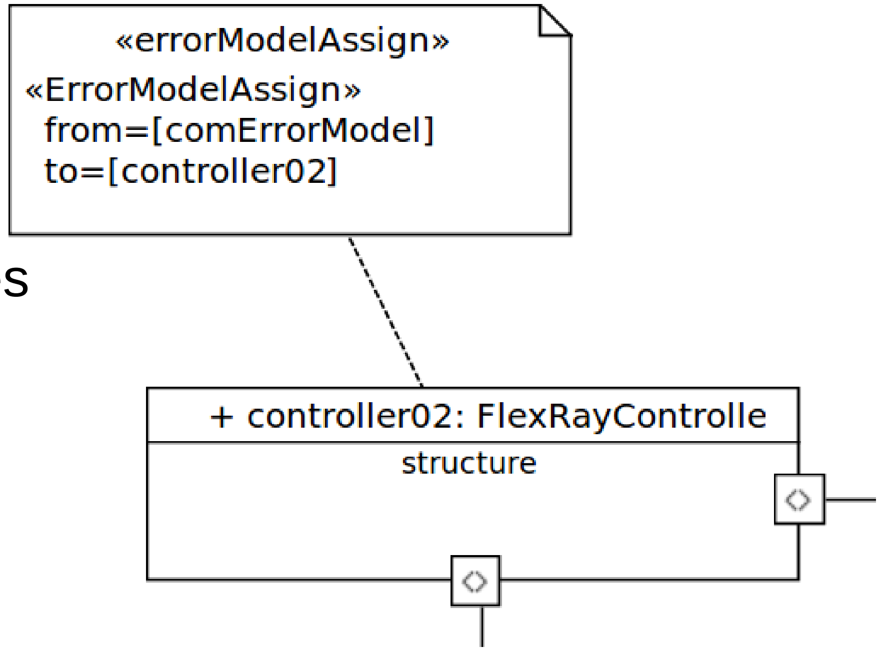
- Fault, Error and Failure specification
  - Grouping of errors and failures
    - Partitioned into state diagrams
    - A service oriented system partitioning
    - Concatenation of state diagrams
      - Specification of causality chains
      - Entry points stereotyped as << internal fault >>





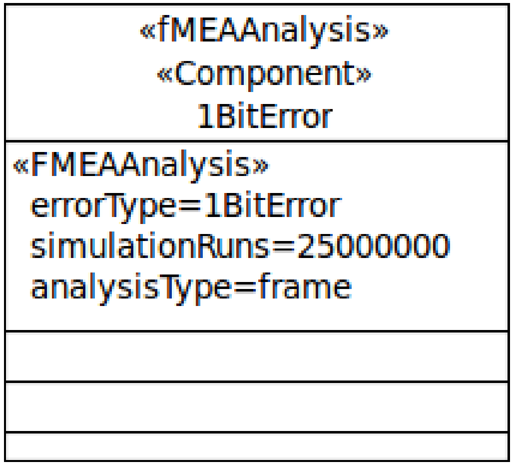
Modeling Framework Extension

- Fault, Error and Failure specification
- Error deployment
  - Assignment of errors to hardware/software instances
  - Composite structure diagram
    - Specifies the system structure
    - Extended with error deployment information
    - Stereotype to associate error models with hardware/software instances



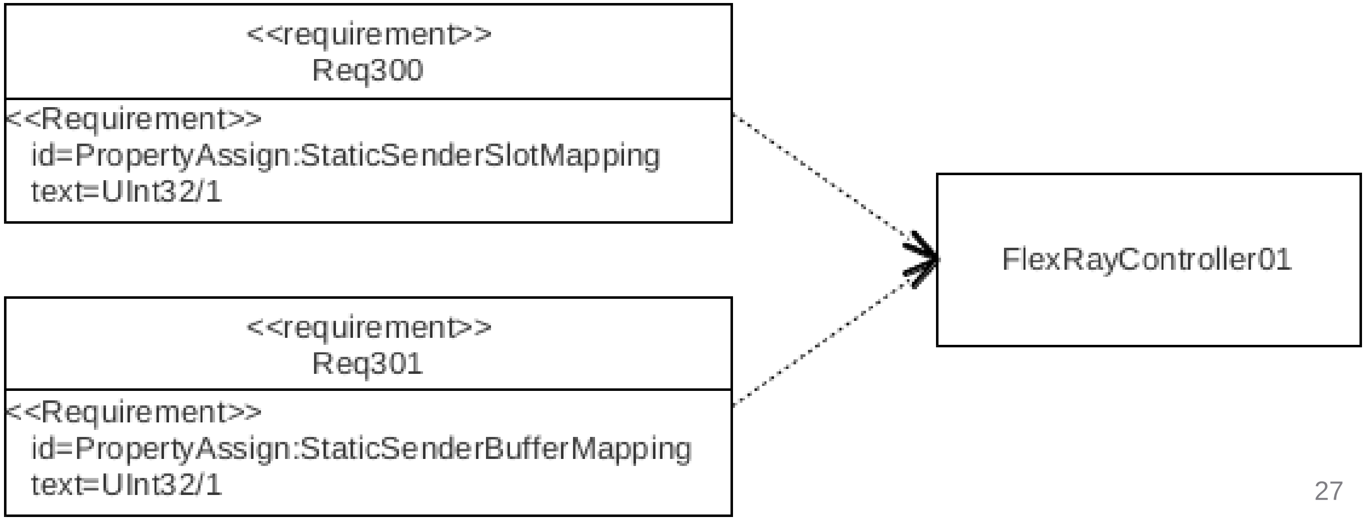
Modeling Framework Extension

- Fault, Error and Failure specification
- Error deployment
- Analysis boundary conditions
  - Class diagram containing analysis contexts
    - Analysis type
    - Simulation amount
  - Each error state is associated with a single analysis context



Modeling Framework Extension

- Fault, Error and Failure specification
- Error deployment
- Analysis boundary conditions
- System specification
  - Automatic configuration of the virtual prototype
    - Structural information
    - Parameterization of the virtual system
      - Requirement annotations for each component instance

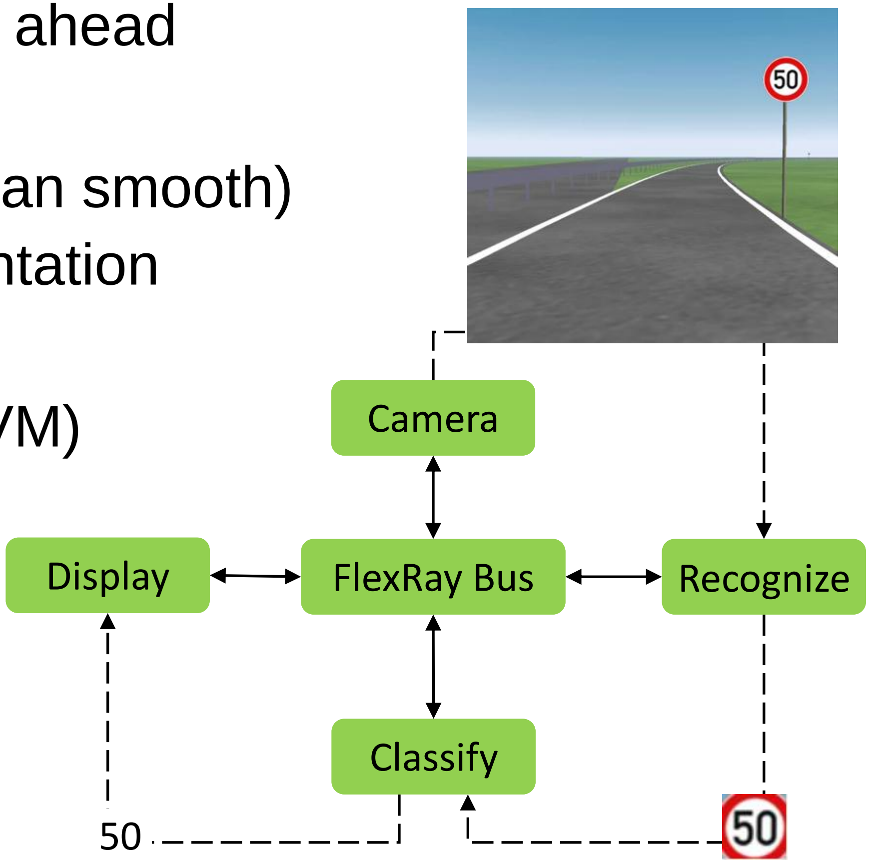


Use Case - Traffic Sign Recognition

- Traffic Sign Recognition Scenario
- Experimental Results
  - Traffic Sign Recognition
  - FlexRay
  - TSR Enhancement

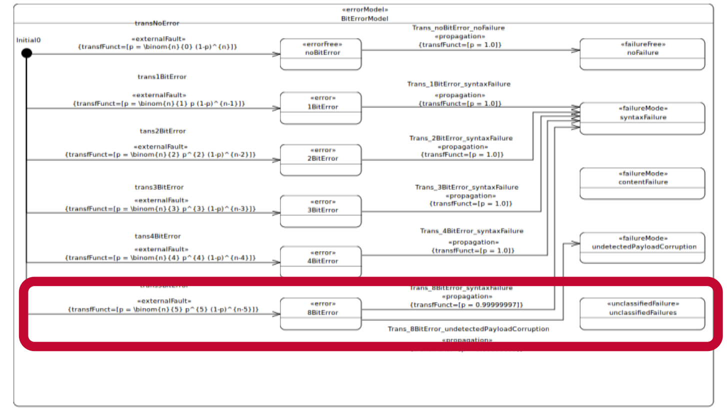
Traffic Sign Recognition (TSR)

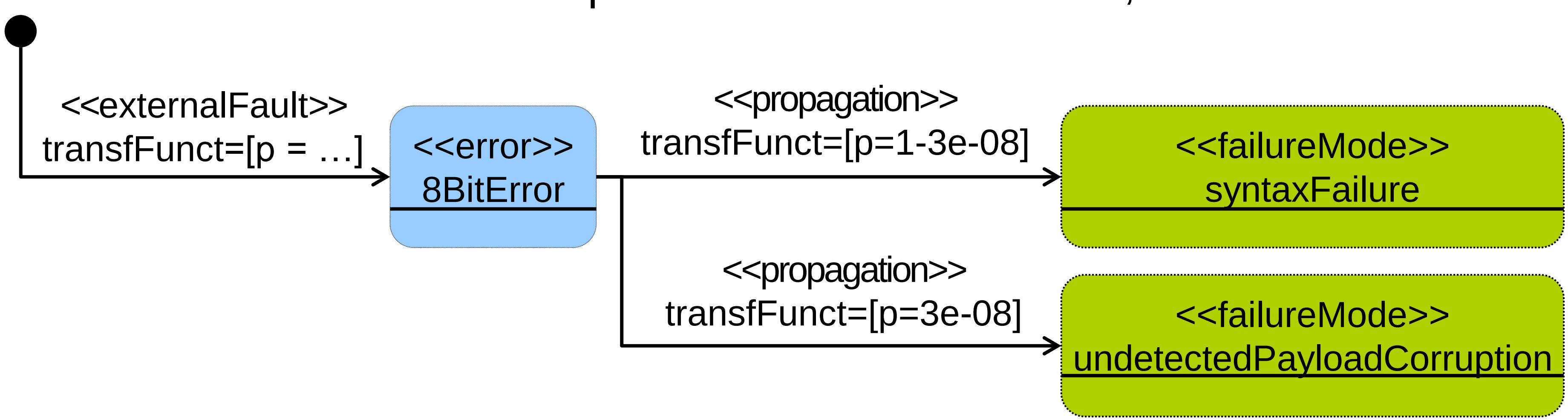
- Camera module
  - Image stream from the road ahead
- Recognize module
  - Pre-processing (e.g. Gaussian smooth)
  - Circle detection and segmentation
- Classify module
  - Support-vector-machine (SVM)
  - Classify speed signs
- Display module
  - Human machine interface
  - Visualize speed limitations
- FlexRay bus [6]
  - Connection of the different TSR modules



Experimental Results: FlexRay

- Bit-Errors within the FlexRay Frame
  - Monitor FlexRay controller service interface
  - Mostly *syntax failure* [6] raised
  - Undetected payload corruption
    - Probability lesser 1E-08
    - Amount of corrupted frames simulated: ~6,7E08



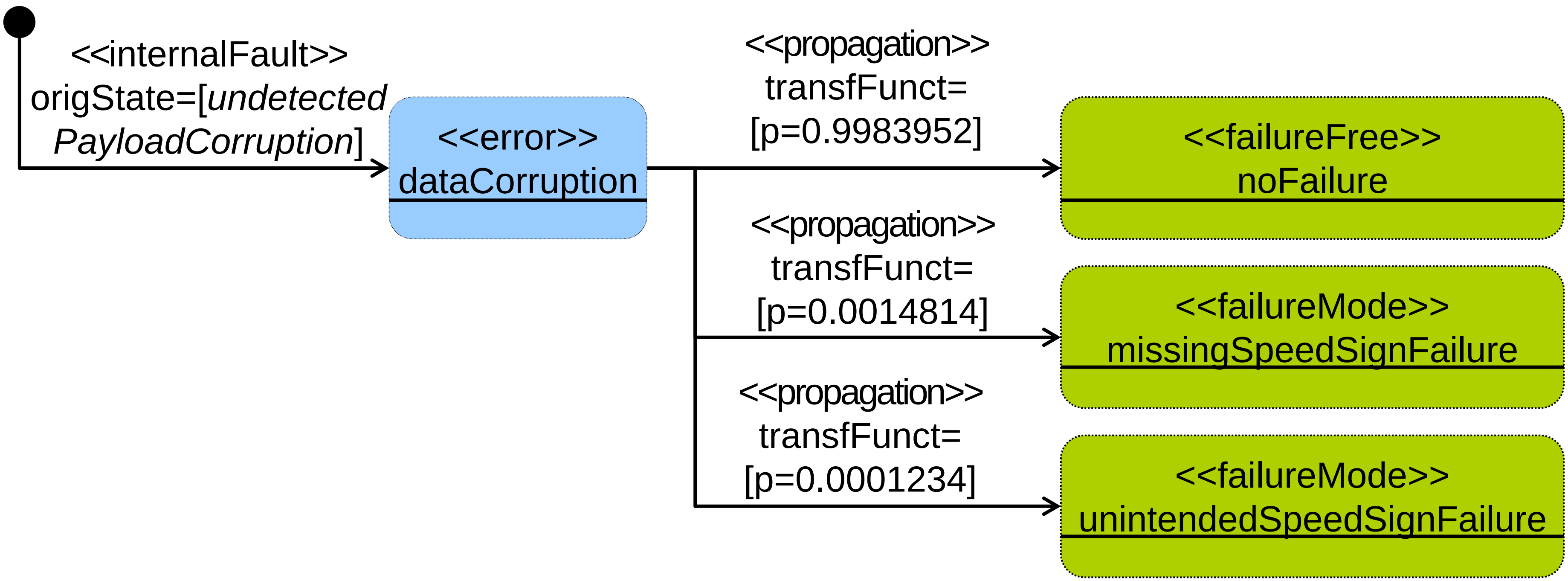


Experimental Results: FlexRay

- Next step: FlexRay controller errors onto the TSR
  - Create causality chain of service failure to error
  - Error modes
    - Complete frame losses
    - Payload corruptions
- Runtime influence
  - Context of the TSR/FlexRay scenario
  - Error injectors and watch points in the most utilized device
  - Increases the runtime by 6-7%

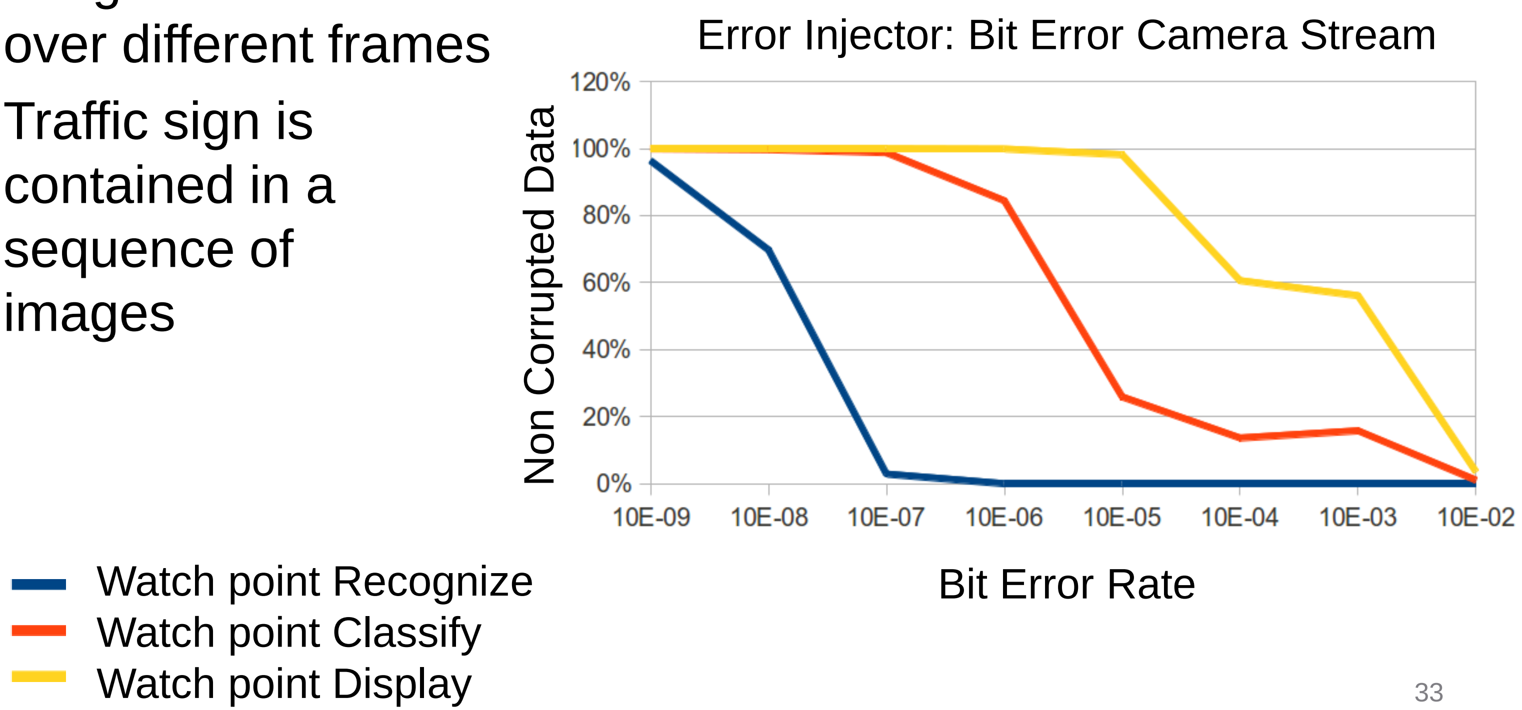
Experimental Results: TSR

- TSR application assessment – raw results
  - Errors injected at FlexRay controller interface
  - Watch points at TSR module results
  - Bit Errors affect transmitted images



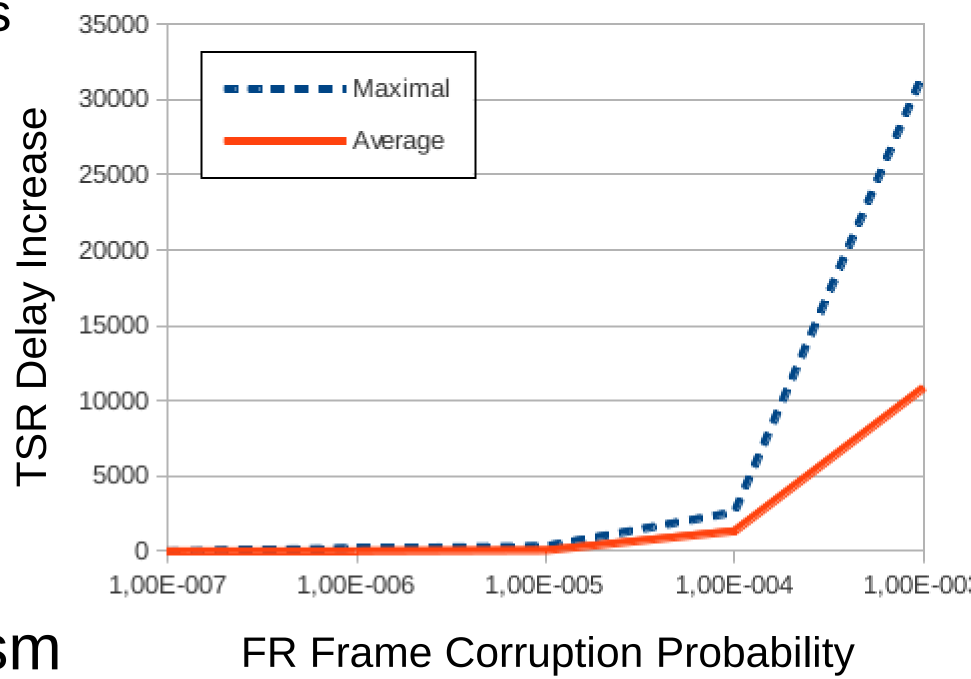
Experimental Results: TSR

- TSR application assessment – processed results
  - All monitored probabilities summarized
  - Application robust under data corruption
    - Only parts of the transmitted image are evaluated
    - SVM tolerates corrupted data
    - Images are distributed over different frames
    - Traffic sign is contained in a sequence of images



Experimental Results: TSR Enhancement

- Increase the reliability of the TSR system
  - Hough-Transformation
    - Detect circles with a wider spectrum of radii
    - Increases the multiplicity of single traffic sign detections
  - Voting algorithm in the classify device
    - React on sign changes more rapidly
  - Communication layer
    - Acknowledgement mechanism
    - Message retries
- Easy reassessment with existent model
- Asses influence of the error tolerance mechanism



Conclusion

- Simulation based assessment of failure rates
  - Reducing subjective estimations
  - Automatic identification of causality chains
- Acceleration of re-design loops
  - Re-execution of already existing models
- Integration in a model-based design flow
  - Re-use of already modeled information
    - Reducing the overhead of the analysis
  - Seamlessly integrated by back-annotation of the results
    - Single source of information to support a FMEA
- Analysis can range from a rough estimation to an in-detail analysis

Thank you for your attention!

Contact person

Sebastian Reiter

FZI Forschungszentrum Informatik
Systementwurf in der Mikroelektronik
Haid-und-Neu-Str. 10-14
76131 Karlsruhe

sreiter@fzi.de
www.fzi.de/ispe

References

(1) IEC. IEC 61508-7: Overview of techniques and measures, 2000

(2) ISO/DIS 26262, June 28, 2009, ISO standard

(3) IEEE 1666-2011 –
IEEE standard for standard SystemC language reference manual

(4) CHESS project page: http://chess-project.ning.com

(5) MARTE specification version 1.0 (formal/2011-06-02);
http://www.omg.org/spec/MARTE/1.1/PDF

(6) FlexRay Consortium. FlexRay Communications System Protocol Specification Version 3.0. Revision A, 2010