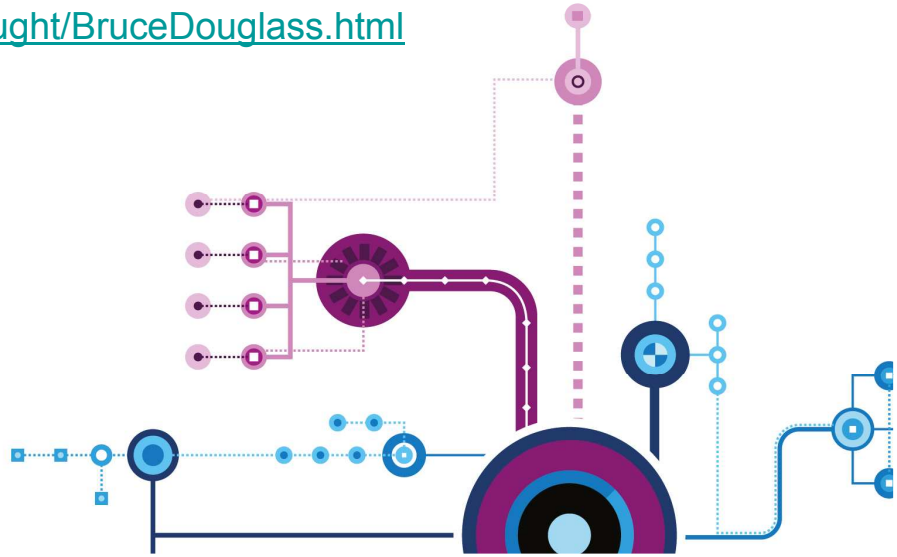
IBM Symposium Systèmes 2014

Concevoir plus rapidement des systèmes de plus en plus flexibles et complexes

Agile Model-Based Systems Engineering (aMBSE)

Bruce Powel Douglass, Ph.D.
Chief Evangelist, Global Technology Ambassador
IBM Rational
Bruce.Douglass@us.ibm.com
Twitter: @BruceDouglass
Yahoo: tech.groups.yahoo.com/group/RT-UML/
IBM: www-01.ibm.com/software/rational/leadership/thought/BruceDouglass.html

Jeudi 27 mars 2014
à l'IBM Client Center Paris



State of the Practice for Systems Development

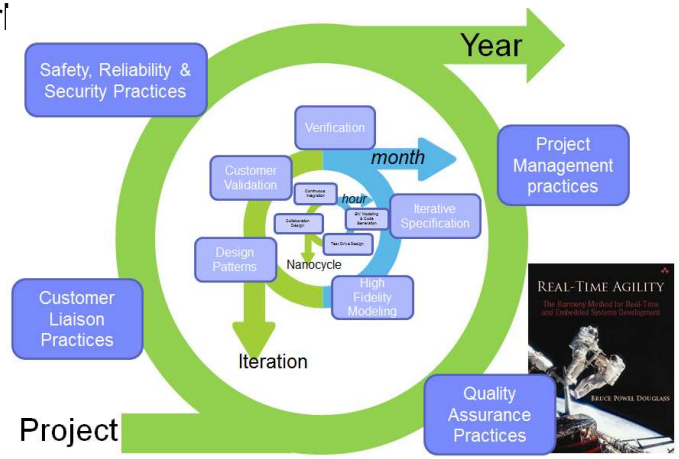
- Systems Engineering Environments in general
  - Are document-centric
  - Require huge investment in planning that doesn't reflect actual project execution
  - Have difficulty adapting to change.
  - Require expensive and error-prone manual review and update processes.
  - Require long integration and validation cycles
  - Are difficult to maintain over the long haul
- Additional standards constraints (eg DO-178C, ARP4761, ISO26262, AUTOSAR, DoDAF) add to the challenge
  - Tooling Selection
  - Dependability engineering
    - Safety
    - Reliability
    - Security
  - System certification

What do we mean by “verification”?

- **Syntactic verification** – “well-formedness” (*compliance in form*)
  - Performed by quality assurance personnel
  - Two types
    - Audits – work tasks are performed as per plan and guidelines
    - Syntactic review – work products conform to standard for organization, structure and format
  - Ex:
    - Requirements shall be uniquely numbered, be organized by use case, use the word “shall” to indicate the normative phrase of a requirement; functional requirements shall be modified by at least one quality of service requirement, …
- **Semantic verification** – “correct” (*compliance in meaning*)
  - Performed by engineering personnel
  - Three basic techniques
    - Testing – requires Executability of work products, impossible to fully verify
    - Formal methods – strongest but hard to do and subject to invariant violation
    - Semantic review (subject matter expert & peer) – most common, weakest means

What does “agile” mean for Systems Engineering?

- Do what you need to do, no more and no less
  - This depends heavily on industry, regulation, and business environment
  - Provide the necessary level of rigor, precision, and repeatability
  - Often requires detailed traceability links among work products (e.g. requirements traceability)
  - Use tooling to automate manually-intensive, error-prone wor
- Work iteratively and incrementally
  - Group requirements with user stories or use cases
  - Incrementally add traceability
  - Incrementally develop system architecture
- Verify work products continuously
  - With syntactic verification (Q/A) activities
  - With semantic verification
  - With customer (aka “validation”)

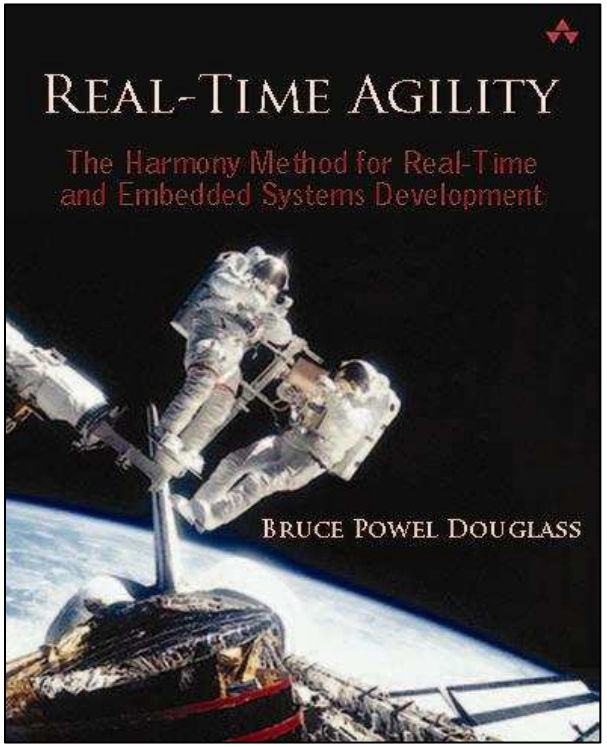


- Outcome contains textual specifications but also linked executable specifications
- Use dynamic planning to adjust project plans based on “ground truth” and responsiveness to change
  - Use goal-based metrics (KPIs) to track project progress
  - Continuously track progress against plan. Adjust planning frequently
- Safety, Reliability, Dependability
  - Not “done once” but continuously assessed

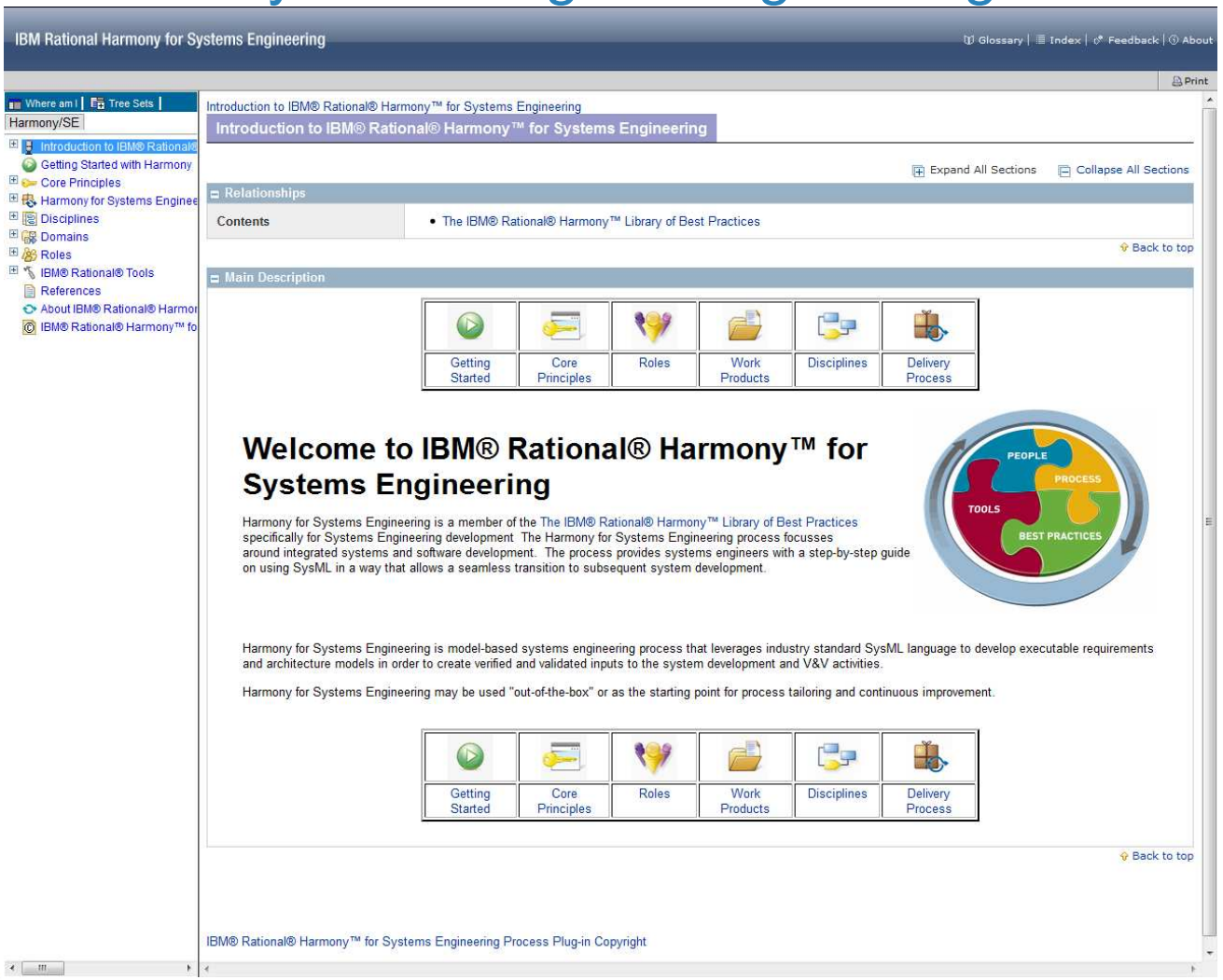
Best Practices for Agile Systems Engineering

- High-fidelity model-based engineering (Hi-MBE)
- Incremental functional analysis with use cases
- Test-driven development of system specifications
  - –Example: Requirements verification via executable requirements modeling with SysML / UML
- Project risk management
- Incrementally add traceability
- Integrated safety and reliability analysis
- Model-based handoff to downstream engineering
- Automated document generation from model artifacts

Note: a key difference between agile SW and agile SE is that the *outcome* of SE is *specifications* and the *outcome* of SW is *implementation*



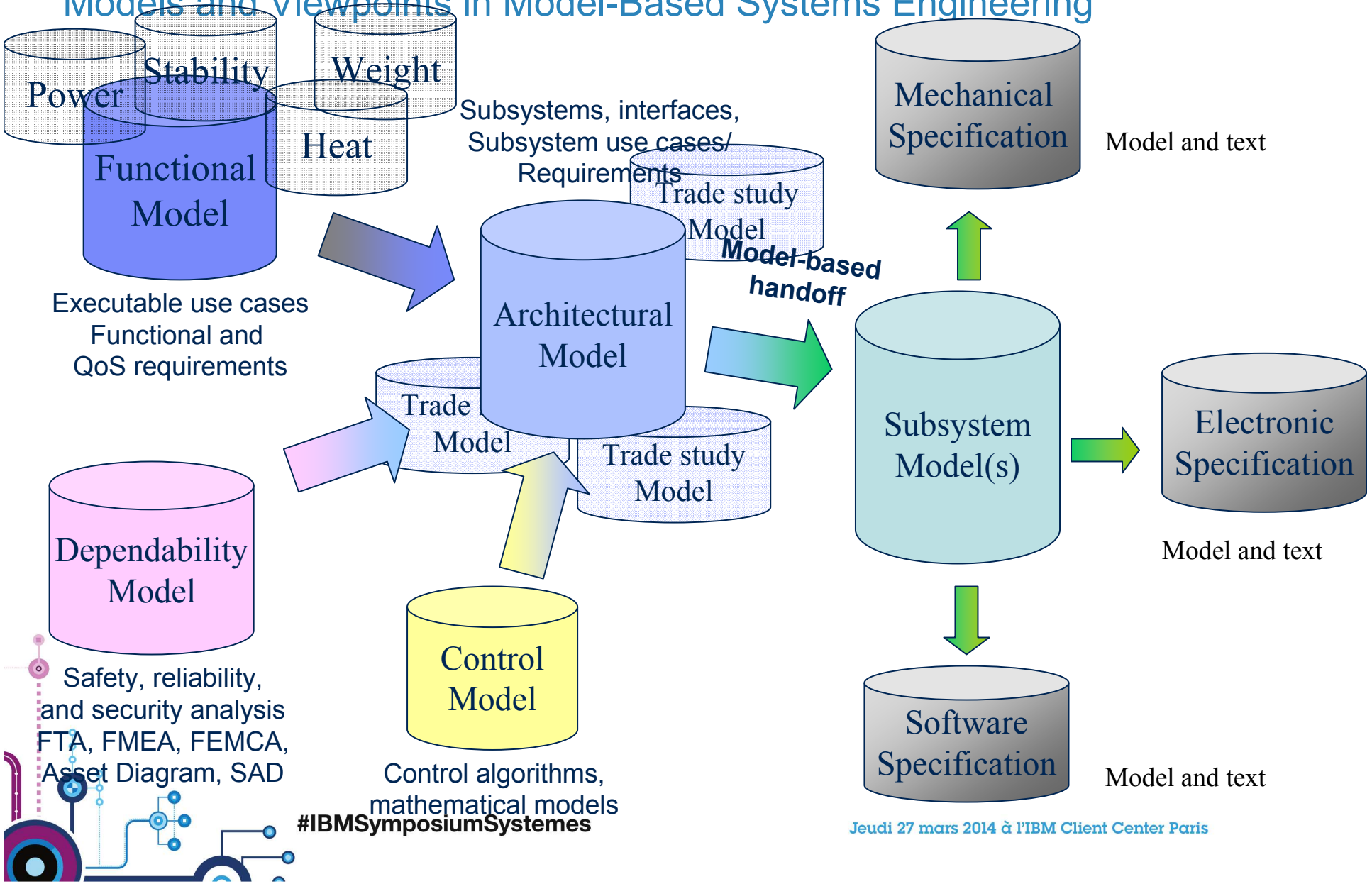
Model-Based Systems Engineering and Agile?



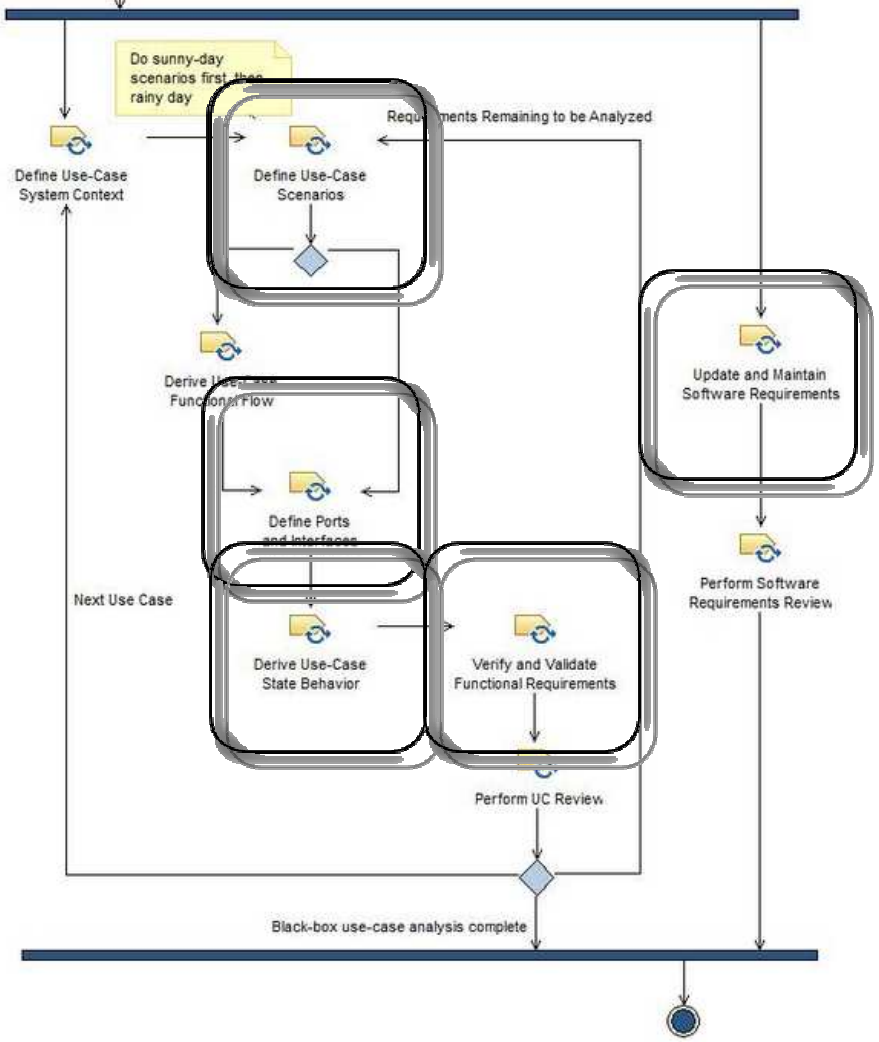
Advantages of MBSE

- Precision
  - Models constructed in formal (or semi-formal) languages are more precise than text
  - Recommendation: Link description informal text to precise, formal models
- Verification
  - Models can be executed, simulated, or (formally) analyzed
    - Requirements models
    - Architecture models
    - Dependability models
    - Control models
- Improved Handoff from systems engineering to downstream engineering
  - Precise models are less likely to be misinterpreted
  - If systems and software engineers use the same modeling languages, then no translation is required
- Improved understanding of architecture
- Improved visualization of functional, structural, and behavioral aspects
- Decreased design learning time

Models and Viewpoints in Model-Based Systems Engineering



Scenario Driven Use Case Construction / Validation



Making it Agile

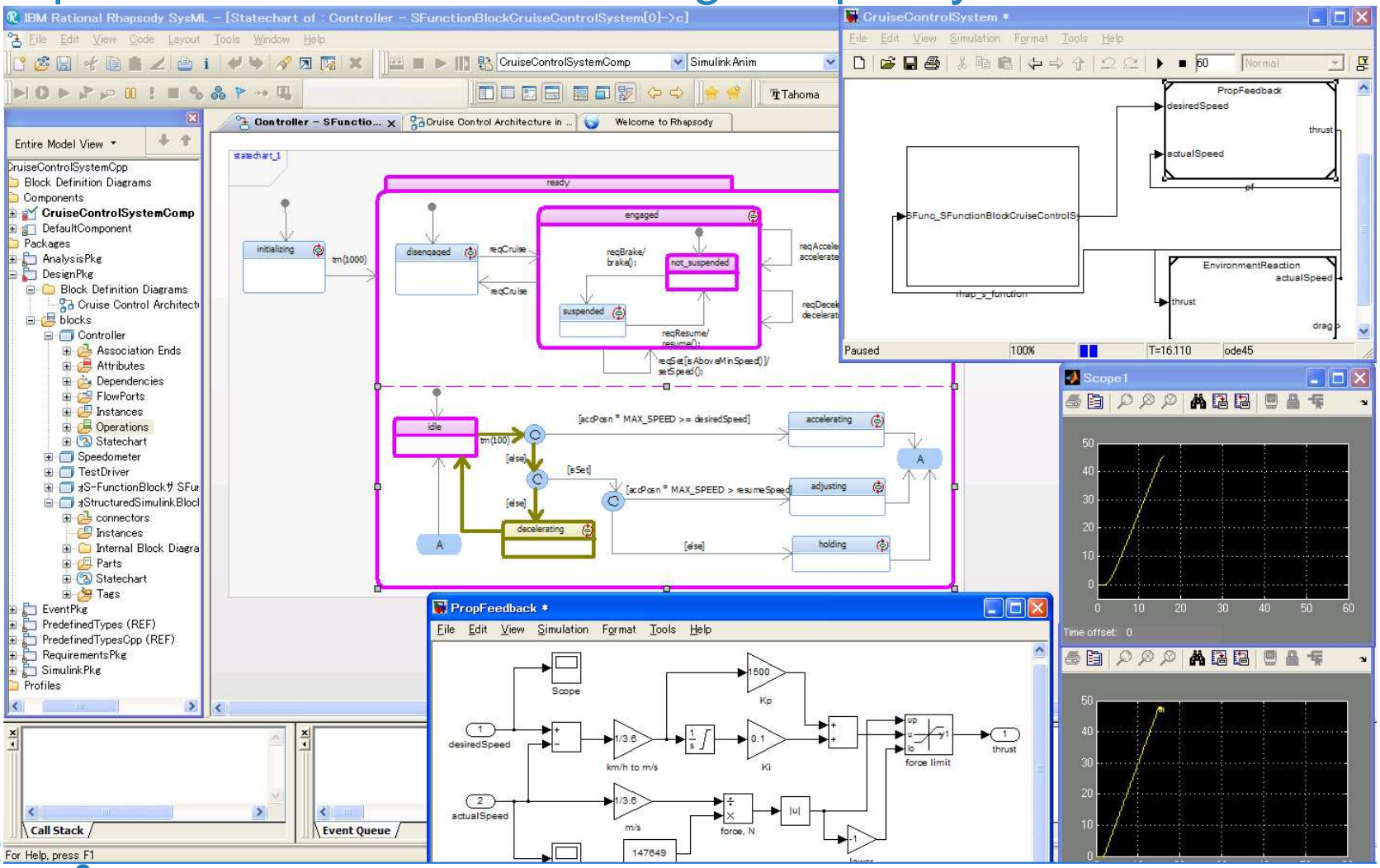
Loop

- **Loop**
  - **Conceptualize requirement aspect**
  - **Incrementally augment model**
  - **Verify**
- **Repeat until all requirements added**

< 1 hr

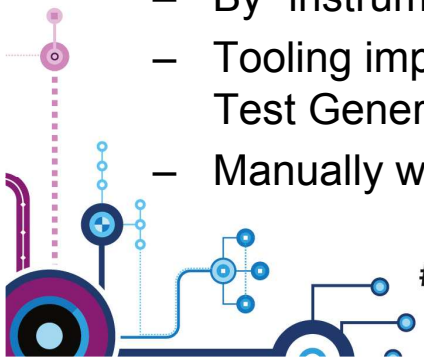
Repeat for all use cases

Requirements Verification Using Rhapsody and Simulink



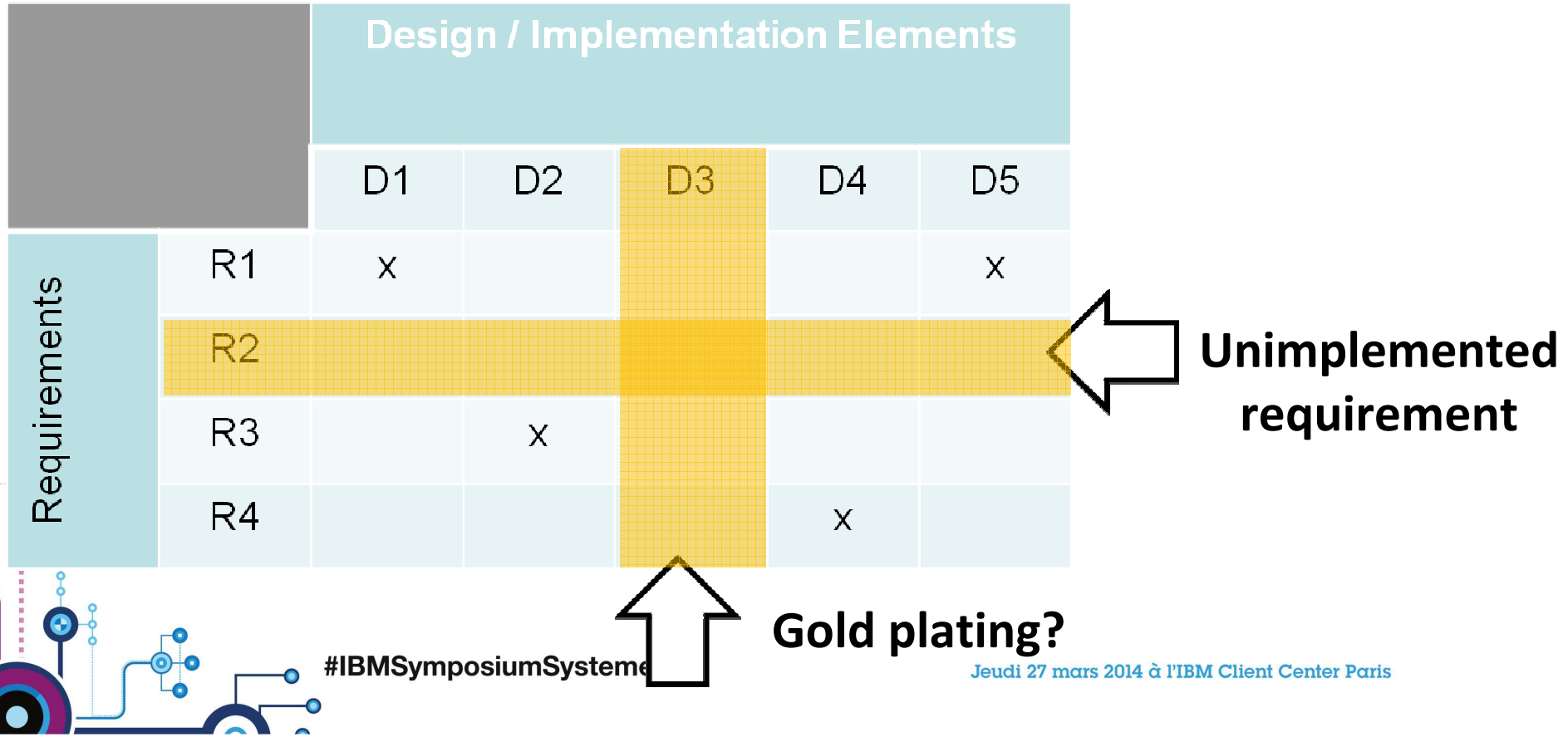
Test-Driven Development isn't just for software anymore

- The principle behind TDD is to develop and apply test cases as you develop a system to demonstrate that it is correct
  - This is done in parallel with the system development and *not* ex post facto
  - This is about *defect avoidance* not so much *defect identification and repair*
- TDD applies to the development of complex system use case models
  - During the nanocycle of a use case's development
    - Make small incremental changes (e.g. add a state, or a couple of actions, or a transition or two)
    - Identify what is the desired behavior of the system that you've specified *so far*
    - Execute that incomplete use case model to ensure that it is correct
    - Repeat until all requirements for the use case and all scenarios defined for the use case have been met in the normative specification
- TDD may be realized in SE Models
  - By "instrumenting the actors" – specifying behavior of the actors to perform tests
  - Tooling implementing the UML Profile for Test (e.g. Test Conductor™ and Automatic Test Generator™)
  - Manually writing test scripts

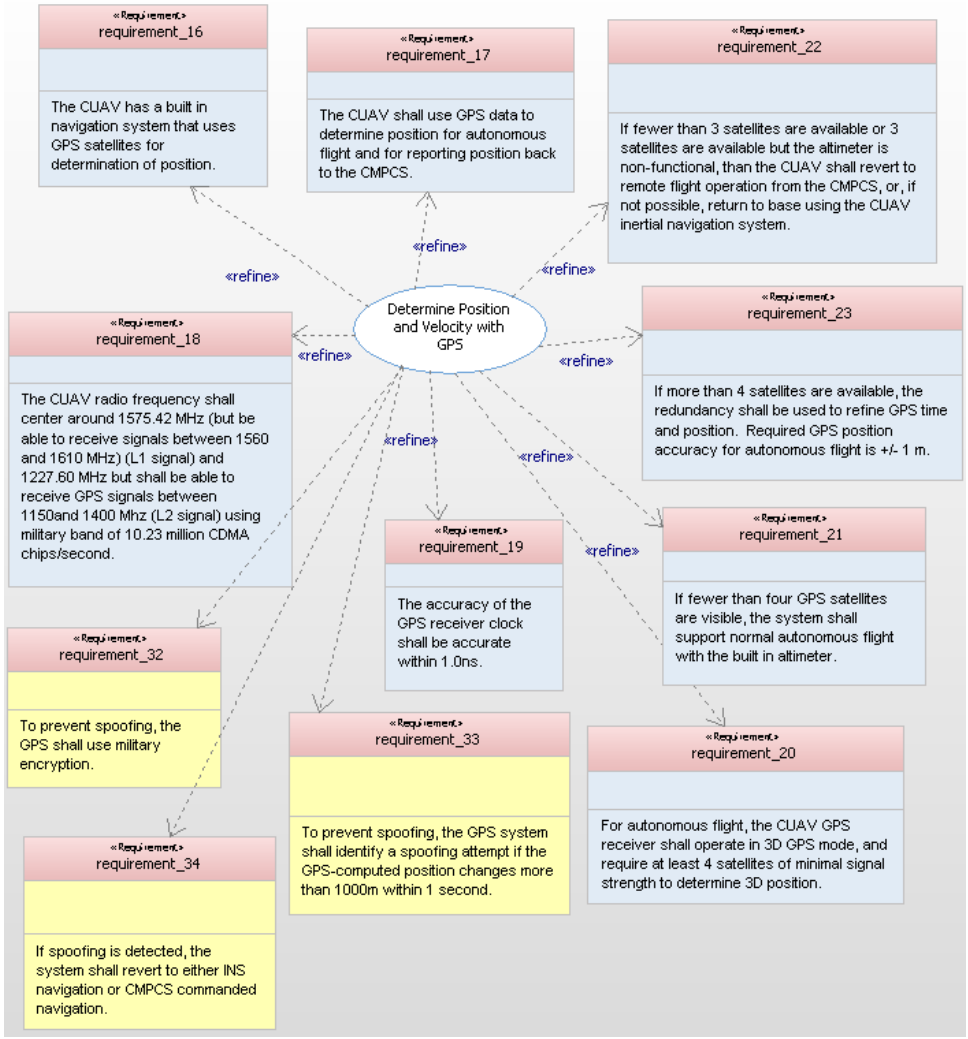


Traceability

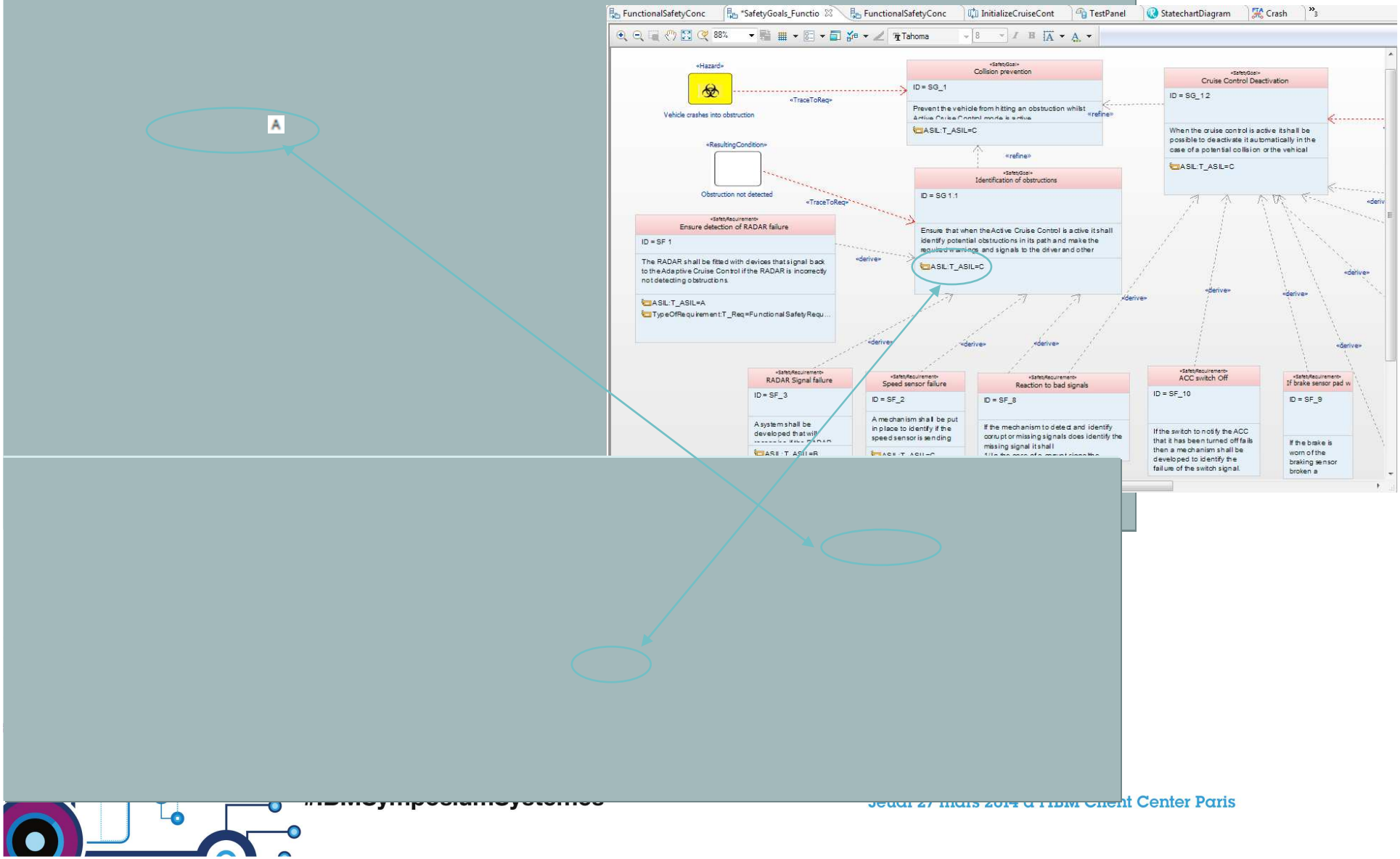
- Traceability serves a number of purposes
  - It allows *impact analysis* – what is the impact if I change this element?
  - It allows for *coverage analysis* – are all elements realized?
  - It allows for *consistency analysis* – are these different elements in different work products consistent and compatible with each other?



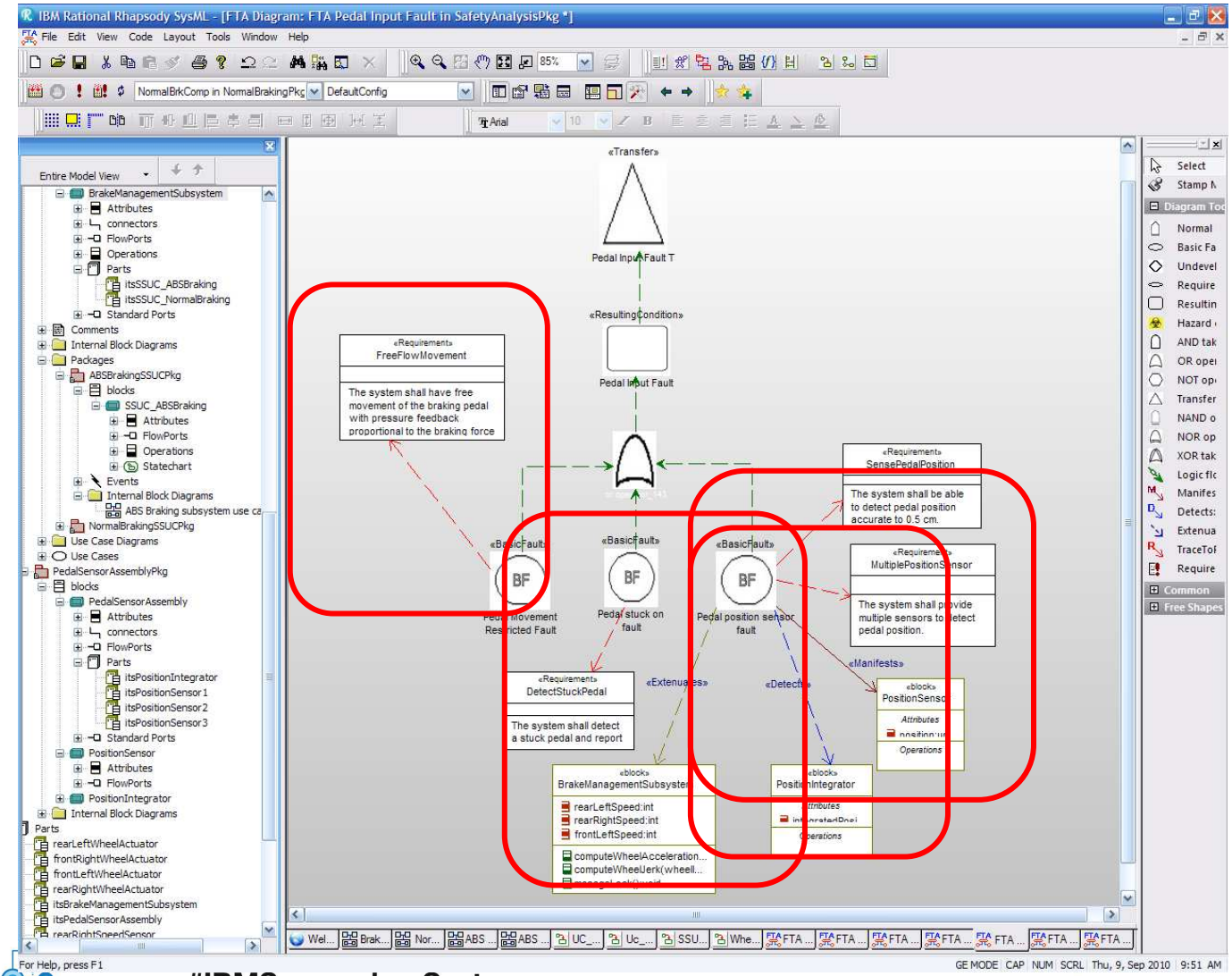
Traceability in Models



Important to Relate Safety Information Through Lifecycle

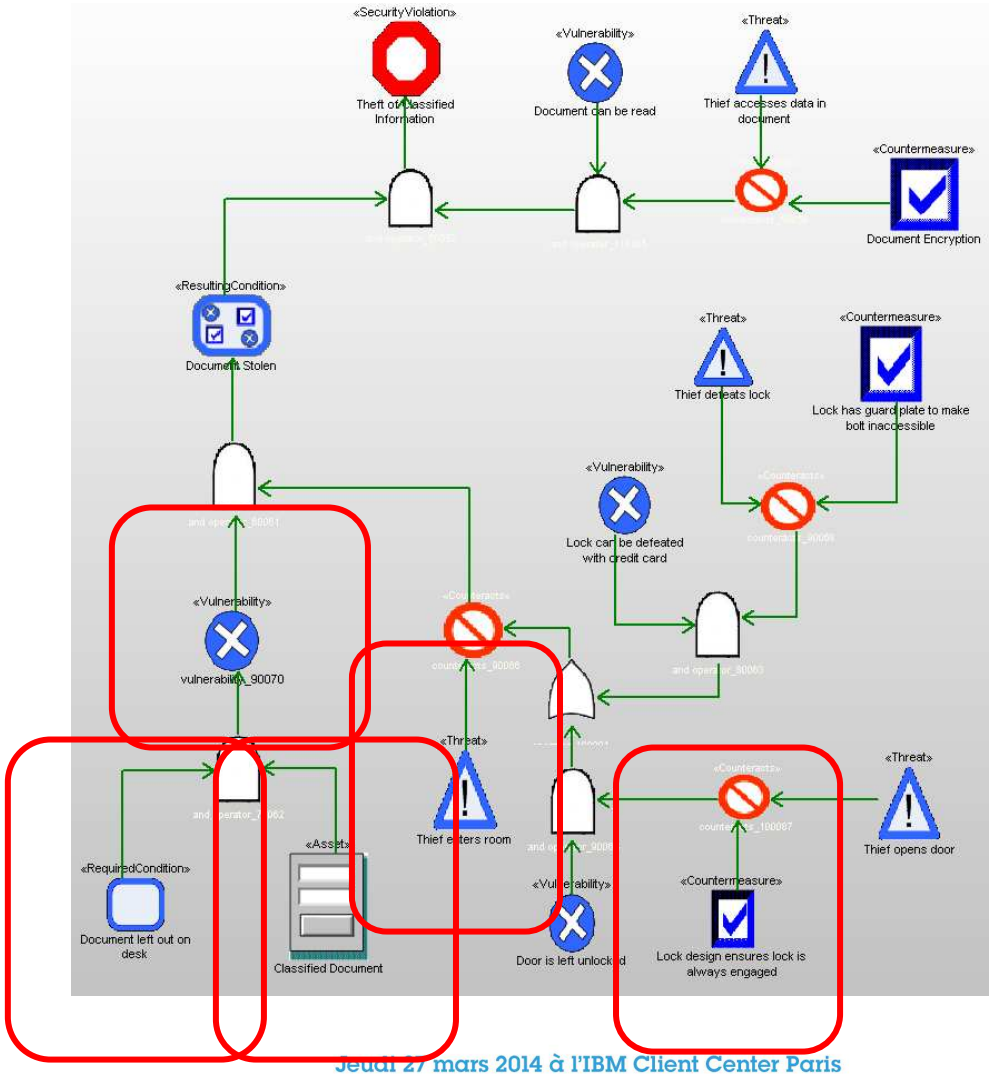


Integrated Dependability Analysis: UML Fault Tree Analysis Profile



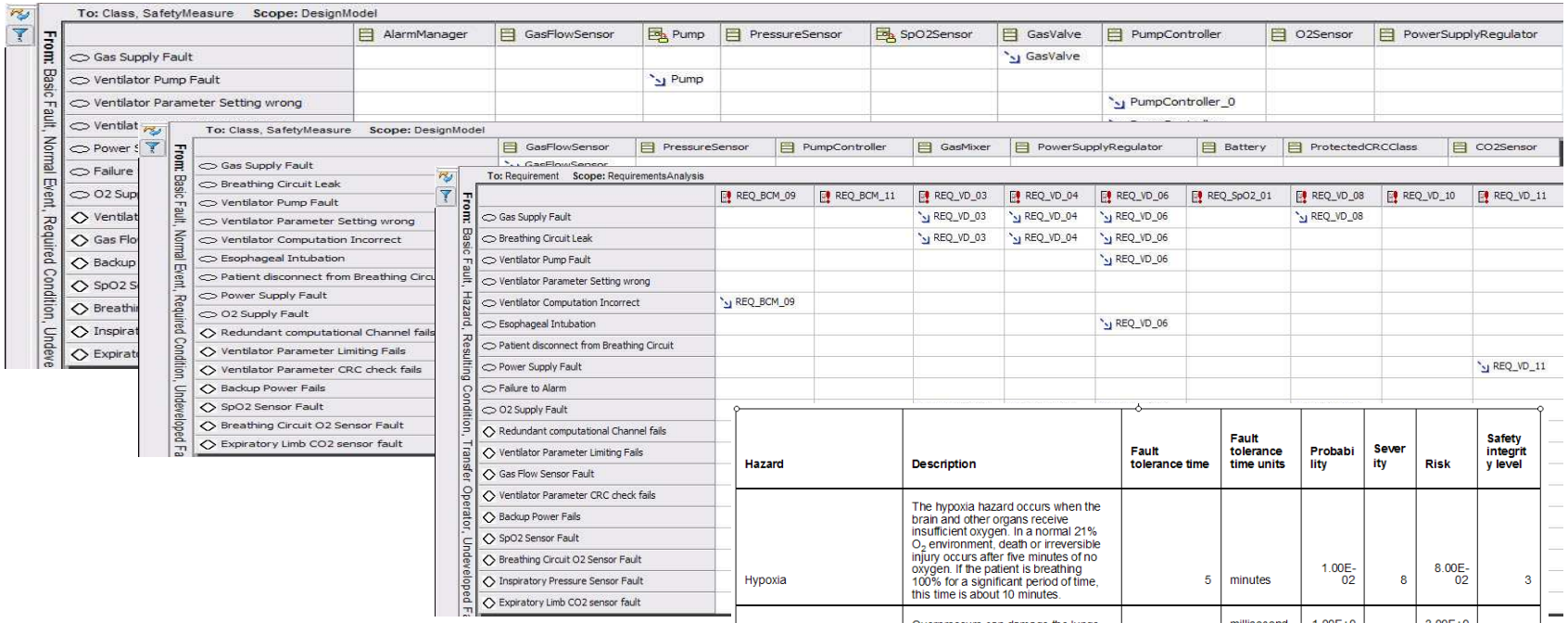
Integrated Dependability Analysis: UML Security Analysis Profile

- Security Analysis Diagram (SAD) is like a Fault Tree Analysis (FTA) but for security, rather than safety
  - It looks for the logical relation between assets, vulnerabilities, attacks, and security violations
  - Permits reasoning about security
    - What kind?
    - How much?
    - Risk assessments



Auto-generation of documents (summary data)

Fault Source Matrix, Fault Detection Matrix, Fault-Requirement Matrix, FMEA, Hazard Analysis…

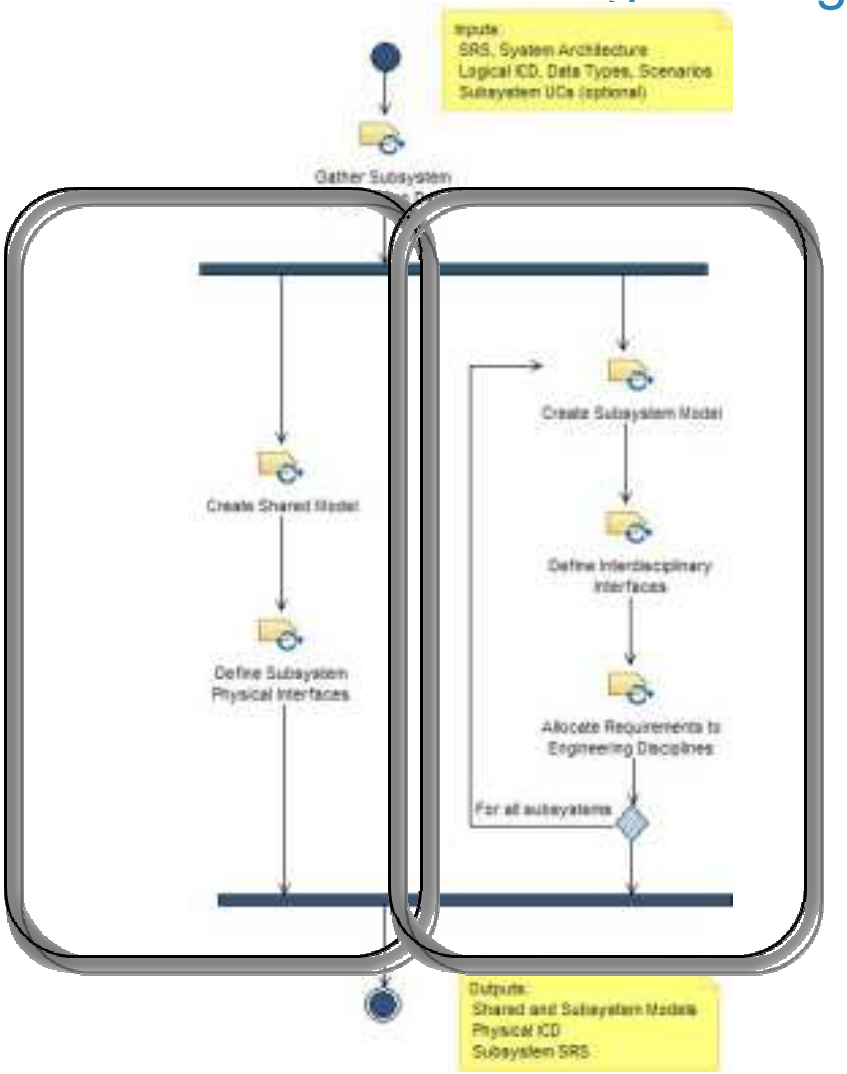


Hazard	Description	Fault tolerance time	Fault tolerance time units	Probability	Severity	Risk	Safety integrity level
Hypoxia	The hypoxia hazard occurs when the brain and other organs receive insufficient oxygen. In a normal 21% O_2 environment, death or irreversible injury occurs after five minutes of no oxygen. If the patient is breathing 100% for a significant period of time, this time is about 10 minutes.	5	minutes	1.00E-02	8	8.00E-02	3
Overpressure	Overpressure can damage the lungs. This is an especially severe trauma, possibly fatal, to neonates.	200	milliseconds	1.00E+04	4	3.00E+04	3
Hyperoxia	Hyperoxia problems are usually limited to neonates, where it can cause blindness.	10	minutes	1.00E+05	4	4.00E+05	4
Inadequate anesthesia	Inadequate anesthesia leads to patient discomfort and memory retention of the surgical procedures. This is normally not life threatening but can be severely discomforting.	5	minutes	1.00E+04	2	2.00E+04	2
Over anesthesia	Over anesthesia can lead to death.	3	minutes	1.00E+03	4	4.00E+03	4
Anesthesia leak into ER	Anesthesia leak can lead to short or, in smaller doses, to long-term poisoning of medical staff.	10	minutes	1.00E+05	5	4.00E+05	5

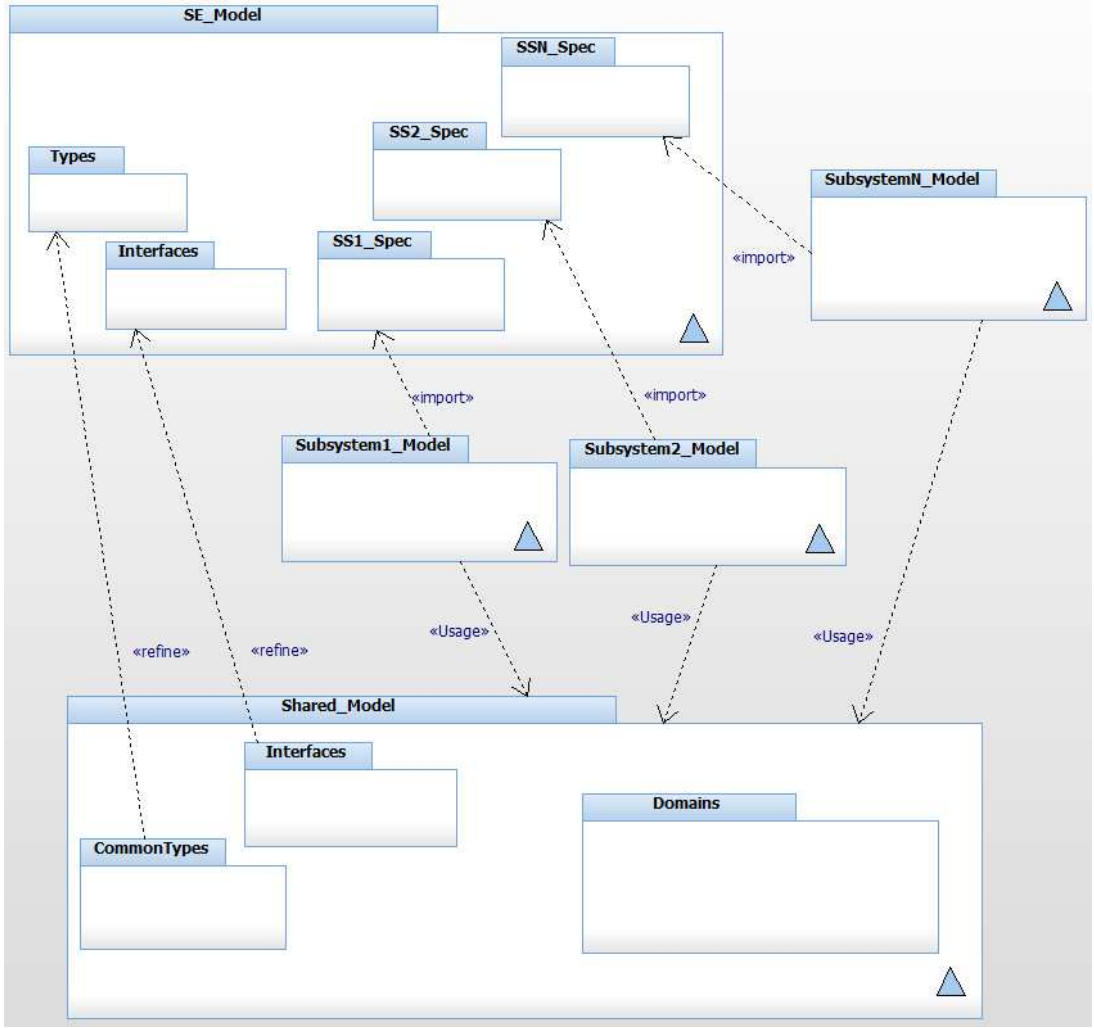
Traceability improves your ability to make your safety/security case

Generate documents are a natural (and automated) outcome of engineering work rather than as a separate activity

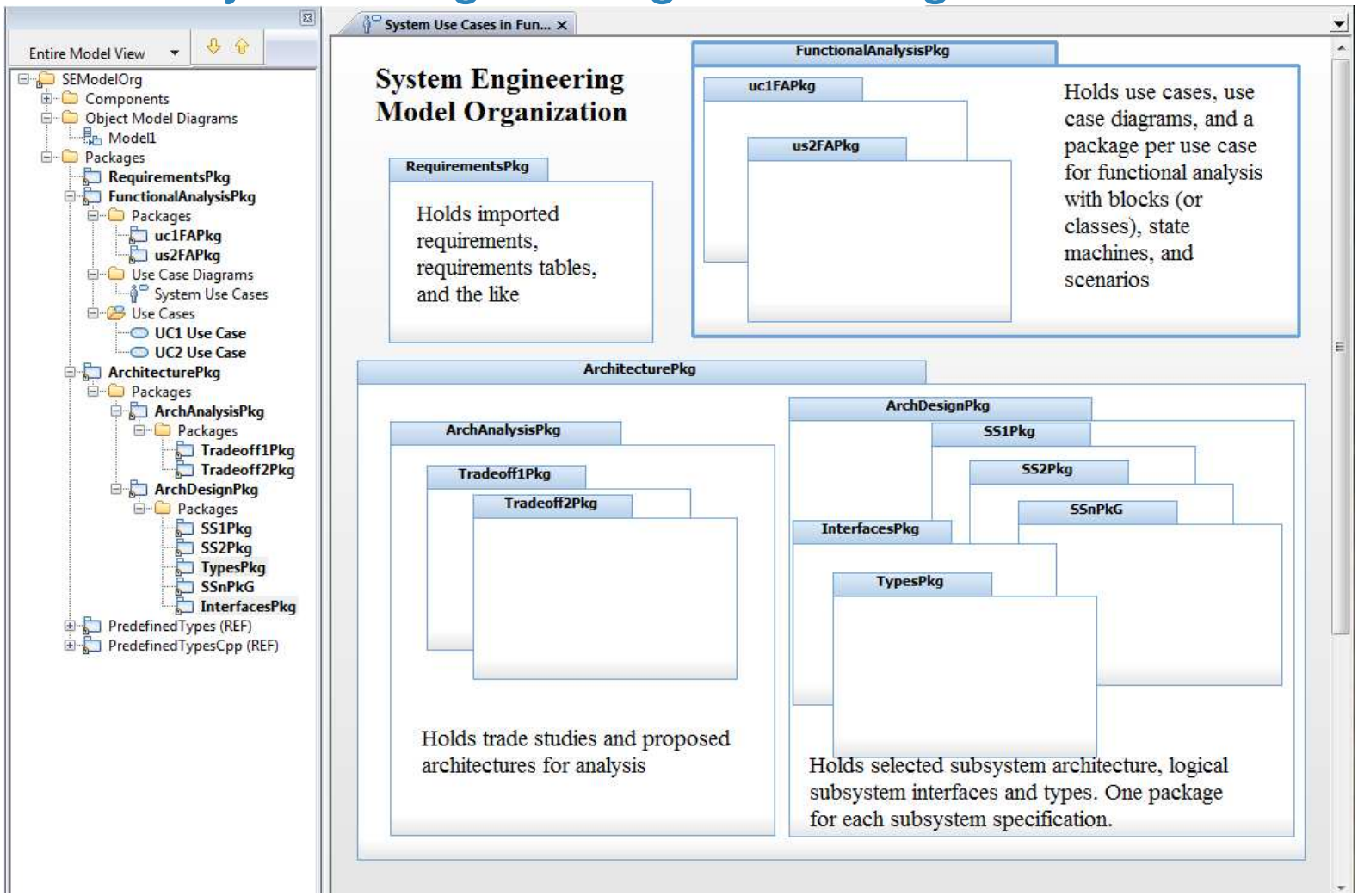
Model-Based Hand-off to Downstream Engineering



Canonical Model Organization



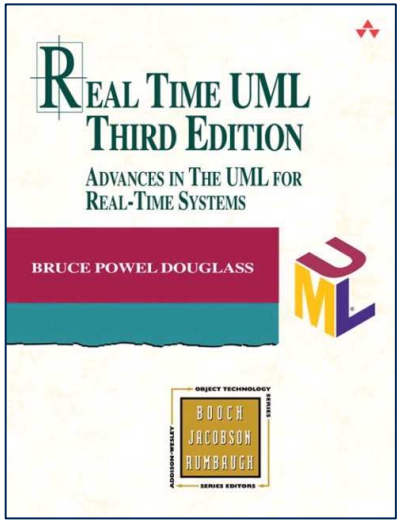
Canonical System Engineering Model Organization

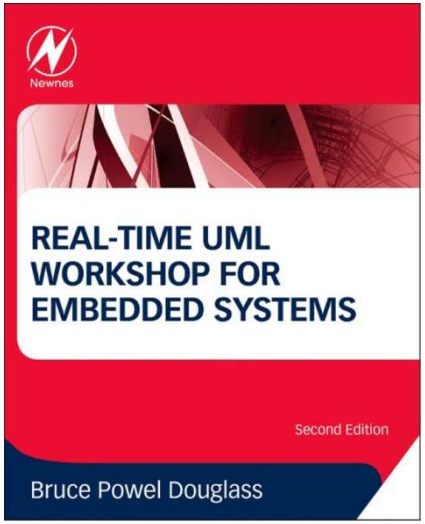


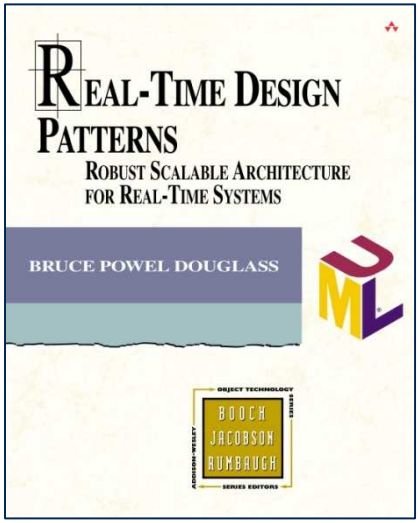
Summary

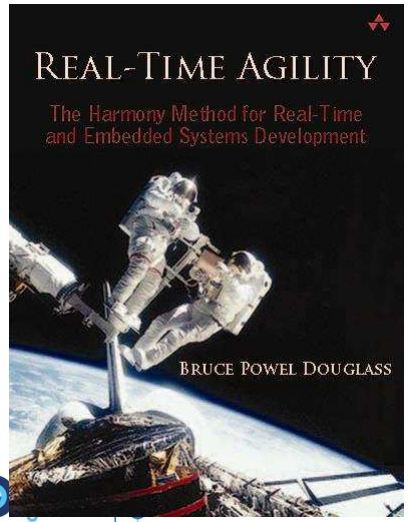
- Systems Engineering capability can be greatly enhanced with two key technologies
  - MBSE - Use of SysML/UML Modeling to capture system
    - Functionality and Qualities of service (executable use cases)
    - Structure (architecture)
    - Model-based hand off to downstream engineering
    - Automatic generation of documentation from model-based work products
  - Agile methods employing
    - Incremental construction and verification of models
    - Test Driven Development nanocycle-level iteration
    - Incorporating dependability analysis with the SE workflow
    - Incremental traceability
- Harmony best practice workflows can be employed in an agile way
  - Process guidance – linked guidance to performance of tasks and creation of work products
  - Project Planning – create project plans with Harmony process templates in Rational Team Concert
  - Project Governance – monitor KPIs in project dashboards

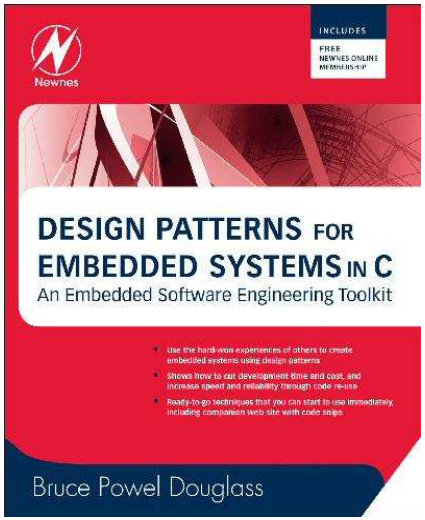
References

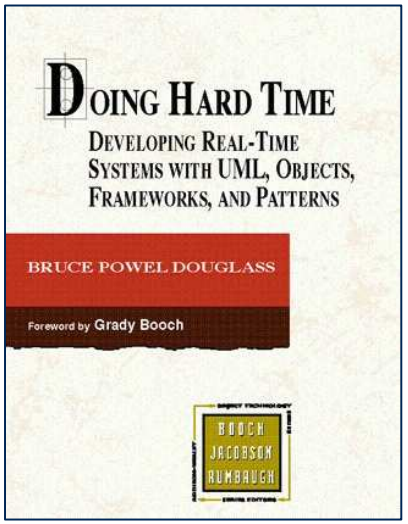

Real Time UML
Third Edition
Advances in the UML for
Real-Time Systems
Bruce Powel Douglass
Booch
Jacobson
Rumbaugh


Newnes
Real-Time UML
Workshop for
Embedded Systems
Second Edition
Bruce Powel Douglass


Real-Time Design
Patterns
Robust Scalable Architecture
for Real-Time Systems
Bruce Powel Douglass
Booch
Jacobson
Rumbaugh


Real-Time Agility
The Harmony Method for Real-Time
and Embedded Systems Development
Bruce Powel Douglass


Newnes
Includes
Design Patterns for
Embedded Systems in C
An Embedded Software Engineering Toolkit
Bruce Powel Douglass


Doing Hard Time
Developing Real-Time
Systems with UML, Objects,
Frameworks, and Patterns
Bruce Powel Douglass
Foreword by Grady Booch
Booch
Jacobson
Rumbaugh