IBM Symposium Systèmes 2014

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

Model Driven Development of Highly Dependable Medical Devices

Bruce Powel Douglass, Ph.D.

Chief Evangelist
Global Technology Ambassador

IBM Rational

Bruce.Douglass@us.ibm.com

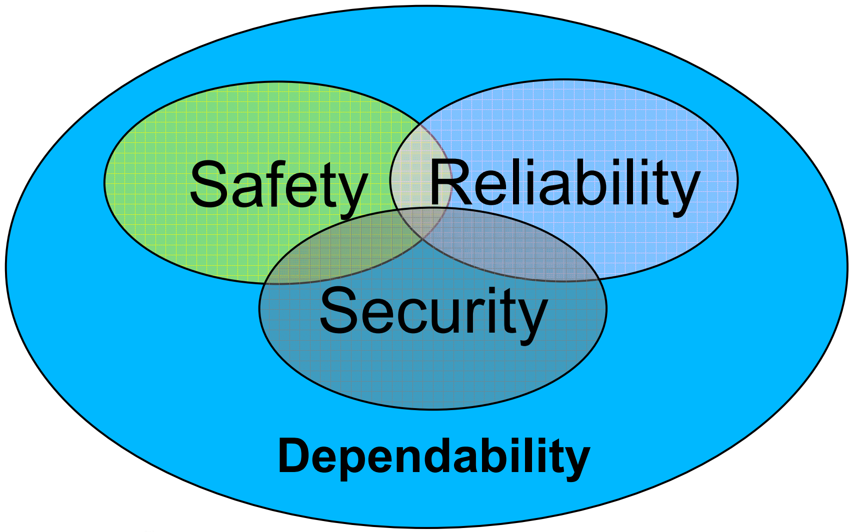
Twitter: @BruceDouglass

http://tech.groups.yahoo.com/group/RT-UML/

Jeudi 27 mars 2014
à l'IBM Client Center Paris

Dependability – I need to get some of that!

- Dependability refers to *the ability for stakeholders to depend upon a device for mission- and life-critical services*
- Three primary aspects
  - Safety – freedom from harm
  - Reliability – availability of services
  - Security – freedom from theft and interference
- Although there is some correlation among these aspects, they sometimes are in conflict



Typical Dependability Data

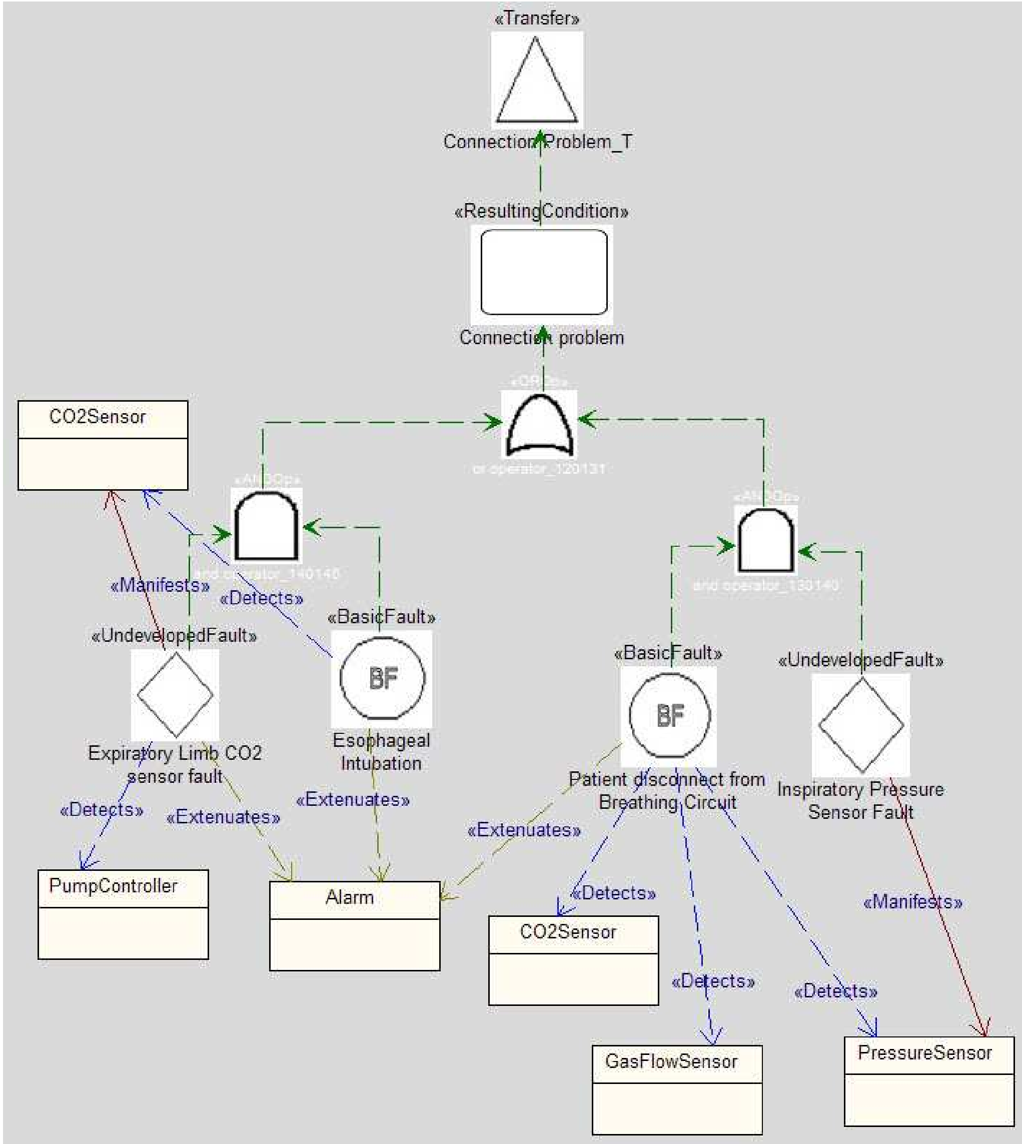
- Hazard Analysis
  - A deductive (top-down) approach that ties together hazards, faults, and safety measures
- Fault Tree Analysis (FTA)
  - A deductive (top-down) approach that links causal factors (e.g. faults, conditions, and events) together via logic operators to create hazards
  - Is recommended in requirements, systems analysis, and design phases
- Fault Means, Effect, and Criticality Analysis (FMECA)
  - An inductive (bottom-up) approach that relates component faults with hazards
  - Cannot be applied until you have a parts list, which is late in the development process
- Security Analysis Diagram
  - UML Profile for modeling cybersecurity, threats, and countermeasures

Hazard Analysis

Hazard Analysis for Target Misidentification											
Hazard	Fault	Severity (1 (low) - 10 (high))	Likelihood (0 - 10)	Computed Risk	Time units	Tolerance Time	Detection Time	Control Measure	Control Action Time	Exposure Time	Is Safe?
Target Misidentification	Noise reduction insufficient	10	8	80	seconds	1.00	0.1	Use multiple algorithms for noise reduction	0.05	0.15	TRUE
	SW error in primary computation	10	4	40	seconds	0.20	0.001	Use 2 computational algorithms	0.05	0.051	TRUE
	SW error in secondary computation	10	4	40	seconds	0.20	0.001	Use 2 computational algorithms	0.05	0.051	TRUE
	Message corruption	10	7	70	seconds	0.20	0.001	CRC on messages	0.001	0.002	TRUE
	Target spec corruption	10	5	50	seconds	0.20	0.05	CRC	0.001	0.051	TRUE
	Target data corruption	10	5	50	seconds	0.20	0.05	Bit inversion redundancy	0.002	0.052	TRUE
	Image data corruption	10	5	50	seconds	0.20	0.05	CRC	0.001	0.051	TRUE

Fault Tree Analysis

- Traceable links among requirements, analysis, design, and safety elements makes your "safety case"
- FTA elements can have traceable, navigable links to
  - Requirements
  - Elements that can manifest faults
  - Elements that can detect faults
  - Elements that handle faults
- FTA elements contain metadata
  - Severity
  - Likelihood
  - MTBF
  - Risk
  - Safety Integrity Level
- UML Fault Tree Analysis Profile

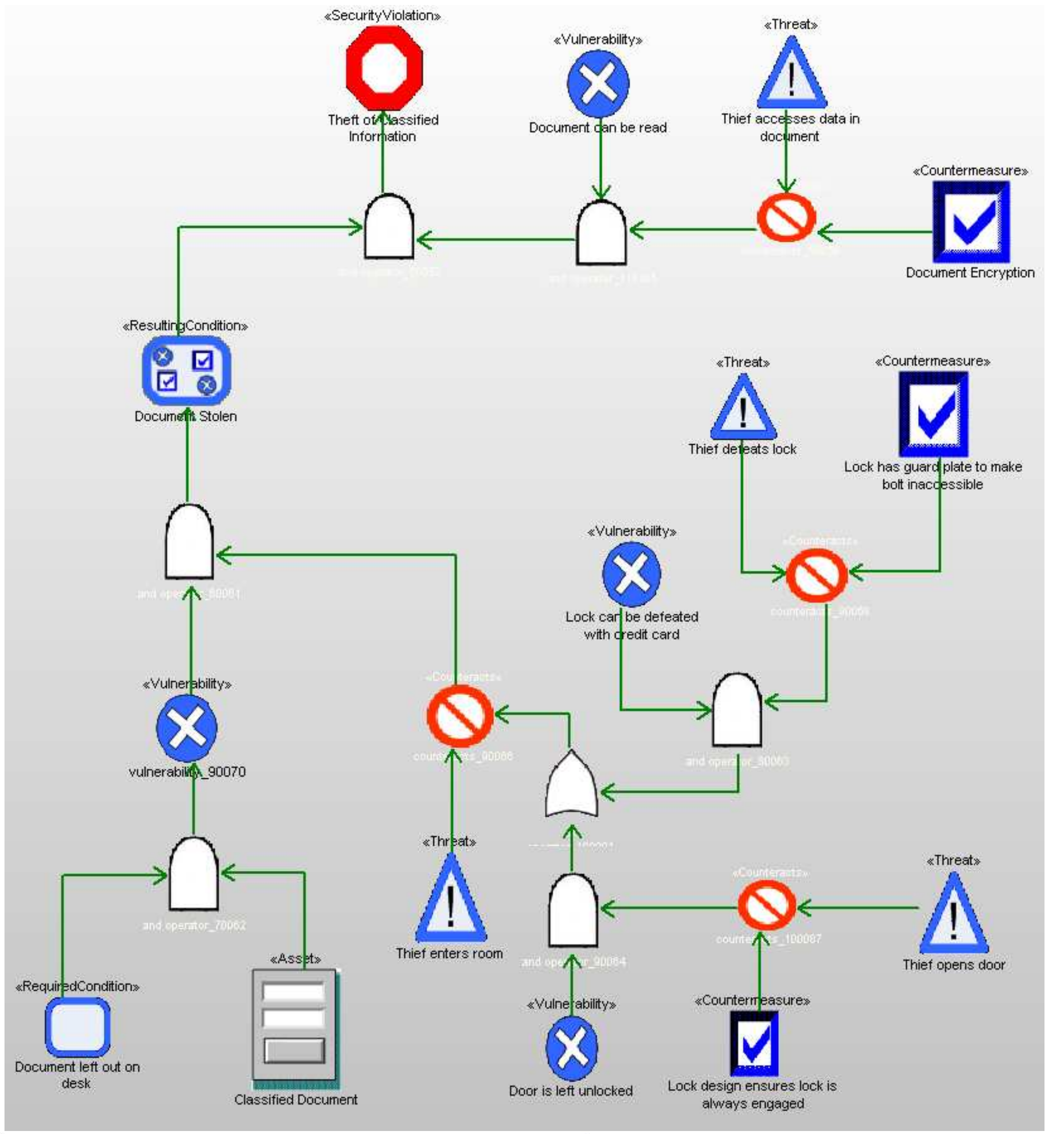


FMEA / FMECA

Failure Means and Effect Analysis														
				Pre-action								Post-action		
Service / Function	Failure Mode	Faults	Failure Effects	Likelihood (1=impossible, 10=certain)	Severity (1=no effect, 10=catastrophe)	Detectability (1=certain, 10=no detection)	RPN (= sev * likely * detect)	Existing Control Measures	Recommendations	Responsible	Actions	Likelihood	Severity	RPN
Braking		Pedal stuck	Pedal doesn't move; No braking action occurs	3	9	10	270	none	Make pedal assembly self lubricating	Joe	Added sealed piston with lubrication	2	9	180
	Pedal nonresponsive	Pedal position sensor fails	Pedal depresses; No braking occurs	4	9	8	288	start up comm check with sensor	Use 3 pedal position sensors	Susan	Added 2 more sensors with voting	2	9	144
		CAN Bus failure	Braking message not conveyed	3	9	2	54	continuous monitoring of CAN bus	none	n/a				
	Comm fault	Loose bus connector	Braking message not conveyed	4	9	9	324	continuous monitoring of CAN bus	update monitoring to send lifeticks to every node on bus	Samuel	updated lifetick protocol	2	9	162
	Processing fault	Braking ECU Failure	No braking occurs	2	9	2	36	Lifeticks every 1.0 seconds	none	n/a				
		Brake actuator processing failure	No movement of brake master cylinder	2	9	10	180	none	Include in lifetick poll	Samuel	updated lifetick protocol	1	9	90
		Brake hydraulic pressure low; O-ring leak; gasket leak	reduced or no braking force	7	9	10	630	none	Add hydraulic pressure sensor; Alert driver	Joe	Added sensor; updated periodic driver messages	3	9	270
		Push rod fracture	Hydralic pressure not increased on braking	1	9	10	90	none	none	n/a				
		Vacuum boost leak	Reduced braking force	6	4	6	144	sensor on pressure differential valve	none	n/a				
	Wheel actuator fault	Brake pads worn	Reduced braking force	9	4	10	360	none	Add periodic driver reminder for service every 6 months	Samuel	updated periodic driver messages	4	6	240

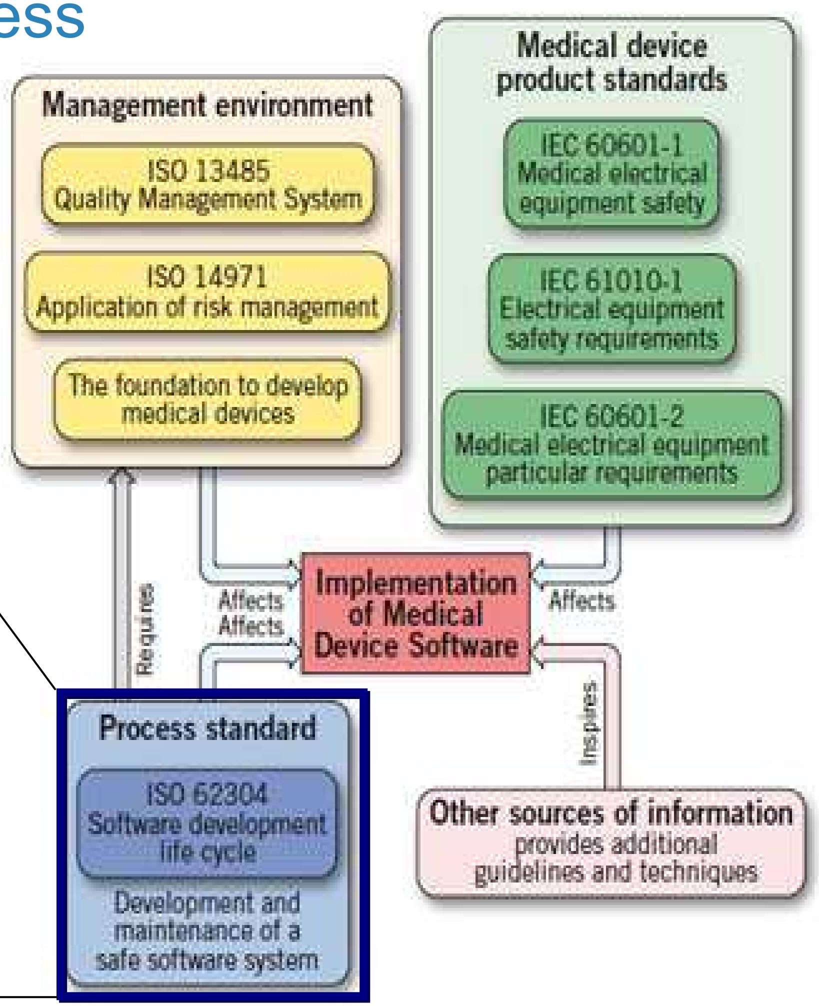
Security Analysis Diagram

- Security Analysis Diagram (SAD) is like an 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



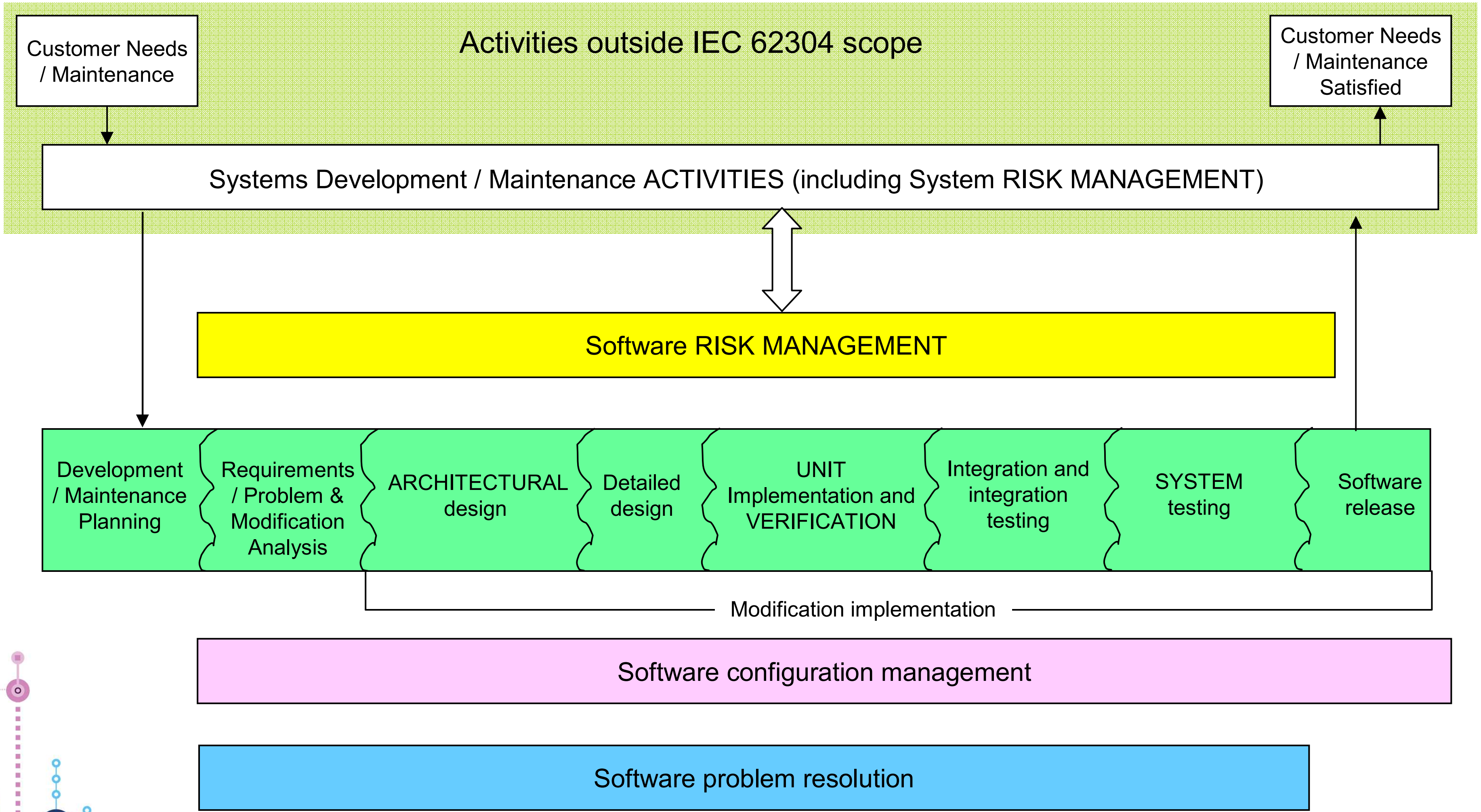
Standards Landscape and Process

1. Scope
2. Normative References
3. Terms and Definitions
4. General Requirements
5. Software Development Process
6. Software Maintenance Process
7. Software Risk Management Process
8. Software Configuration Management Process
9. Software Problem Resolution Process

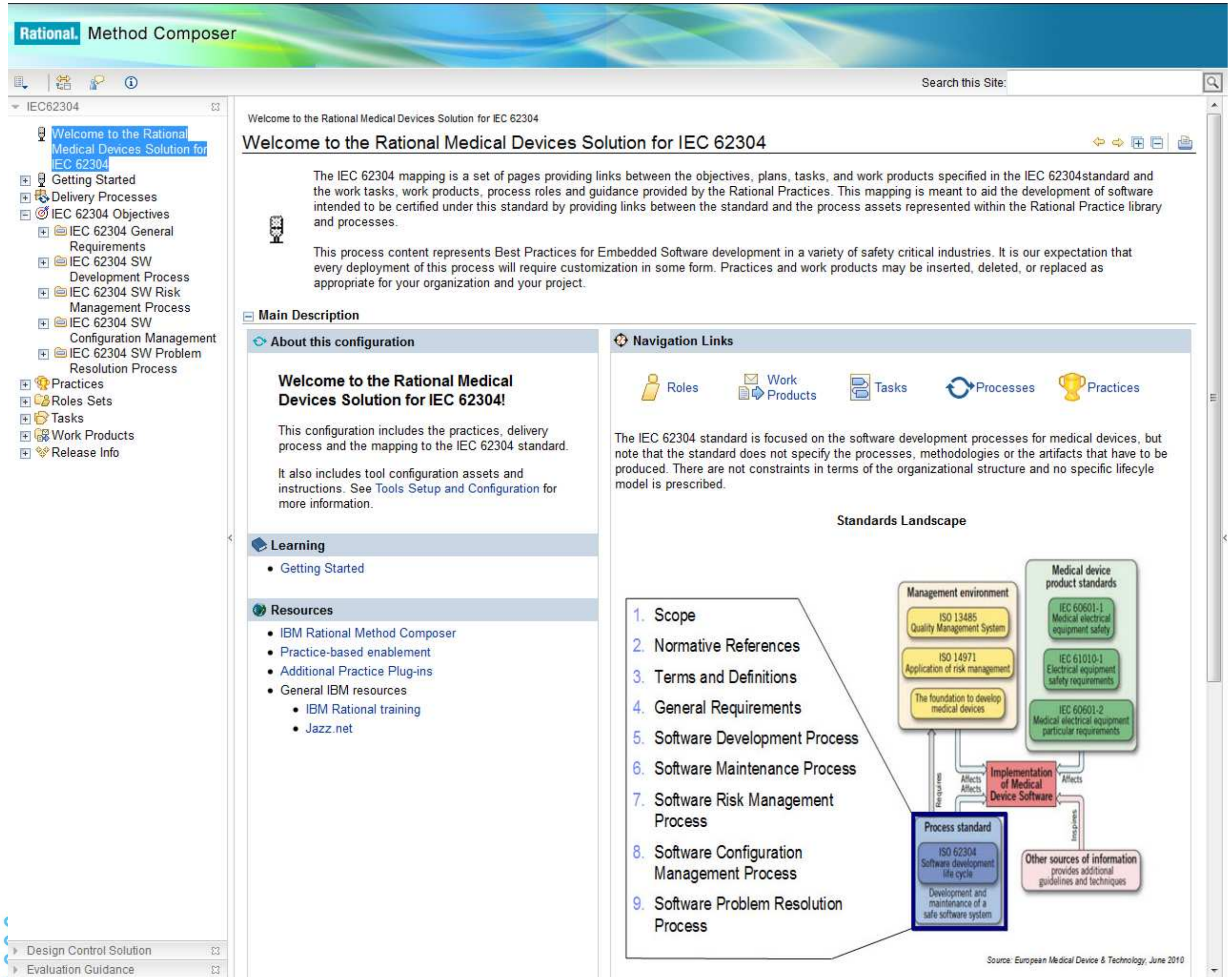


Source: *European Medical Device & Technology, June 2010*

Overview of SW Development / Maintenance from IEC 62304

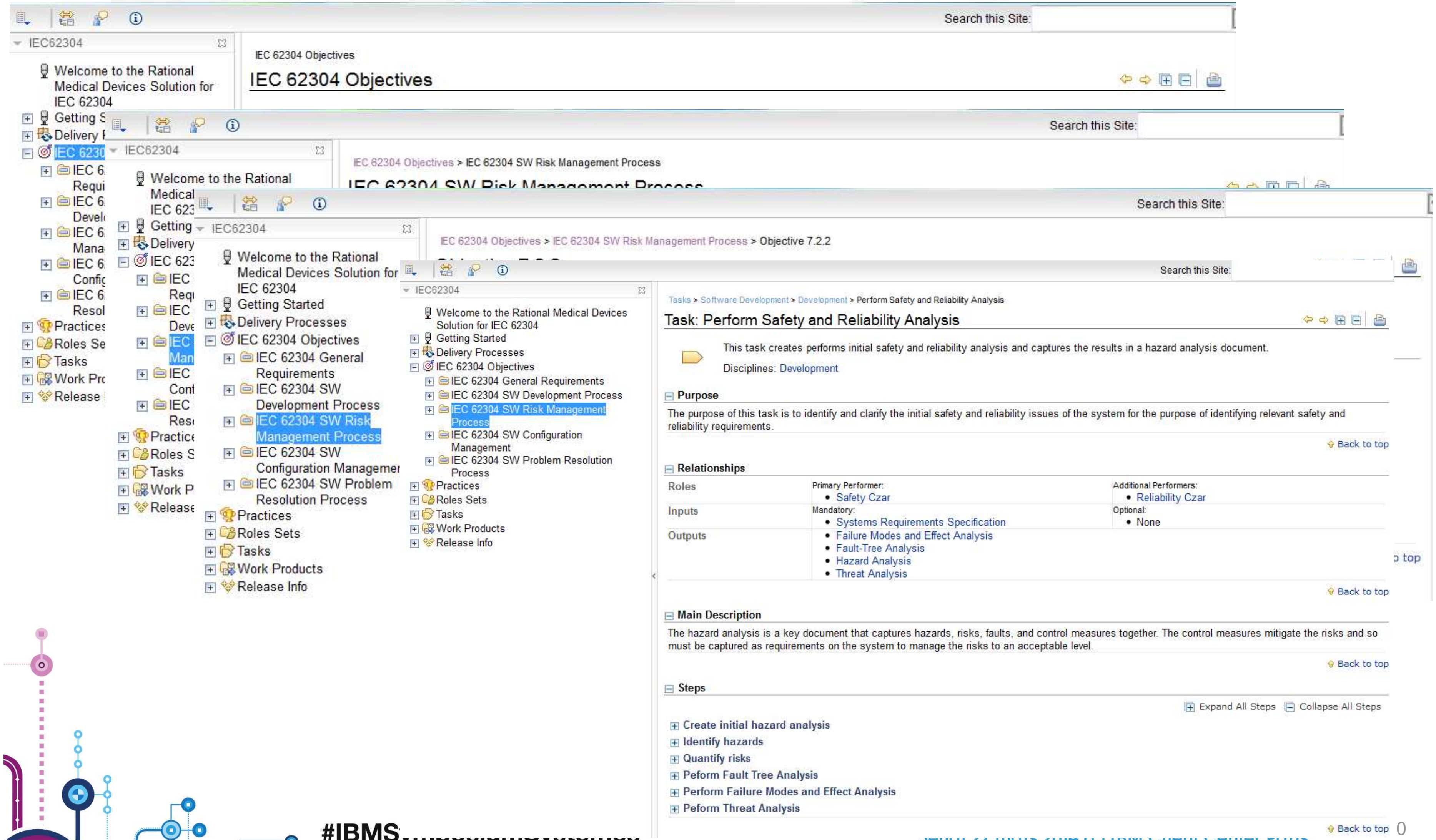


Practices for Dependable Systems Development



Mappings of standards to process elements for
- CMMI
- DO-178B (avionics)
- IEC 62304 (medical)
- EN50128 (rail)

Harmony Mapping to IEC 62304



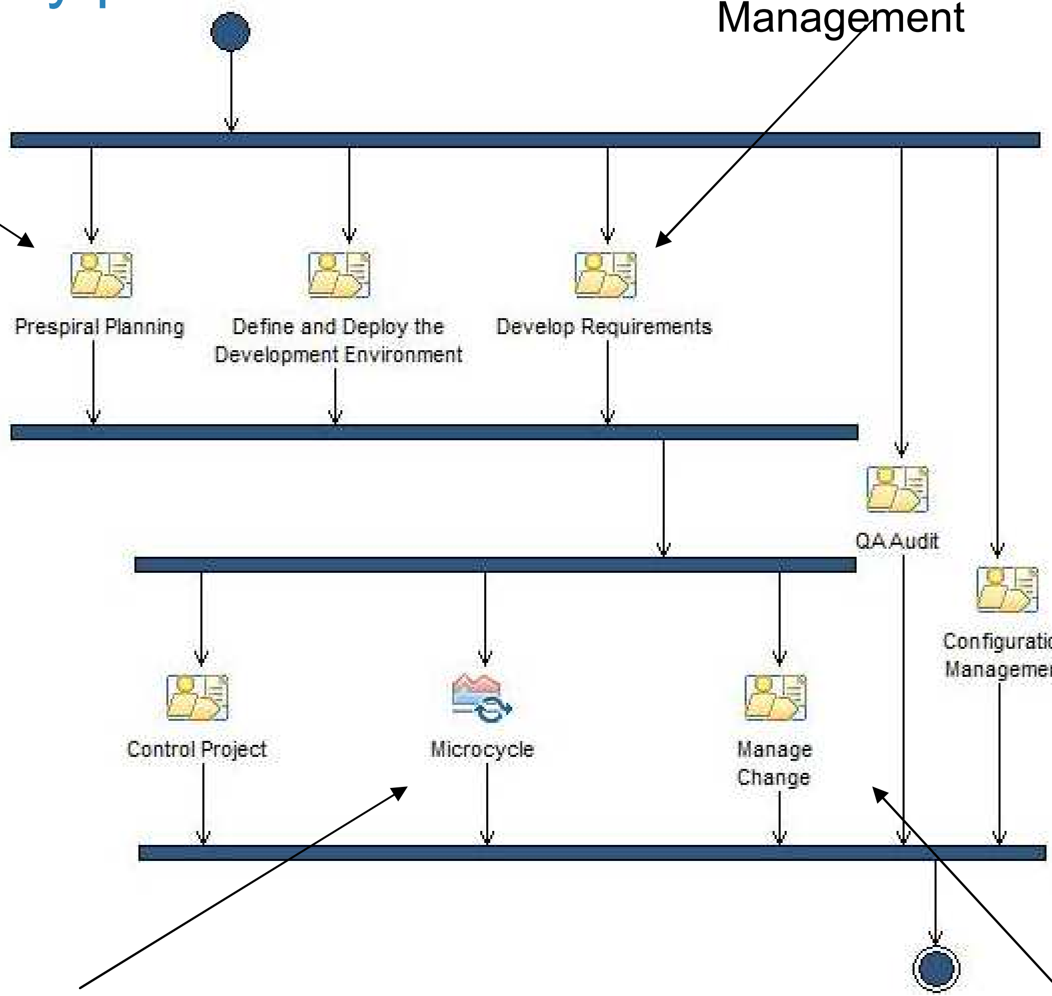
Harmony™ Embedded Software Practices

- Dynamic 2-tier planning
  - Update plan based on Truth on the Ground acquired via on-going measurements
- Test Driven Development
  - Develop & apply test cases at the same time you develop your software
- Continuous integration
  - Integrate and test a common baseline at least daily
- Incremental Development
  - Incrementally construct and validate your software every 4-6 weeks
- Optimizing with design patterns
  - Identify & rank design criteria and identify patterns that address your needs
- Active (project) risk management
  - Use a risk management plan to reduce risks throughout your project
- Frequent project retrospectives
  - The "Party phase" at the end of each increment evaluates how the project is progressing
- Use model-code associativity
  - Use automation to ensure models and code always remain in sync
- Automated Documentation generation
  - Use tooling to generate required certification evidence *from the work you're already doing*

Harmony/ESW Delivery process

Practice: Requirements Management

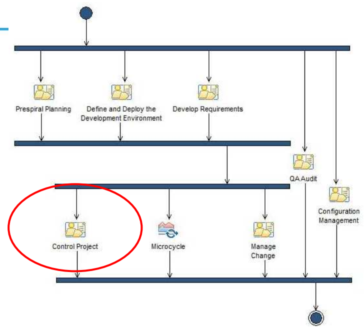
Practice: Dynamic Two Tier Planning



Practice: Incremental (Iterative) Development

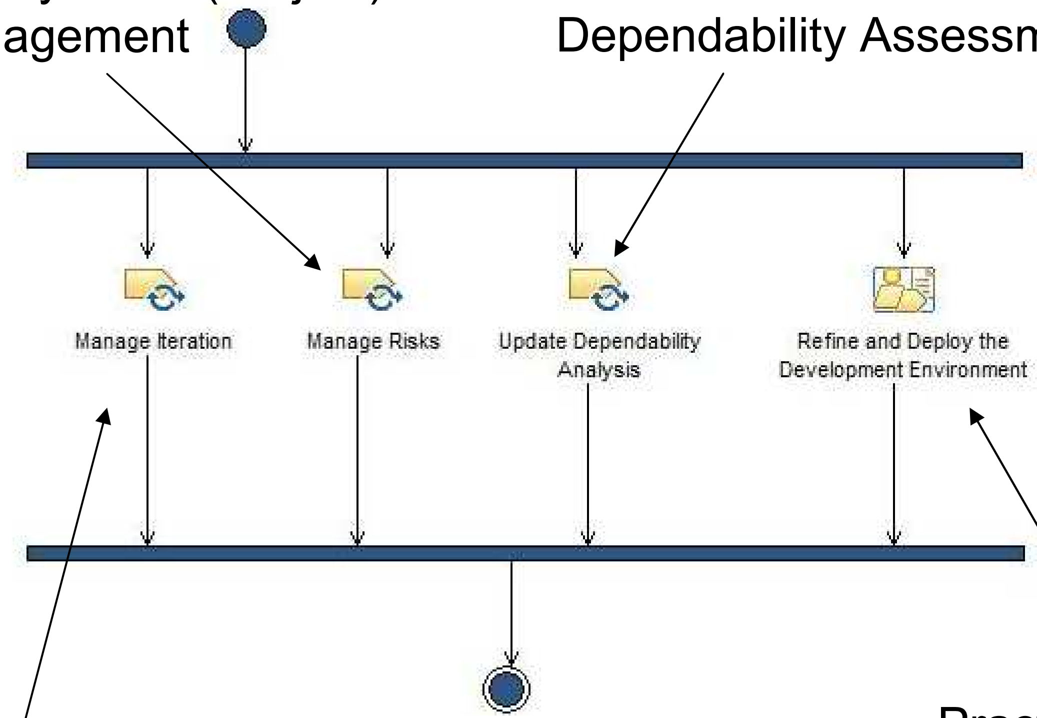
Practice: Change Management

Control Project



Practice: Dynamic (Project) Risk Management

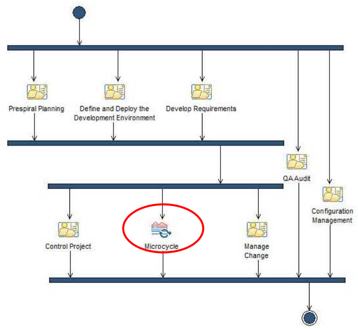
Practice: Continuous Dependability Assessment



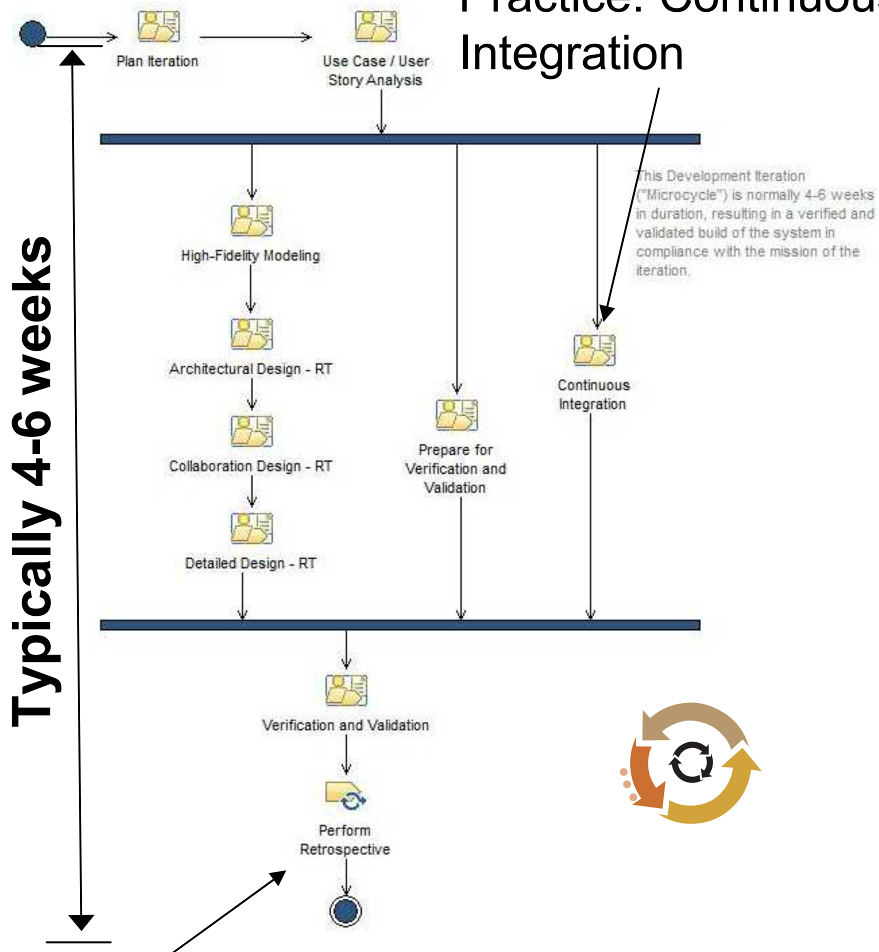
Practice: Manage Project

Practice: Continuous Process Improvement

Harmony Development Iteration



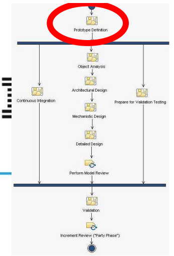
Practice: Continuous Integration



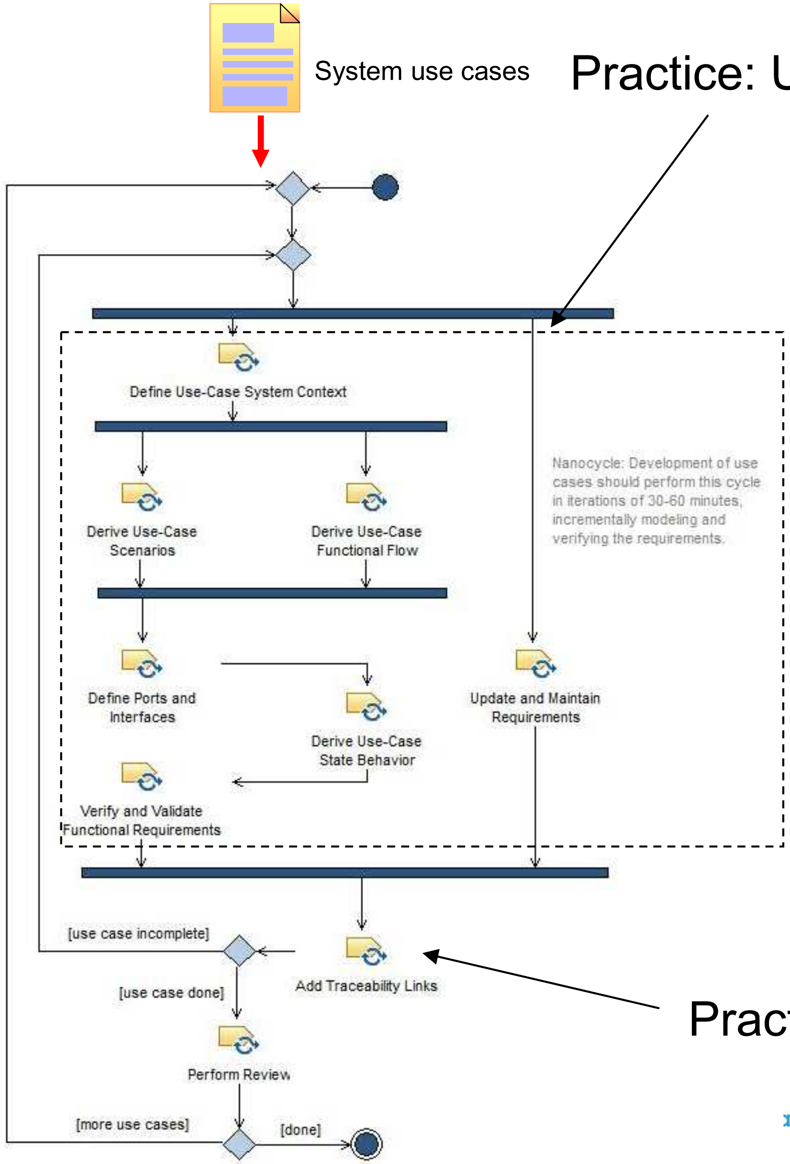
Practice: Incremental (Iterative) Development

Practice: Frequent Project Retrospectives

Incremental Requirements Analysis

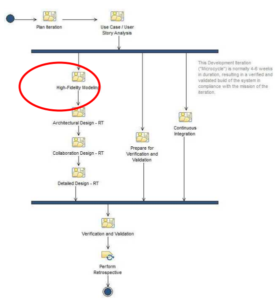


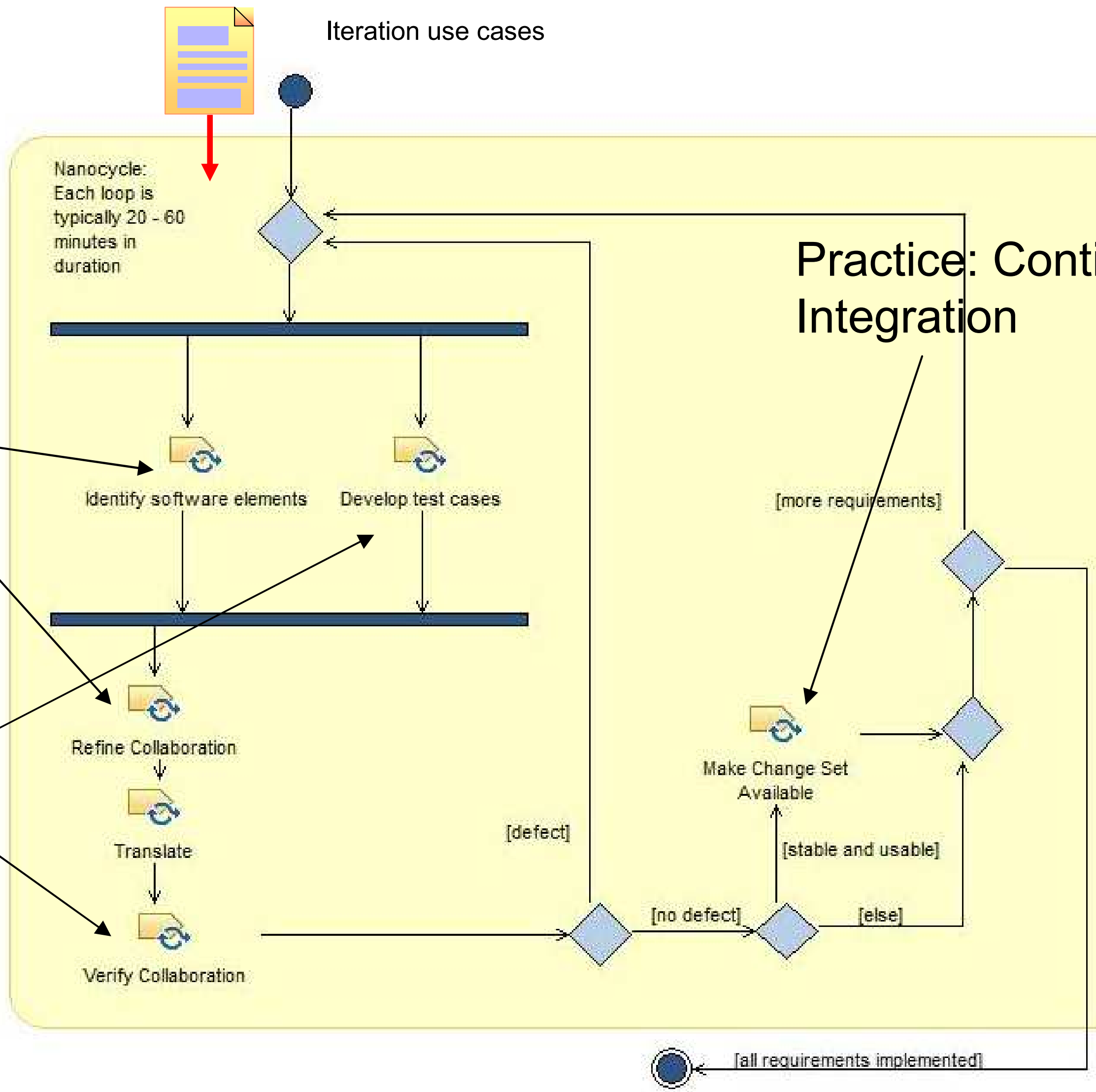
Practice: Use Case Modeling



Practice: Manage Traceability

High Fidelity (Functional) Modeling



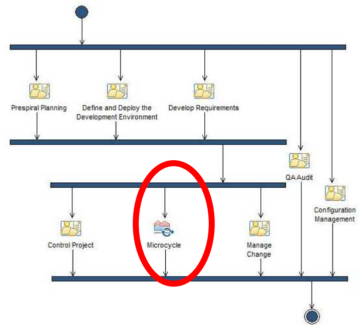


Practice: Continuous Integration

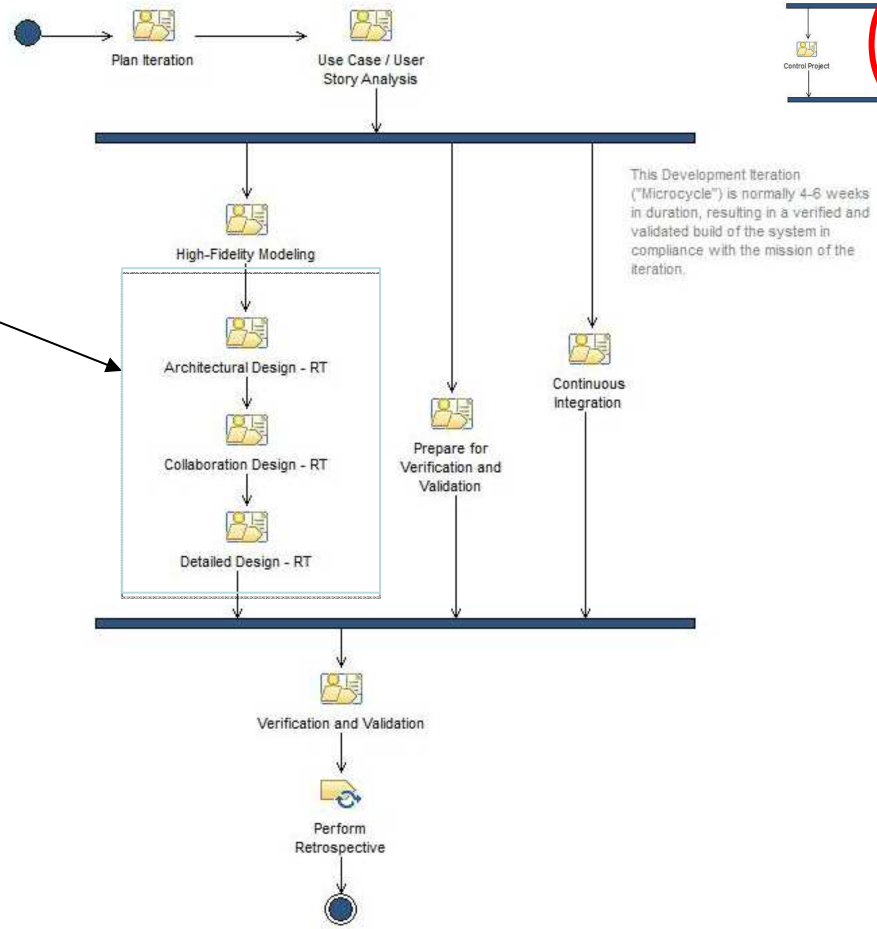
Practice: Modeling

Practice: Test Driven Development

Iterative & Incremental Design



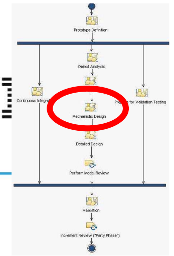
Practice: Design at Three Levels



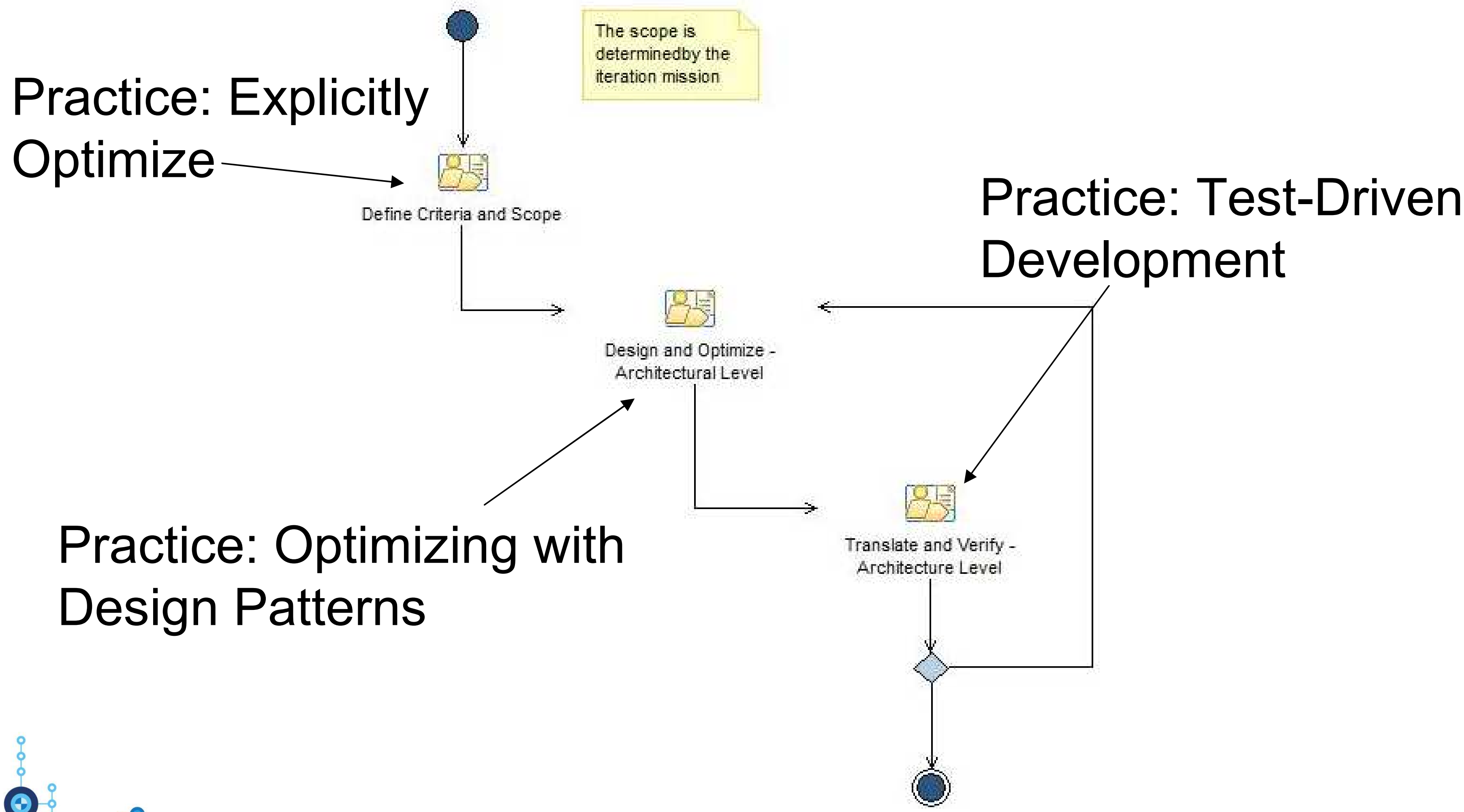
Architectural Design

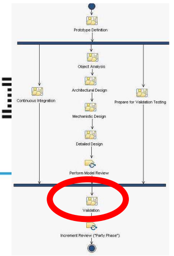
- Architectural Design consists of 5 key interrelated model views:
  - Concurrency and Resource View
  - Deployment View
  - Distribution View
  - Dependability View
  - Subsystem and Component View

Each Architectural View will have its own design patterns. The complete system architecture is the set of design patterns used in all of the various aspects of physical architecture.



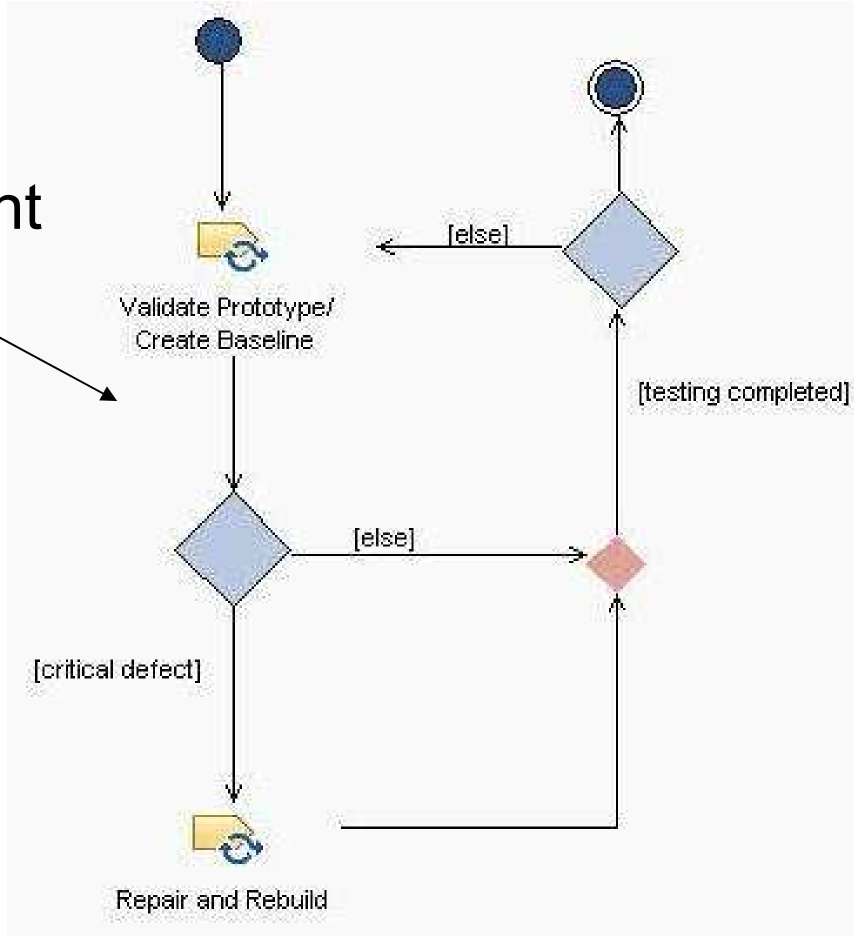
Architectural Design



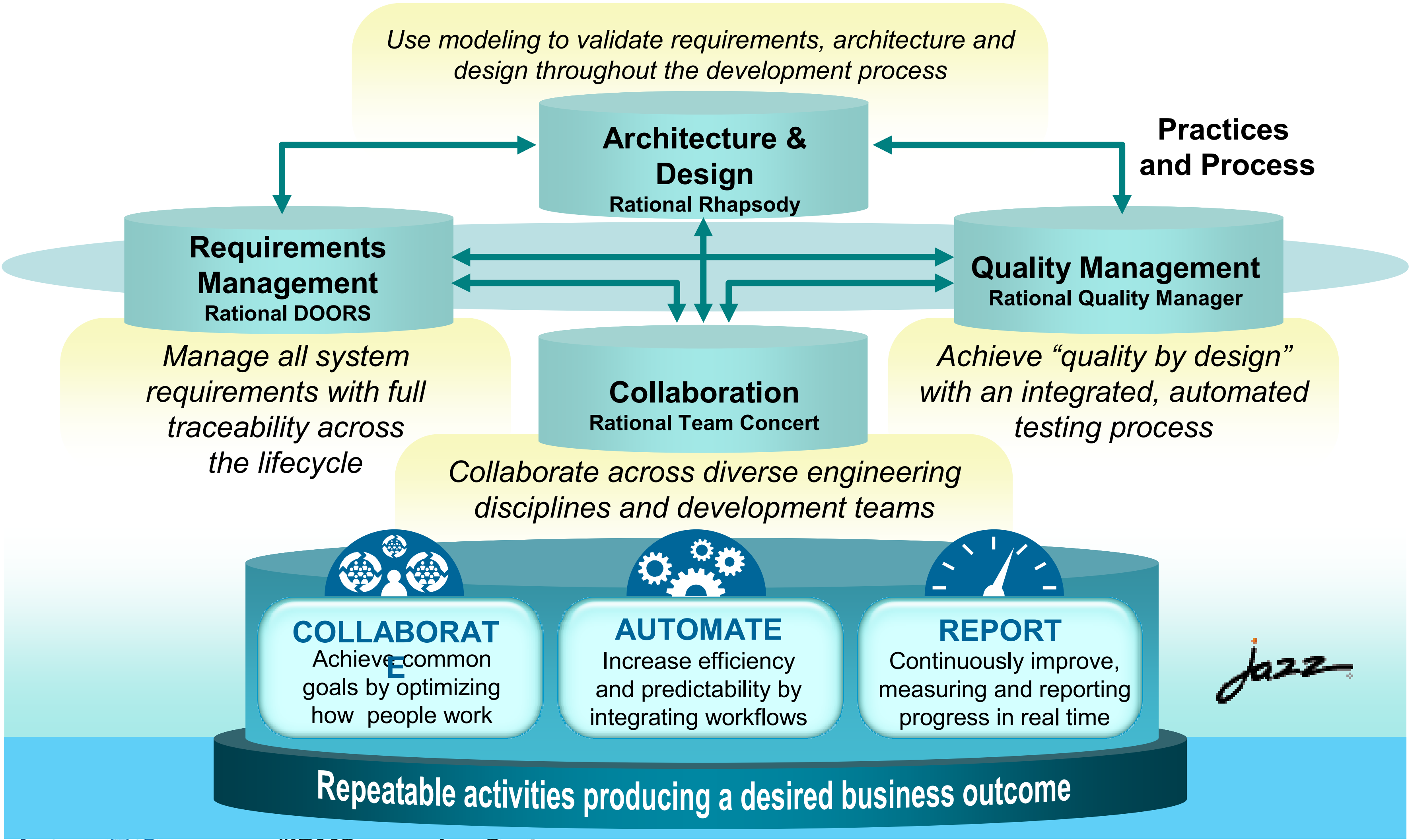


Verification and Validation

Practice: Incremental (Iterative) Development and Verification

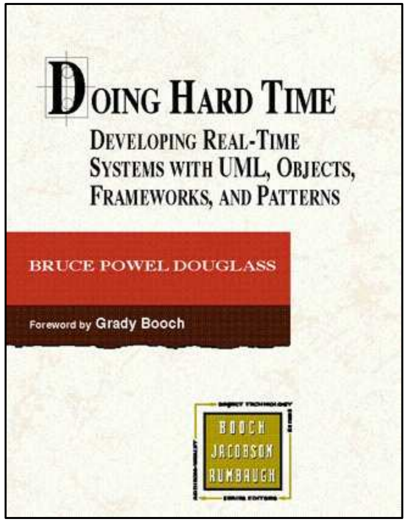


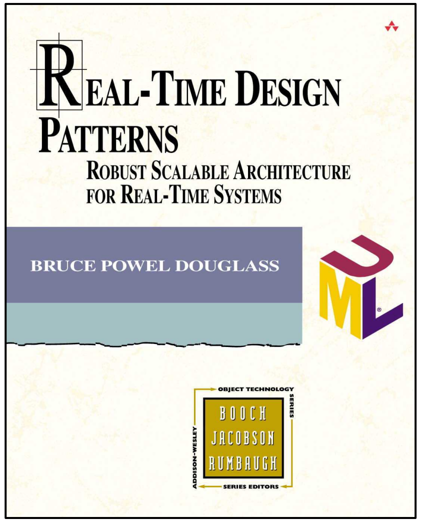
Putting it all together with Rational System Solution

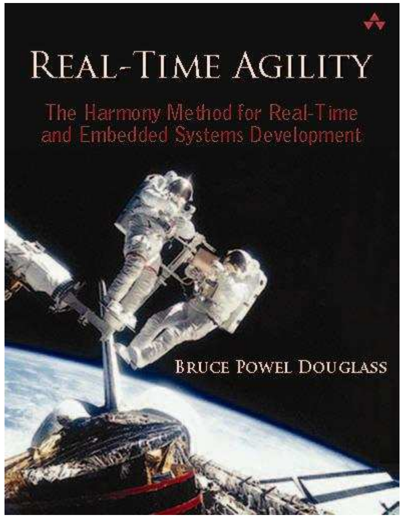


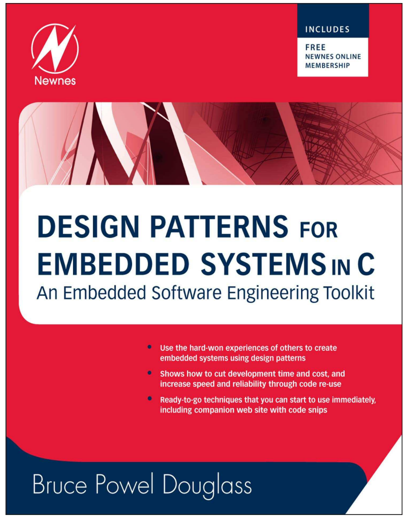
Summary

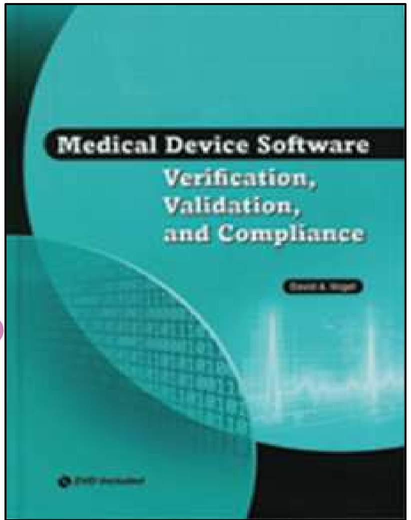
- Dependability has three aspects
  - Safety
  - Reliability
  - Security
- Dependability is created with
  - Initial and on-going risk assessments
  - Traceability among relevant work products
  - Verification activities (testing, QA)
- IEC 62304 is used as a standard metaprocess (process objectives) for medical device delivery. Includes metaprocesses for
  - Software Development
  - Software Maintenance
  - Software Risk Management
  - Configuration Management
  - Problem Resolution Management
- The Harmony process is an agile process providing a set of best practices that
  - Perform the tasks to meet the FDA device certification needs
  - Map to the IEC 62304 standard
  - Are supported by the Rational Systems and Software Solution

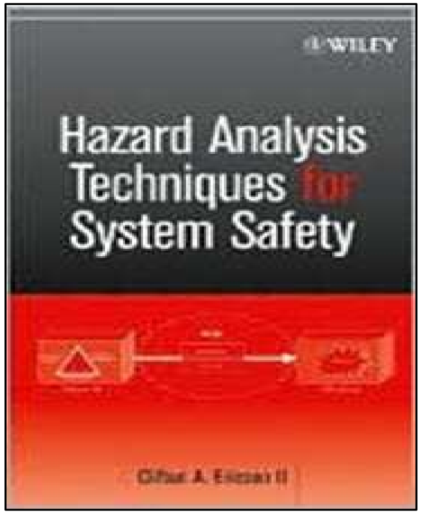

Doing Hard Time
Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns
Bruce Powel Douglass
Foreword by Grady Booch
Booch Jacobson Rumbaugh

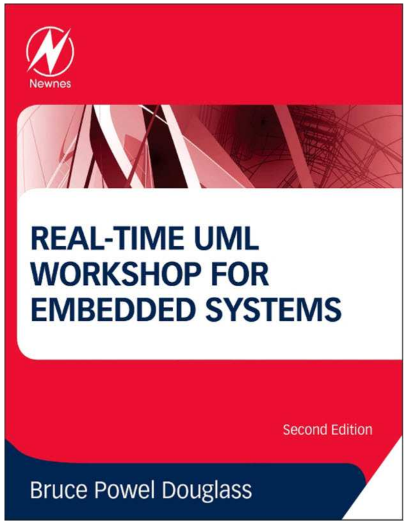

Real-Time Design Patterns
Robust Scalable Architecture for Real-Time Systems
Bruce Powel Douglass
UML
Booch Jacobson Rumbaugh

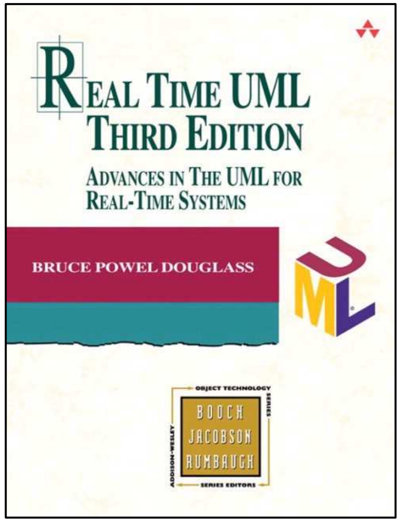

Real-Time Agility
The Harmony Method for Real-Time and Embedded Systems Development
Bruce Powel Douglass


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


Medical Device Software
Verification, Validation, and Compliance


Wiley
Hazard Analysis Techniques for System Safety


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


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