EDICT™ Overview

Dr. Chris J. Walter

cwalter@wwtechnology.com
410-418-4353

Brian LaValley

blavalley@wwtechgroup.com
401-348-8525

July 27, 2008
Challenges

- Modern high confidence systems are increasingly:
  - network-centric
  - distributed embedded systems
  - real-time
  - shared platforms

- Performance and dependability of systems are coupled to hardware/software arrangement:
  - non-standard/custom
  - multiple vendors
  - reuse is problematic
  - tightly coupled designs

- Costly development and upgrades
Integration Challenges

- Integration may reduce the physical connections but radically increase information/logical connections
- Not all connections/dependencies are apparent
  - System failures/rework due to
    - sneak paths
    - hidden interactions
    - unforeseen trade-off consequences
    - unnecessary complexity
- How do we manage the complexities?
  - from specification, design, analysis, certification

A framework for representing architectural concerns from a common information base is essential for performing trade-offs and managing risk

© Copyright 2008 WW Technology Group. All rights reserved.
Model Based Systems Analysis Benefits

- Early Identification and resolution of architectural problems leads to:
  - More consistent and evolvable systems
    - When problems are found later (integration or deployment test), it may not be possible to provide the best technical fix due to the high costs of rework
    - As a result problems are often resolved with a band aid approach that addresses the symptoms but does not provide an elegant solution
    - The accumulation of band aids leads to fragile systems that are difficult to evolve in the future
    - *These accumulated effects have large cost impacts to systems with long lifetimes*
  - Less Test Cost and Greater Test Confidence
    - Reductions in the level of system complexity and increases in understanding of expected and desired system behaviors leads to more testable systems
      - This is born out by the results of the application of complexity measures to software
Cost Effectiveness Of Model Based System Analysis

- Architectural issues are a leading source for integration problems
- Without systematic upfront analysis these problems are costly to repair
- Application of complexity, safety and dependability analysis leads to addressing the issues early on

Source: https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/code/212.html?branch=1&language=1
EDICT Vision

- An integrated tool suite for the *specification*, *design*, *evaluation* and *deployment* of high confidence systems
  - Utilize a Model Driven development approach to support the specification and evaluation of system properties *throughout the system lifecycle*
  - Provide views and tools that are tuned to the needs of system stakeholders cross cutting concerns and activities
    - Safety – Security – Dependability - Performance
    - Specification – Design – Validation – Certification
  - Utilize analysis after system deployment to support
    - Upgrades – Changes In Threat – Changes In Operations

© Copyright 2008 WW Technology Group. All rights reserved.
EDICT Vision Is Full Lifecycle, Model Driven, Cross Domain Analysis
State of the Practice Safety and Dependability Analysis for Certification

Generation of Safety and Dependability Artifacts is a manual, labor intensive process.

Safety Artifacts
- SSHA
- Fault Trees
- Inspections

Certification

System Architecture Description
- Documents
- Models
- Personnel

System Implementation
- Manual translations are in efficient and prone to error

Dependability Artifacts
- FMEA Models
- Reliability Models
- Manual FT Analysis

Certification

© Copyright 2008 WW Technology Group. All rights reserved.
Integration of Safety and Dependability Analysis for Certification

EDICT Tools
System Composer

System Architecture Description
Architecture Description Model

Documents
Supporting Models

EDICT Tools
Safe Suite

Safety Artifacts
- SSHA
- Fault Trees
- Safety Cases
- Inspections

Certification

EDICT Tools
Depend Suite

Dependability Artifacts
- FMEA Models
- Reliability Models
- FT Analysis
- Dependability Cases

Certification

© Copyright 2008 WW Technology Group. All rights reserved.
EDICT™ has been designed to be Adaptable and Extensible
EDICT is built on the Eclipse platform for rapid development of professional-grade applications

Extensible UI features (menus, tool bars, etc.) come with the Eclipse platform.

“Perspectives” may be defined that facilitate task based reconfiguration of the UI layout.

Editors may be added to the Eclipse Platform for custom applications. Editors may be text based, form base, or graphical.

Views are built-in to the Eclipse platform that may be used by applications. Applications may also define their own views and add them to the platform.

01.25.2007
The OSATE tool provides a development environment for AADL.

System architecture model information is stored in OSATE project structure.

AADL editor and underlying compiler capabilities facilitate AADL specification of system architecture composition and component properties.
EDICT Design Workspace View
Simplifies Model and Analysis Navigation

- Models are organized by category, domain and type.
- Problems associated with models are depicted by error / warning icons.
- Individual models are presented in a simplified manner (independent of physical format and location of storage).
- Different view organizations are supported.
- Design-level operations can be performed.
EDICT
Error Handling Domain

SBIR: OSD03-023

© Copyright 2008 WW Technology Group.
All rights reserved.
Error Handling SBIR Objectives

• Facilitate Improvements in System Error Handling By:
  – Early and continuous attention to errors and their effects
  – Establishment of processes for the analysis and implementation of effective error handling strategies

• Develop Tools That Support:
  – Capture and tracking of dependability properties
  – Characterization of system error environment

  • Underlying error cases
  • Error detection strategies and mechanisms
  • Error reporting
  • Error handling abstractions
  • Error propagation characteristics and effects
  • Error localization and recovery

  – Establishment and realization of error handling patterns that mitigate issues identified in the modeling and analysis stages
  – Interpretation of model driven information into schema suitable for realization using middleware driven solutions
Error Regions are Powerful Abstractions that Reduce the Scope and Complexity of Analysis

Error Regions with containment properties drive analysis to determine whether errors that occur within a defined subset of system components can impact components outside that subset.

Error Regions with protection properties drive analysis to determine whether errors that occur external to a defined subset of system components can impact components in that subset.
System Composer Error Handling Extension

• This extension provide the capabilities for *augmenting* the base system architecture and component models with error handling related models and properties.
• Error handling related models currently include:
  – *Error Semantics* – describes the individual kinds of errors that may occur in the system
  – *Component Error Descriptors* – describes the errors that may be exhibited by a specific kind of component in the system model
  – *Component Error Model* – describes the error characteristics of a *specific component* in the system model (including exhibited errors and transformed errors)
  – *Error Mitigator* – describes a kind of mechanism that is capable of detecting and/or tolerating certain kinds of errors
  – *Component Error Mitigator Model* – describes which kinds of Error Mitigators are fielded with a *specific component* in the system model
Adaptiveness in Error Domain

Traditional Models
- Domains have been traditionally handled as discrete cases.
- Does not facilitate scalable fault tolerance or efficient resource utilization

highly available
Benign Masking Algorithms

highly dependable
Byzantine Agreement Algorithms
System Composer Provides Editors To Specify Error Aspects
System Analyzer Error Handling Extension

• This extension provides the capabilities configuring and executing analyzers that examine the system from the error propagation and error handling perspective.
• Error handling related analyzers currently include:
  – **Error Region Specification Analyzer** – determines whether or not a given ERS is supported by the currently modeled system. It searches for error scenarios that cause error region boundary crossing violations.
  – **Error Propagation Analyzer** – general purpose analyzer that facilitates free-form exploration of the error propagation characteristics of the system.
The Error Region Specification Set (ERSS) editor supports the modeling of error regions and their properties.

The “Region Allocation” page allows a user to allocate specific components from the system architecture model to a specific error region.
The Error Region Specification Analyzer evaluates a system’s ability to uphold ERS properties

The “Results” page presented here shows a full results visualization after the analyzer is able to execute.

The analyzer may encounter problems along the way. These are presented first.

For each error region in the ERS, a full report of errors that crossed the region boundary is provided.

Details of a selected boundary crossing report are provided.
EDICT
Safety Domain

SBIR: AF04 - 246
EDICT SafeSuite

- Specify run-time V+V safety properties

- Assess system properties at architectural level

- Assessment drives both verification as well as specification and design of run-time monitors

- Tool framework capable of integrating formal methods

- Tools provide rapid analysis capability throughout life cycle

Accomplishments:
- Developed model driven approach to mitigate safety concerns
- Demonstrated open standard based architecture analysis tools
- EDICT Tool gaining support among avionics suppliers

Model Driven Design Enables **Specification, Evaluation and Mitigation Of Safety Concerns Early in Design Cycles**
SafeSpec Tool Maps Safety Concerns To System Architecture

- Safety commitments that the system must uphold are captured within SafeSpec
  - Current version uses fault trees as the method for specification of safety concerns and root causes for unsafe system operations
- Safety Events are mapped to system architecture elements and the fault models that have been established by system architects.
  - Ensure continuity of assumptions between system designers and safety analysts
- Safety domain analysis tools are provided
  - Fault Tree Assessment
    - Specify analysis cases that evaluate fault tree assumptions against the modeled system architecture
  - Hazard Source Analysis
    - Analyze architecture to find potential error sources that are not contained in fault tree logic
  - Hazard Risk Analysis
    - Analyze error sources that are driving overall hazard risk
  - Hazard Mitigation Maps
    - Show the distributed set of error mitigators that are employed to prevent system hazards at run-time
SafePlace Evaluates Run-Time Protections

- SafePlace analysis determines if any of the system safety commitments can be compromised through design-time or run-time errors
  - Analysis driven by safety commitments and analysis from SafeSpec
  - SafePlace utilizes error models and error mitigator properties specified though the system composer
- Ability of system to meet safety commitments on-line is assessed
  - Analysis Cases from SafeSpec are evaluated with error propagation in the system model
  - Propagation data is used to determine impact of event
- Error detection and toleration mechanism are integrated to prevent safety property violations
  - SafePlace provides visualizations and metrics for the placement of additional error mitigators in the system to preserve desired system safety properties
SafePlace Evaluates Error Propagation And Impacts
SafePlace Provides Metrics And Visualizations To Aid In Run-time Mitigator Placement
EDICT
Information Assurance Domain

SBIR: AF07-060
Multiple Independent Levels of Security/Safety (MILS/S)

• Goal is to protect the flows of information and guarantee that information assigned to different security levels is handled appropriately.

• Significant challenge to design MILS/S that is guaranteed to perform correctly with respect to security and safety.
  – John Rushby first introduced concept in the early 1980's for architecting secure systems using a separation kernel to reduce the security burden.
  – Separation kernel mediates interactions between applications and enforces a security policy of information flow and data isolation on those interactions.

• Our approach builds a hierarchy of system Information Assurance properties that are used to build and certify the system
Property Based Information Assurance

• Goals of Property Based IA
  – Reduce development, certification and accreditation costs
    • Break problems into manageable pieces
    • Analyze pieces independently
    • Provide a structured method for synthesis of pieces into a system that supports incremental certification/re-certification
  – Support top-down specification/decomposition of problem
    • Establish critical property requirements based on system needs
    • Flow down properties to drive system architecture and component level selection
  – Support bottom-up certification chains for full traceability
    • Flow up component level IA properties to system architecture and requirements
      – Integrate stand-alone component certifications (CC) into a system context
    • Provide property traceability for certification evaluation
Property Based Information Assurance

IA Requirements
- Define Information Domains
- Define functional dependencies between domains
- Define Security Service Requirements

IA Architecture
- Define Processing
- Define Information
- Define Authorized Info Flows
- Define Partitioning
- Define Security Services

Logical Architecture Model (Deployment Independent)
- Map Process/Information to implementations
- Map partitions to implementations
- Map authorized flows to architecture flows

IA Deployment
- Define Processing Implementation
- Define Information Storage
- Define Connections/Data Flow
- Define Partition Implementation

System Deployment Model (AADL with IA annotations)

Properties to Specify/Analyze
- Confidentiality
- Access Control
- Integrity
- Availability
- Non-Repudiation

Domain
- Functional Dependency

Properties to Specify/Analyze
- Information Flow
- Least Privilege
- Damage Limitation
- Data Isolation

Secrecy
- Colleague
- Diverting
- Interception
- Message Router

Authorized Info Flow
- Partition
- External System

Private
- Information
- Security Processing

Text
- Partition
- External System

System Components
- Text
- MLS Partition
Augmenting EDICT with Möbius

• EDICT’s ability to analyze the information in the system’s descriptive model base is currently static in nature.
  – Static Analyzers can tell you *what is possible*
  – Stochastic/Simulation-based Analyzers can tell you *what is probable*

• To gain valuable insight into the performance and behavior of the system
  – in the presence of errors, the aspects of time, mode, state, and probability must be introduced.

• We are teaming with the PERFORM group at University of Illinois
  – further EDICT’s integration with the Möbius tool technology to address these needs.
EDICT Summary

• **EDICT addresses multiple domains**
  – Safety
  – Information Assurance
  – Error Handling

• **Strong Sponsorship**
  – Air Force (Safety & Information Assurance)
  – OSD / Navy (Error Handling, Safety & Certification)
  – Army (interest in Error Handling & Information Assurance)
  – MDA (Information Assurance)

• **Strong Interest**
  – Boeing - Northrop Grumman
  – BAE systems - Rockwell
  – General Dynamics - and others
EDICT Transitions

• **Presentations**
  – Invited to present to European community,
    • *Artist2 Symposium on Integrated Modular Avionics*
  – Invited participation in *National Workshop on High-Confidence Automotive Cyber-Physical Systems*

• **Strong interest from AADL community**
  – Software Engineering Institute
    • ongoing dialog
  – Link on the AADL website to EDICT
EDICT Potential Transitions

- Rockwell
  - internal evaluations
- Northrop Grumman
  - B2 and other applications
- BAE systems
  - application to DARPA program
  - potential interface to GME
  - identified opportunities for additional work (JSF)
- Boeing Space & Information Systems
  - evaluating application to multiple programs