U.S. NAVAL RESEARCH LABORATORY

AUTONOMOUS SYSTEMS: DEVELOPING HUMAN TRUST

Connie Heitmeyer
Head, Software Engineering

Center for High Assurance Computer Systems
Naval Research Lab
Washington, DC
constance.heitmeyer@nrl.navy.mil
202-767-3596

NATO
Collaboration Support Office
December 4-6, 2018

U.S. NAVAL RESEARCH LABORATORY

ASSURING AUTONOMOUS SYSTEMS: WHAT ARE THE CHALLENGES

Connie Heitmeyer
Head, Software Engineering

Center for High Assurance Computer Systems
Naval Research Lab
Washington, DC
constance.heitmeyer@nrl.navy.mil
202-767-3596

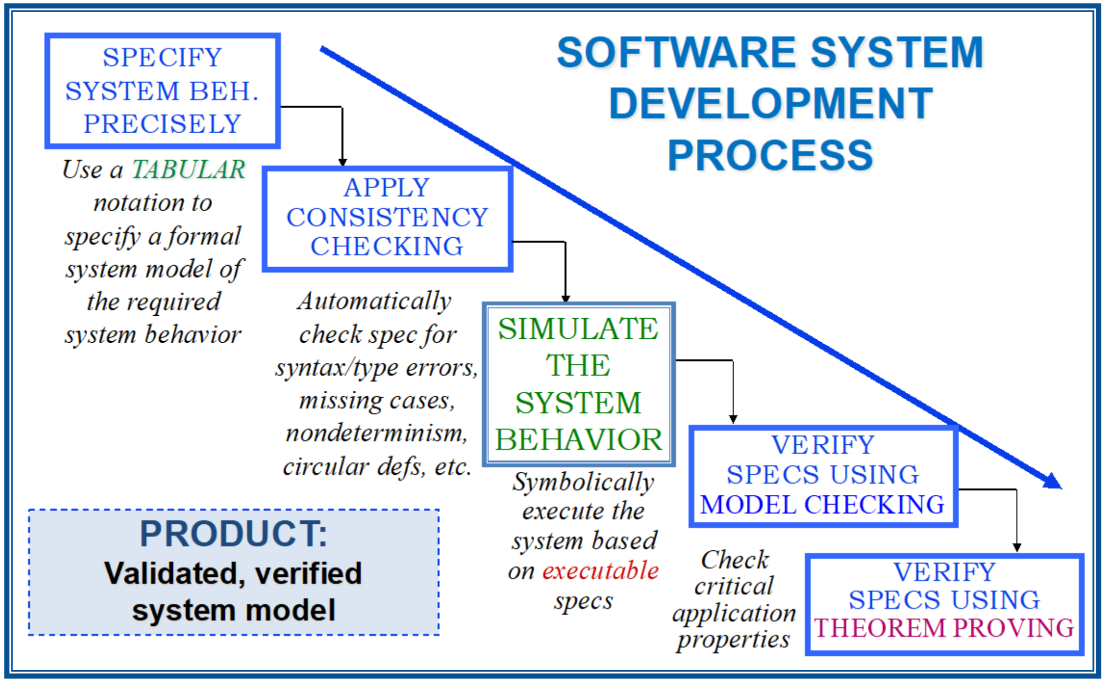
NATO
Collaboration Support Office
December 4-6, 2018

Outline

- My Background
- Definitions
- Challenges in Assuring Autonomy
  - Challenges that have already been addressed
  - Challenges that need to be addressed for autonomy and for other critical systems
  - Challenges unique to autonomy
- Summary

My Background

- 20+ years developing new methods and tools (both formal and informal) for modeling/analyzing requirements of critical software systems
- Research documented in dozens of papers
  - Cited in softw. eng., requirements, and formal methods literature
- Tools distrib'd world-wide
  - Included in univ. courses, software textbooks
  - Many tutorials (conferences, NASA, etc.)



SOFTWARE SYSTEM DEVELOPMENT PROCESS
SPECIFY SYSTEM BEH. PRECISELY
Use a TABULAR notation to specify a formal system model of the required system behavior
APPLY CONSISTENCY CHECKING
Automatically check spec for syntax/type errors, missing cases, nondeterminism, circular defs, etc.
SIMULATE THE SYSTEM BEHAVIOR
Symbolically execute the system based on executable specs
VERIFY SPECS USING MODEL CHECKING
Check critical application properties
VERIFY SPECS USING THEOREM PROVING
PRODUCT: Validated, verified system model

Our process, methods, & tools have been applied to many practical systems, e.g.,

- Safety-critical military systems
- Security-critical comms system
- Autonomous systems (since 2012)

My Earlier Interests

How to obtain assurance of critical systems

My Current Interests

How to obtain assurance of autonomous systems

Terminology

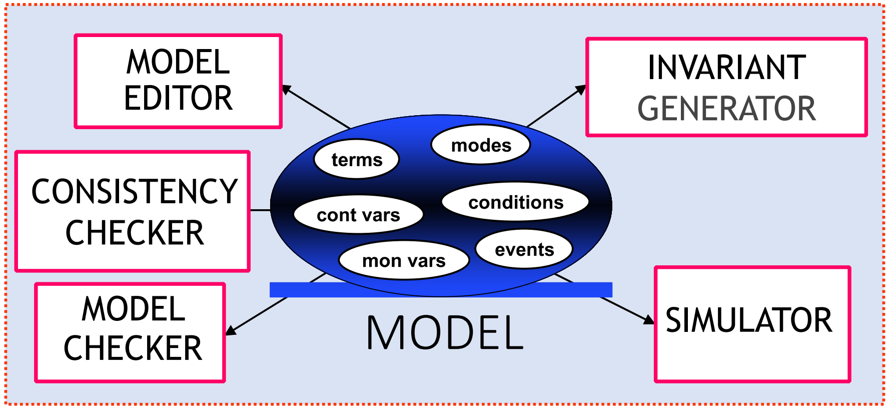
- **Automated:** Automatically controlled operation where the system uses a preplanned set of instructions*
- **Autonomous:** System's ability to make decisions, take actions in presence of uncertainty and to respond to internal and external changes without human intervention*
- **Validation:** Assurance that the system's externally visible behavior is the intended behavior
- **Formal Verification:** Formal proof that the system's externally visible behavior satisfies critical properties, e.g., safety, security, functional correctness, etc.
- **Assurance:** Confidence that the system behavior satisfies its requirements** (also called **Verification**)

*Adapted from Martin Feather, NASA Jet Propulsion Lab, 2018

**Equivalent to J. McDermid's definition that system behaves as intended in its environment of use

Formal Methods (FMs): What are They? What Are the Benefits of FMs?

- **Definition of FMs:** Mathematically rigorous methods and tools for modeling, designing, and verifying critical systems
- FMs make it possible for us
  - To symbolically examine the entire state space of a system
  - To establish functional and safety properties for all possible inputs



C Heitmeyer+, *ACM TOSEM, 1996;*
C Heitmeyer+, *Internat. Journal of Computer Systems Science and Eng., 2005*

- Precisely specify the required system behavior as a mathematical model
- Detect missing cases, ambiguity
- Validate model/check critical properties via model-based simulation
- Formally verify critical properties (model checking, theorem proving)
- Generate invariants from model
- Synthesize code
- Generate tests

All of these require a formal model!

🟪 *performed automatically*

Assuring Autonomy: What Are the Challenges?

1. Challenges that have largely been solved for other critical systems
2. Challenges that are unsolved for autonomous systems and for critical systems that are not autonomous
3. Challenges that are unique to autonomous systems

Future research should focus on the last two sets of challenges

What Are the Solved Challenges?

What processes, methods, and tools
to use in developing assurance of
critical autonomous systems

What Is the Solution?

Apply processes, methods, and tools that have been developed to obtain assurance for
other critical software systems

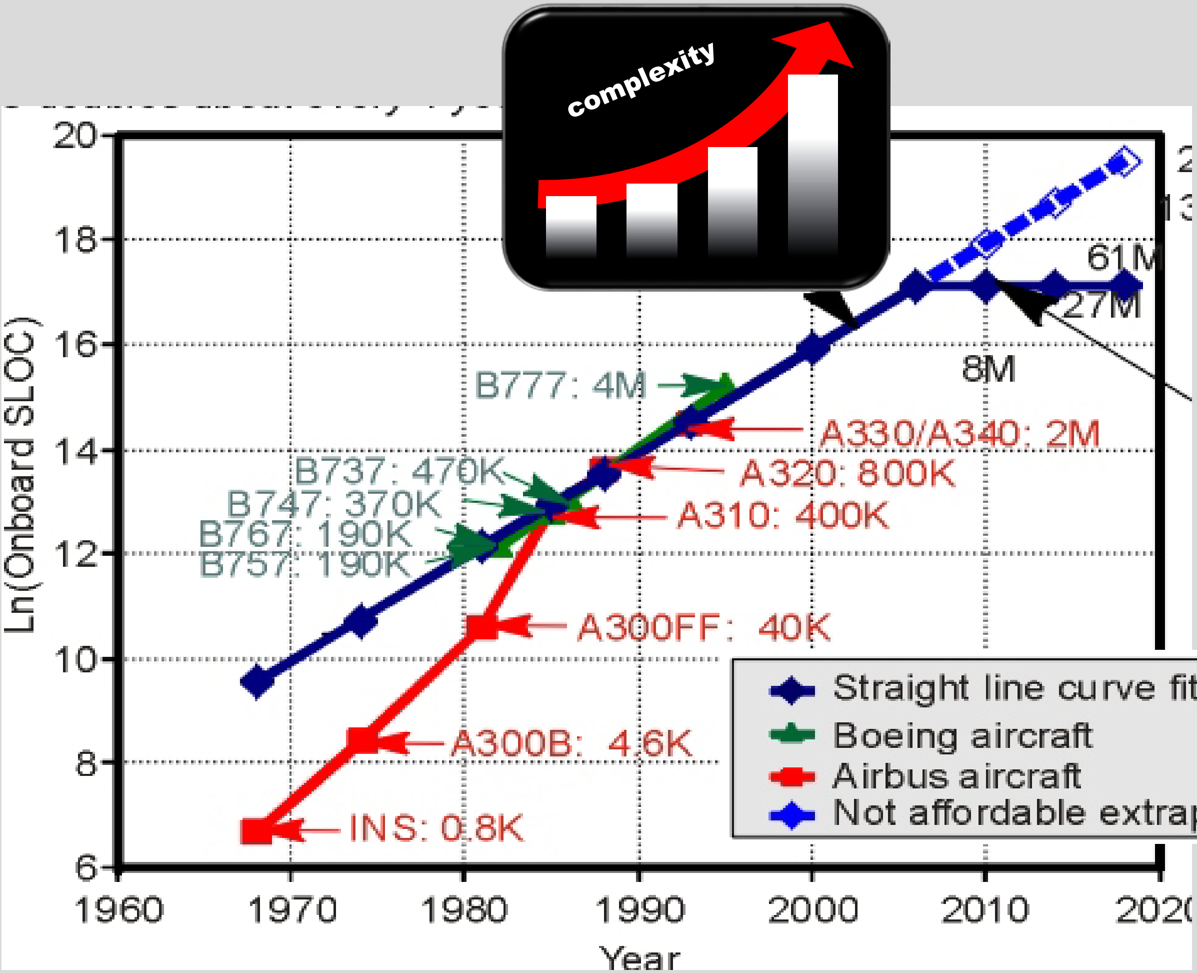
Critical Autonomous Systems Have Many of the Same Problems as Other Complex Systems*

State-Space Explosion

Unpredictable Environments

Emergent Behavior

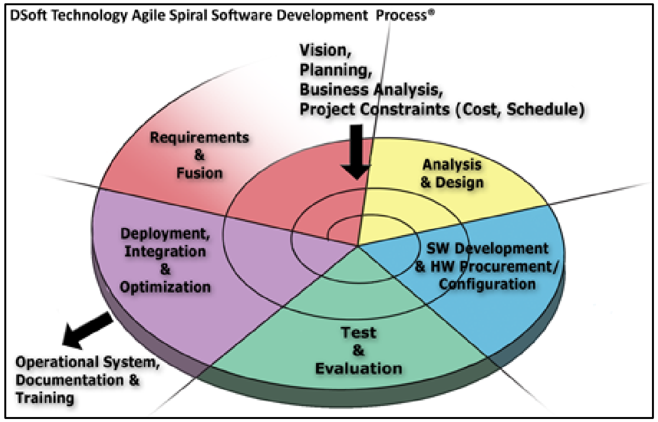
Human-Machine Communication



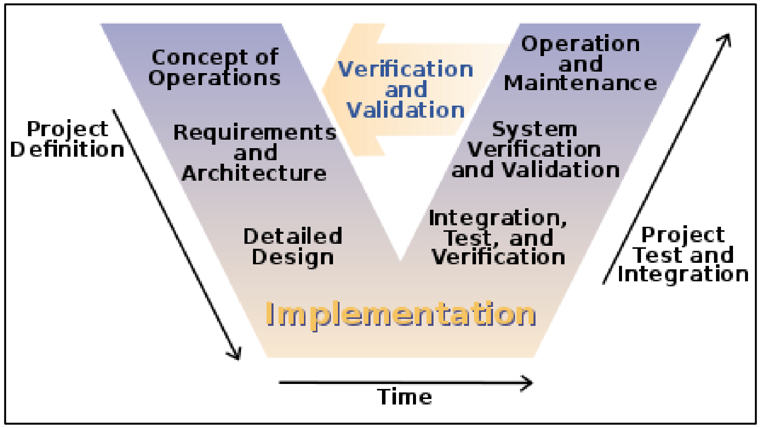
Highly complex systems and increasing autonomous systems face many of the same Verification and Validation challenges

*Kerianne Gross, AFRL, 11th Annual V&V Summit, 2016

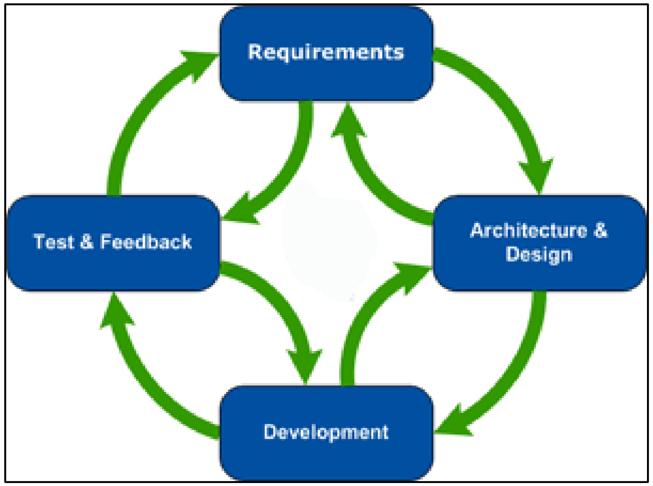
Many Available Software Development Processes


DSoft Technology Agile Spiral Software Development Process®
Vision, Planning, Business Analysis, Project Constraints (Cost, Schedule)
Requirements & Fusion
Analysis & Design
Deployment, Integration & Optimization
SW Development & HW Procurement/ Configuration
Test & Evaluation
Operational System, Documentation & Training

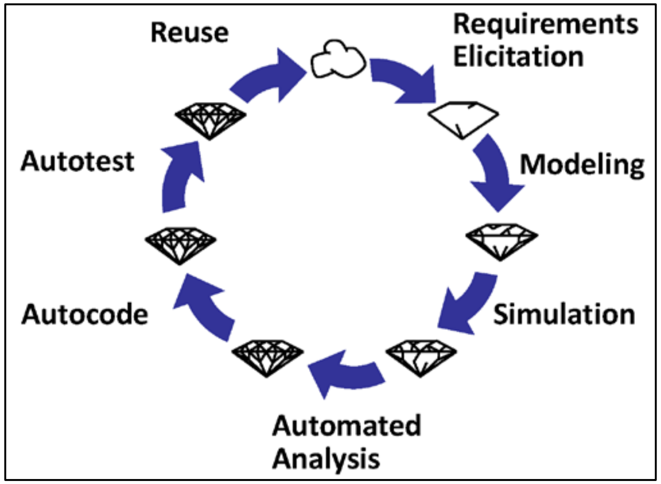
Spiral Model


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

V Model



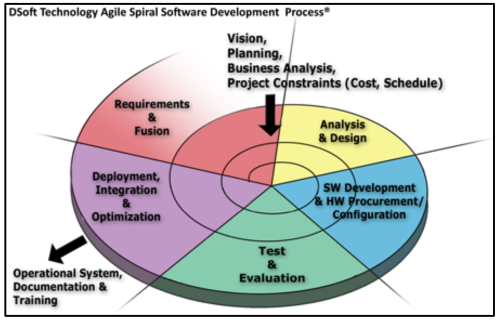
Agile Model



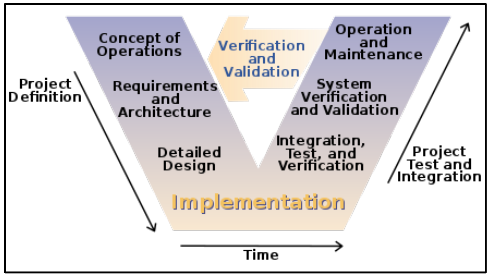
Model-Based Development

Developing Critical Autonomous Systems: Principles and Guidelines

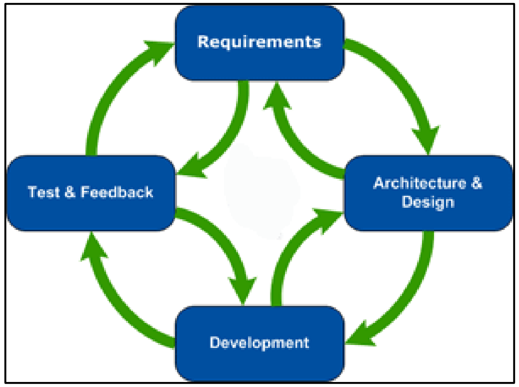
Spiral Model


DSoft Technology Agile Spiral Software Development Process®
Vision, Planning, Business Analysis, Project Constraints (Cost, Schedule)
Requirements & Fusion
Analysis & Design
Deployment, Integration & Optimization
SW Development & HW Procurement/ Configuration
Test & Evaluation
Operational System, Documentation & Training

V Model


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

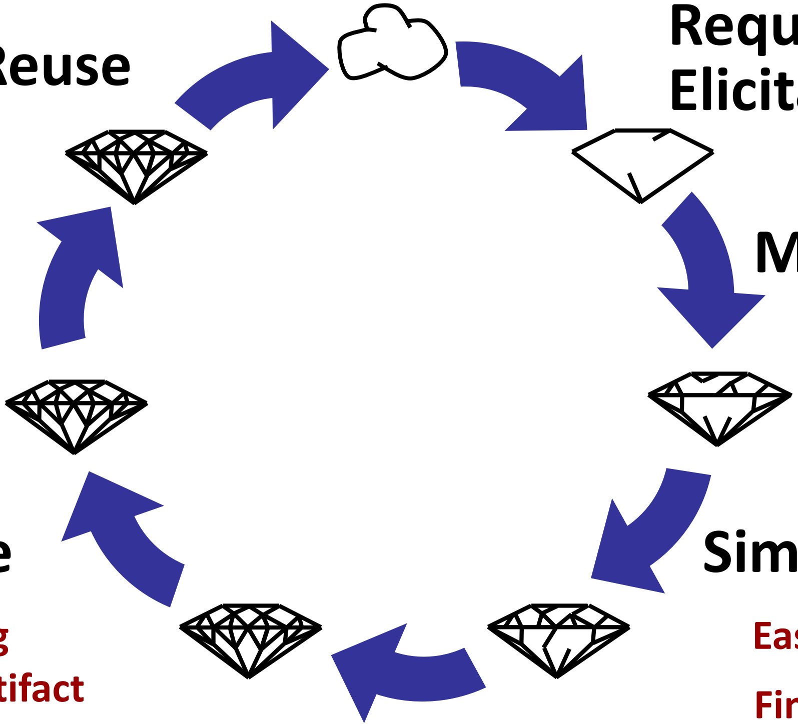
Agile Model



- Incremental/Iterative Development
- Formal verification can find many errors quickly in very large systems
- Need verification and validation throughout the development process
- Testing and modeling & simulation are still important and will remain crucial

What Overall Process To Use in Developing Autonomous Systems?

Model-Based Development



Requirements Elicitation

Modeling

Clear Specifications

Improves Communication

Simulation

Easy Validation

Finds Errors Early

Automated Analysis

Finds the Really Hard Errors
Cheaper Than Manual Analysis

Autocode

Eliminates Manual Coding
Makes Model Primary Artifact

Autotest

Reduces Cost of Testing

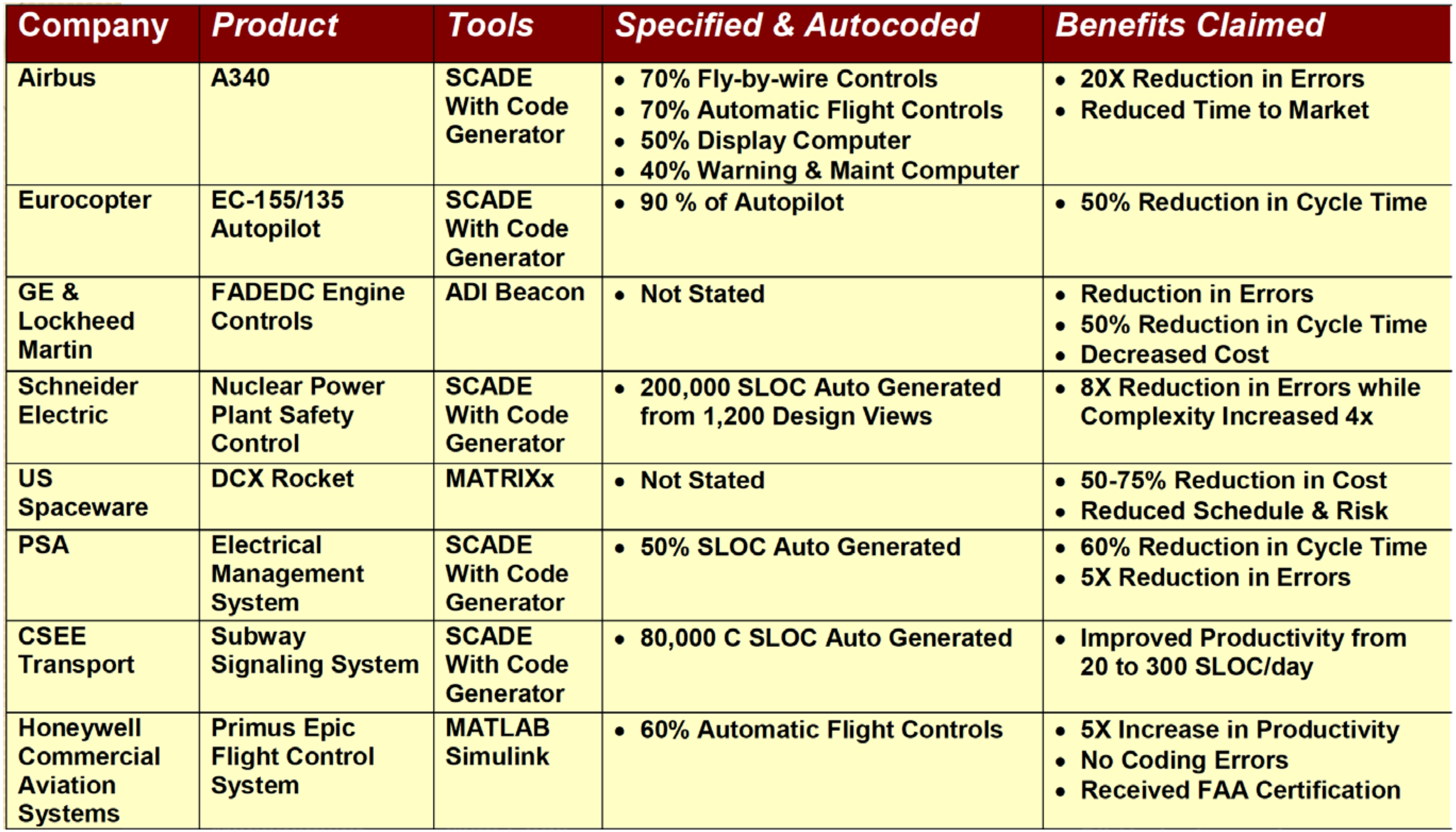
Enables More Testing

Reuse

2/18/2019

Adapted from Steve Miller, Univ. of Minn., 2013

Model-Based Development Examples*



Company	Product	Tools	Specified & Autocoded	Benefits Claimed
Airbus	A340	SCADE With Code Generator	• 70% Fly-by-wire Controls • 70% Automatic Flight Controls • 50% Display Computer • 40% Warning & Maint Computer	• 20X Reduction in Errors • Reduced Time to Market
Eurocopter	EC-155/135 Autopilot	SCADE With Code Generator	• 90 % of Autopilot	• 50% Reduction in Cycle Time
GE & Lockheed Martin	FADEDC Engine Controls	ADI Beacon	• Not Stated	• Reduction in Errors • 50% Reduction in Cycle Time • Decreased Cost
Schneider Electric	Nuclear Power Plant Safety Control	SCADE With Code Generator	• 200,000 SLOC Auto Generated from 1,200 Design Views	• 8X Reduction in Errors while Complexity Increased 4x
US Spaceware	DCX Rocket	MATRIXx	• Not Stated	• 50-75% Reduction in Cost • Reduced Schedule & Risk
PSA	Electrical Management System	SCADE With Code Generator	• 50% SLOC Auto Generated	• 60% Reduction in Cycle Time • 5X Reduction in Errors
CSEE Transport	Subway Signaling System	SCADE With Code Generator	• 80,000 C SLOC Auto Generated	• Improved Productivity from 20 to 300 SLOC/day
Honeywell Commercial Aviation Systems	Primus Epic Flight Control System	MATLAB Simulink	• 60% Automatic Flight Controls	• 5X Increase in Productivity • No Coding Errors • Received FAA Certification

* Steve Miller, “Proving the Shalls” 2006.

Does Model-Based Development Scale?

Airbus A380

Length	239 ft 6 in
Wingspan	261 ft 10 in
Max Takeoff Weight	1,235,000 lbs
Passengers	Up to 840
Range	9,383 miles

Components Developed Using MBD

- Flight Control
- Auto Pilot
- Fight Warning
- Cockpit Display
- Fuel Management
- Landing Gear
- Braking
- Steering
- Anti-Icing
- Electrical Load Management

* Steve Miller, “Proving the Shalls” 2006.

Once a process is selected, we can focus on methods and tools to support the process

Candidate Tools*

- Esterel Studio and SCADE Studio from Esterel Technologies
- Rhapsody from I-Logix
- Simulink and Stateflow from Mathworks Inc.
- Rose Real-Time from Rational

*Mike Whelan, “Why we model: Using MBD Effectively in Critical Domains,” ICSE Tutorial, 2013.

What Are Unsolved Challenges for Autonomous Systems That Remain Challenges for Other Critical Systems?

- Requirements Acquisition: How to obtain an understanding of the system requirements sufficient to create a formal requirements model of the behavior of a system or system component?
- Composition: How to soundly compose different formal models of the components of the system?
- Formal V&V of the Implementation: How to use the verified, validated formal system model to obtain assurance of the system implementation?

Major Barrier in Applying Formal Methods: Obtaining a Formal Model of the Required System Behavior (2)

Use formal methods to support certification by NSA of a security-critical, software-based communications system*

Requirements Acquisition	years…
Formulate the TLS & the Data Separation Property	2.5+ weeks
Translate the TLS to PVS/TAME and Construct the Proofs	3+ weeks
Demonstrate Code Conformance	5+ weeks
Annotate the Code	many months…

*C Heitmeyer+, *IEEE Trans. on Software Eng.*, 2008.

What Are Challenges Unique to Autonomous Systems?

- Human-Autonomy Interaction
  - How to design the human-machine interaction with assurance that the combination of the human and machine will successfully perform the assigned mission
  - How to obtain human trust of the autonomy/How to avoid human overtrust of the autonomy
- Machine-Learning and Other AI Techniques: How to capture and reason about the behavior of non-standard components of the system, e.g., components that use machine learning or other AI techniques?

Role of Unmanned and Autonomous Systems in Future DoD Missions

DEPARTMENT OF DEFENSE
DEFENSE SCIENCE BOARD

July 2012

OFFICE OF THE UNDER SECRETARY OF DEFENSE FOR ACQUISITION, TECHNOLOGY AND LOGISTICS
WASHINGTON, D.C. 20301-3140

Defense Science Board Report, "The Role of Autonomy in DoD Systems," July 2012

- Unmanned and autonomous systems will have a major impact on warfare world wide
  - Rather than replace humans, they will extend/complement human capability
  - Their design/operation needs to be considered in terms of *human-machine collaboration* (HMC)
- **Major problems**
  - HMC is frequently handicapped by **poor design**
  - For commanders and operators, there is a ***lack of trust*** that system's autonomous functions will operate as intended

Issue: Human Mistrust of Autonomy

- Two major notions of trust*
  - **System Trust**: Human confidence that the system will behave as intended
  - **Operational Trust**: Human confidence that the system will help him/her perform the assigned tasks
- To achieve system trust
  - Need **high assurance** that the system satisfies its requirements -> **formal modeling, formal verification, …**
- To achieve operational trust
  - Need well-designed HCI *and* **human validation** that the designed autonomy will help operator accomplish the mission -> human factors literature, modeling/simulation

Both formal methods and human factors guidelines and principles can help overcome human mistrust of autonomy

*Terwilliger, Raytheon, 2014.

Summary

- What challenges have already been addressed?
  - The processes, methods and tools to use in obtaining assurance of autonomous systems
- What challenges remain unsolved for autonomous systems and other critical systems?
  - Requirements Acquisition
  - Composition
  - Formal V&V of System Implementation
- What challenges are unique for autonomous systems?
  - Design of Human-Autonomy Interaction
  - How to deal with machine learning and other AI techniques

Concluding Remarks

- Need demonstrations of real-world experience
- What are key elements for assured autonomy
  1. Formal model of the required system behavior
     - Nominal behavior
     - Identification of faults and how to recover from each fault (e.g., how to mitigate or eliminate the fault)
  2. Precise statement of the required system properties (e.g., functional correctness, safety, security, timing)
  3. Proof that the model satisfies the properties and validation by domain experts that the model captures the intended behavior
  4. Explicit assumptions about the behavior of the system environment and validation of those assumptions
  5. Both formal and less formal evidence (e.g., testing, modeling and simulation) that the system implementation satisfies the system requirements