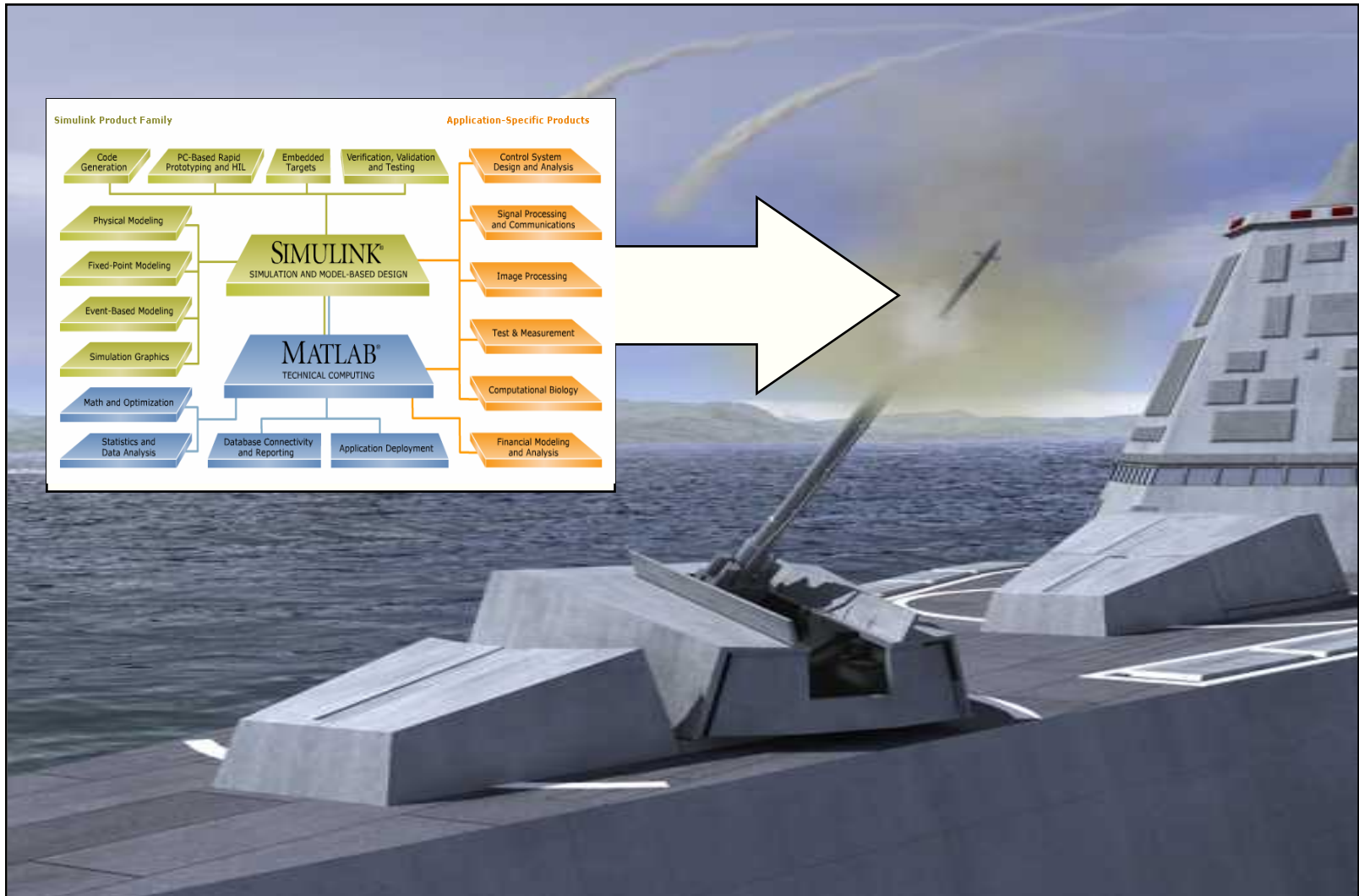


A User's Experience with Simulink® and Stateflow® for Real-Time Embedded Applications



James E. Craft and Bob Rusk, Lockheed-Martin Missiles and Fire Control

Lockheed Martin Corporation

- 140,001 Employees
- 65,000 Scientists and Engineers
- 23,000 IT Professionals, Systems and Software Engineers

LMC writes more code than Microsoft

- My Experience:
 - ✓ MSSE, Software Engineer for 25 years
 - ✓ Lean Six Sigma Blackbelt
 - ✓ C++ and UML Instructor (UML Subject Matter Expert)
 - ✓ Software Developer, Software Development Lead
 - ✓ Software Architecture, CMMI Maturity
- Project Experience:
 - ✓ Comanche
 - ✓ Sniper/ATP
 - ✓ AGS LRLAP
 - ✓ MRM

Why Do We Model ?

➤ *The short answer – to avoid spectacular failures !*

- ✓ *Swedish Naval Warship, Vasa (1625)*
- ✓ *NASA Mars Climate Orbiter*
- ✓ *Denver airport baggage handling system*
- ✓ *FBI's Virtual Case File system*
- ✓ *Talking Barbie*



➤ *Modeling gives us a blueprint of the system before we build it*

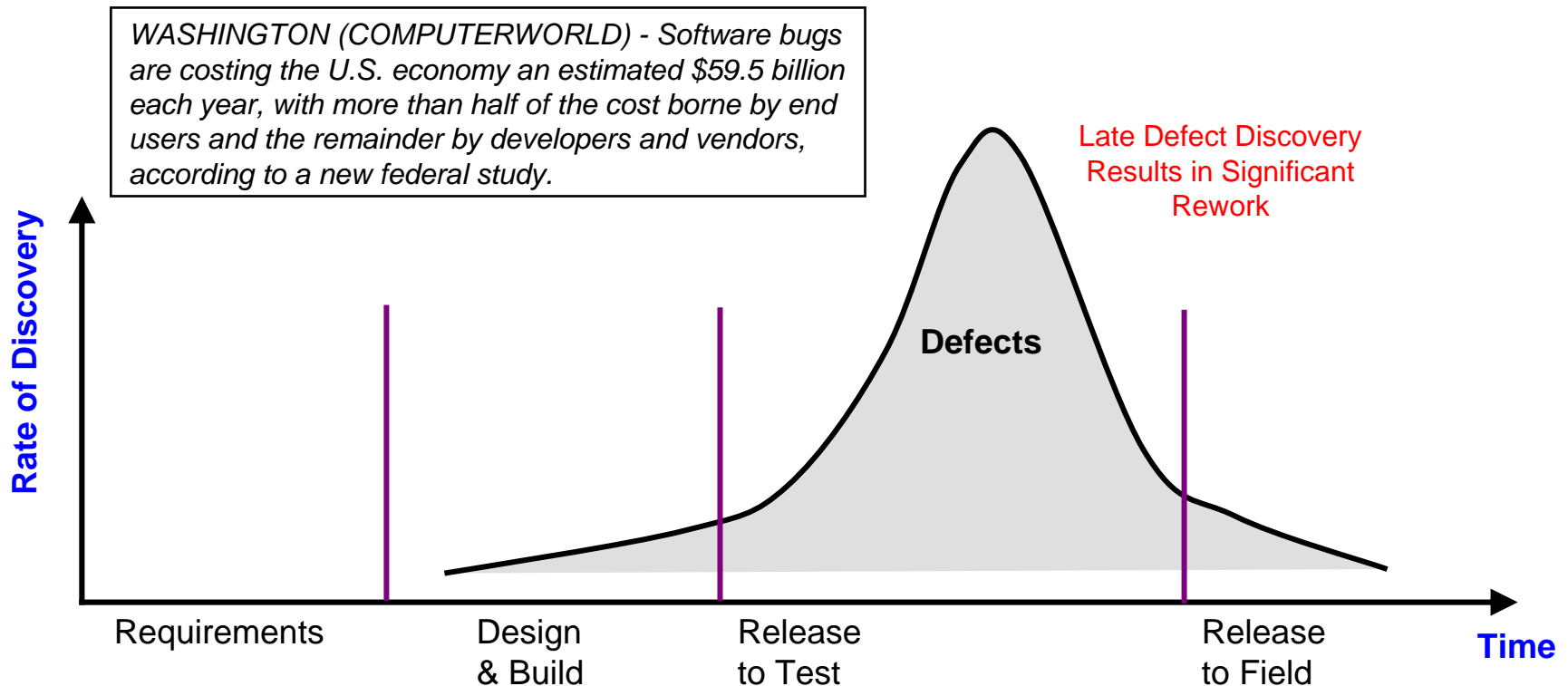
- ✓ *Sketch*
- ✓ *Blueprint*
- ✓ *Executable Design*



➤ *Modeling provides a shared understanding between the customer, the SME, the Systems Engineer, the Software developer, and the tester*

Late Defect Identification is Costly

- Traditional testing approaches result in defect discovery late in the process
- Relatively little improvement over past 20 years*



100X Increase in Cost of Removing Defects

*Source: Boehm, Barry. *Software Engineering Economics*. Edgewood Cliffs, NJ: Prentice-Hall, Inc., 1981
Boehm, Basili, "Software Management." *IEEE Computer*, January 2001.

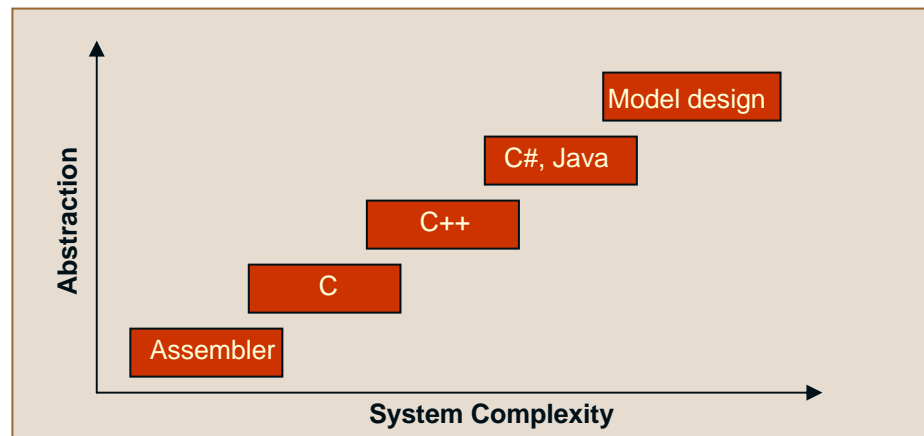
Using Model-Based Design in Embedded Real Time Systems

➤ What is Model-Based Design ?

MBD is an approach to software development where extensive models are created before source code is written.

➤ Using MBD to address the new “Software Challenge”

- ✓ *More software*
- ✓ *Mounting complexity of software*
- ✓ *Decreasing number of electronic components*
- ✓ *Cost, cost, cost*



Fitting MBD into Lockheed's Product Development Process

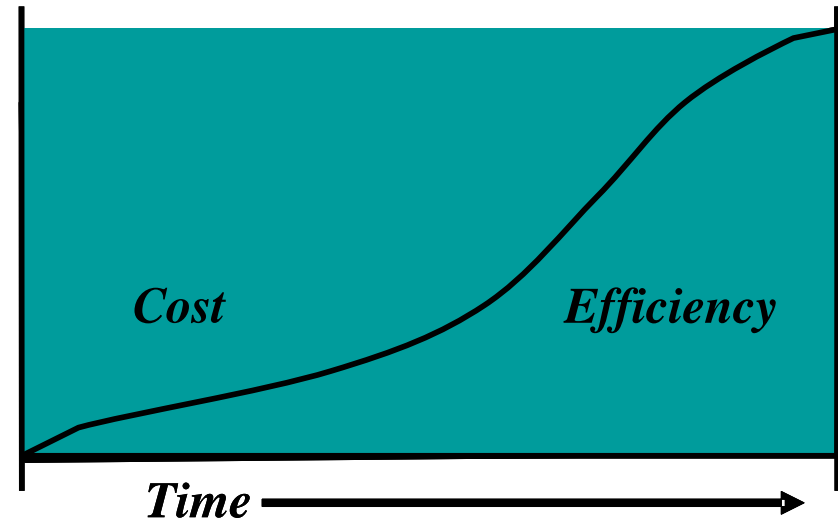
- *Requirements*
 - ✓ *Creating the correct system*
 - ✓ *Use Case analysis*
 - ✓ *Data stores, msg transfers, component interfaces*
- *Test*
 - ✓ *Model checking, test coverage*
 - ✓ *Allows validation of requirements without significant investment in implementation*
- *Peer Reviews*
- *Lean Development*
 - ✓ *Separate computational code and behavioral code*
- *Agile Principles (Agile Modeling)*
 - ✓ *Iterative modeling (build a little, test a little...)*
- *Working within CMMI® Level 5 Environment*
 - ✓ *Code reliability, optimization*
 - ✓ *Component based software*
 - ✓ *Code analysis, Metrics*



Cost Implications for Model-Based Design

➤ Upfront Costs of Model-Based Design

- ✓ *Tool/license costs*
- ✓ *Learning Curve*
 - *Formal training*
 - *On-the-job experience*
- ✓ *Construction of models*



➤ Efficiencies Gained with Model-Based Design

- ✓ *Eliminates human translation of design into software*
- ✓ *Peer reviews focus on consistency between documented design and implemented software*
- ✓ *Engineering effort focuses on correctly defining, designing, and testing software*

Upfront Costs Pay Off With Increased Efficiency

Special Challenges for Embedded Real Time Applications

➤ Optimization strategies

- ✓ *Mapping to Target Processors*
- ✓ *Timing*
- ✓ *Throughput*

➤ Cost and Reliability

- ✓ *Quality*
- ✓ *Mission success*

➤ Software Safety (DO-178B)

- ✓ *Safety Assessment Process*
- ✓ *Hazard Analysis*
- ✓ *Examines the effects of a failure condition in the system*

➤ Software Life Cycles

- ✓ *Waterfall, Spiral, Agile*

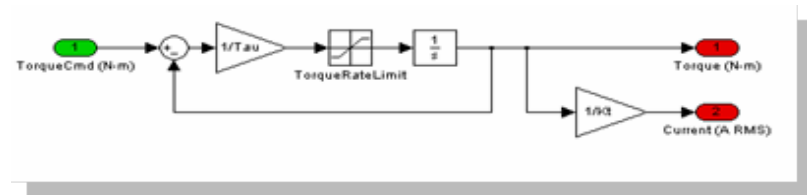
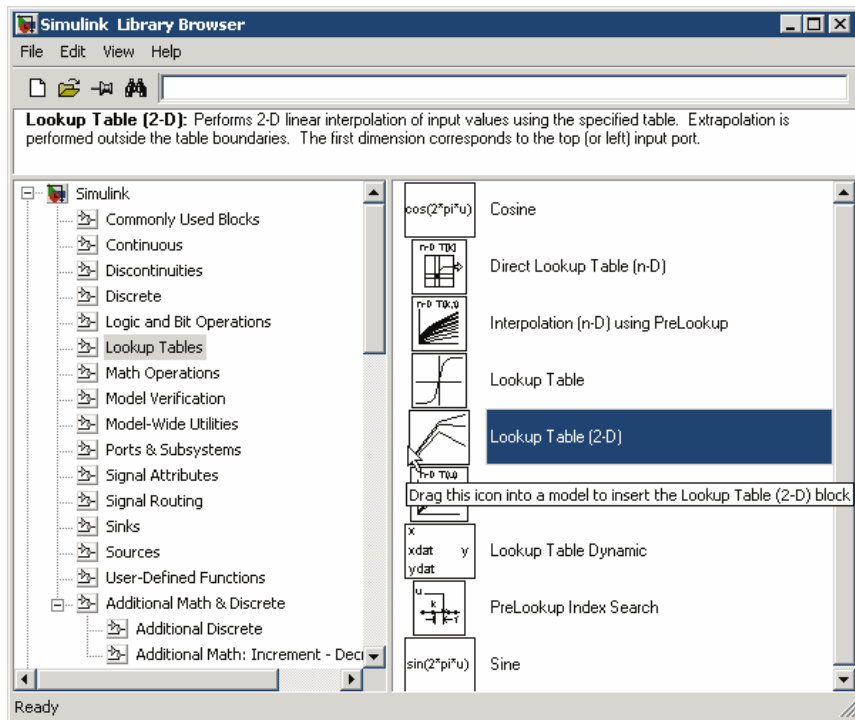
Level	Failure condition	Objectives	With independence
A	Catastrophic	66	25
B	Hazardous	65	14
C	Major	57	2
D	Minor	28	2
E	No effect	0	0

Software Failure Levels

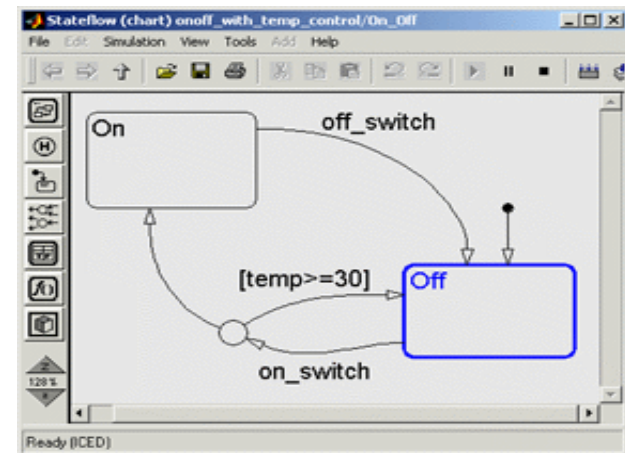
MATLAB®, Simulink® and Stateflow® Models

Meeting the Challenge

MATLAB® and Simulink® form the core environment for Model-based Design for creating accurate, mathematical models of physical system behavior.



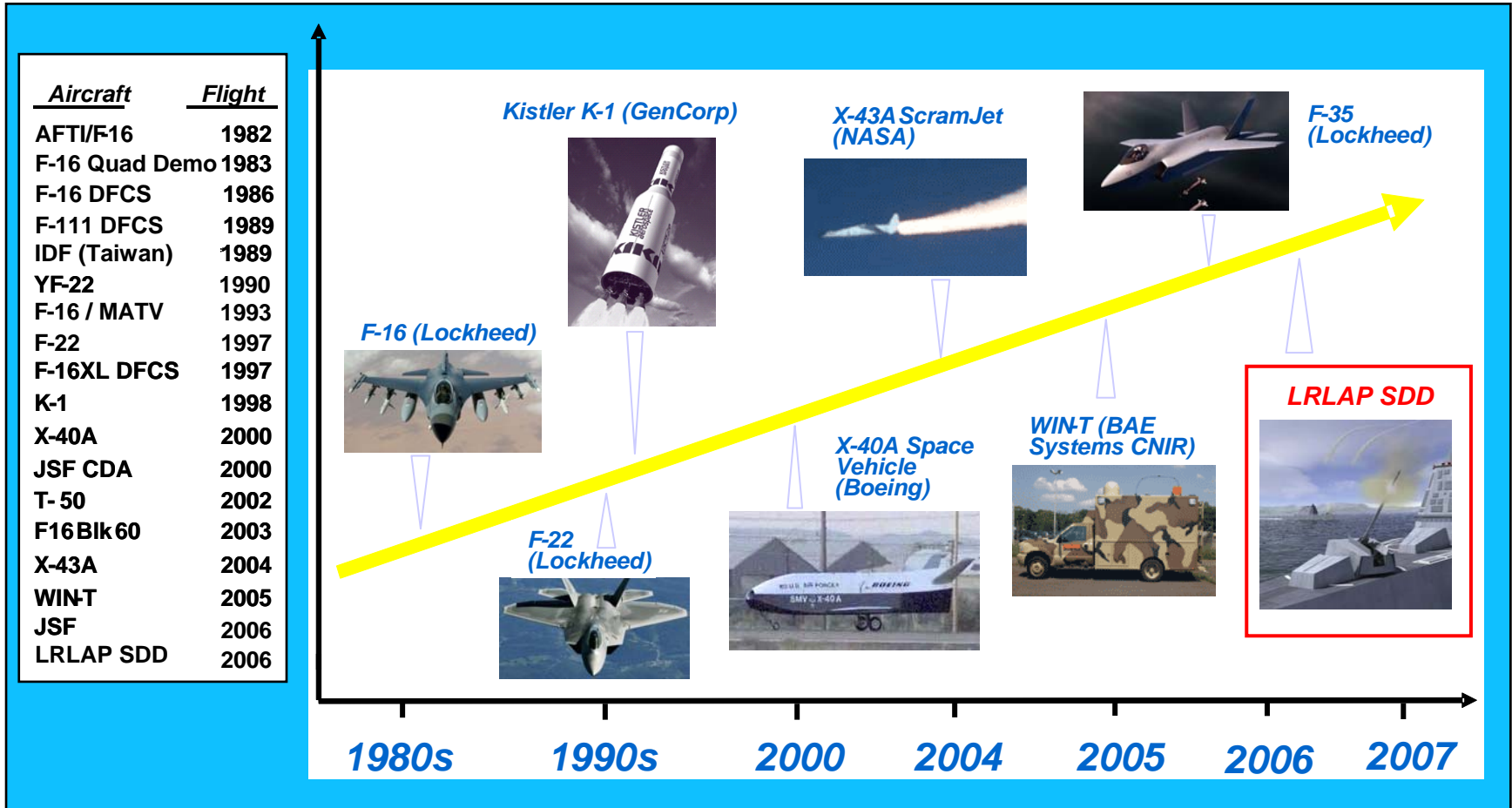
Simulink® for mathematic/control processes



Stateflow® for logical processes

Graphical Software Building Blocks

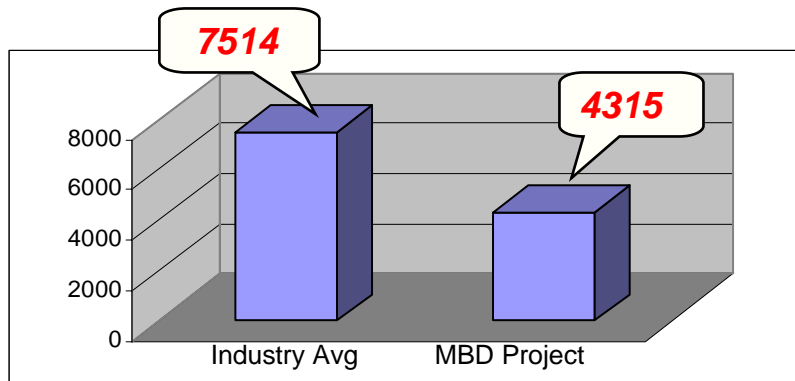
Use of MBD in Defense and Aerospace Applications



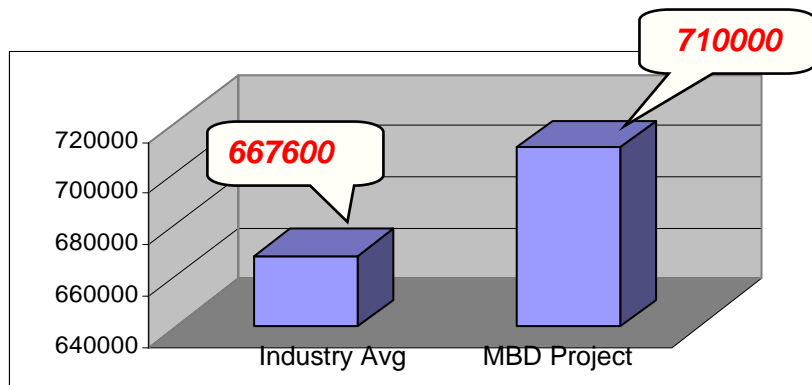
LRLAP builds on heritage of Simulink in Aerospace and Flight Systems ...

Industry Usage of MBD

Developers that use MBD in their designs are able to manage (year over year) more design starts and completions than the industry average. This translates into higher productivity and greater savings for the organization.

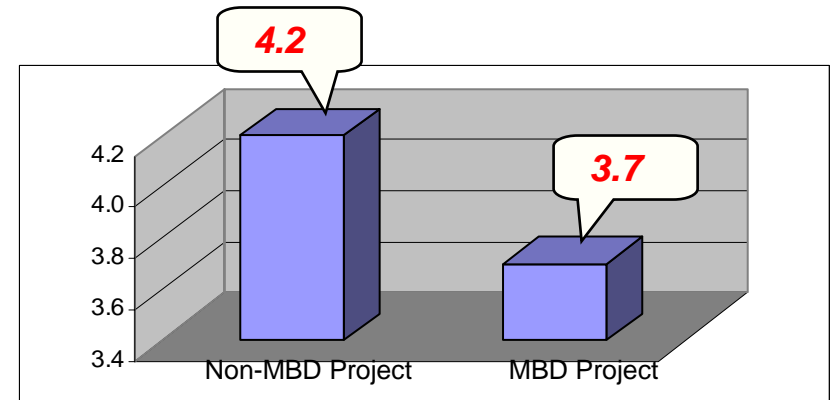


Avg Lines of Handwritten Code Per Developer



Total Lines of Code Per Project

Recent surveys comparing coding efficiencies and schedule impacts for MBD programs show improved performance factors.



Avg Months Behind Schedule

Source: *What Do You Do When the Horse You Are Riding Drops Dead?*, Jerry Krasner, Embedded Market Forecasters, March 2007

Process Examples of Simulink® Model Development for Automatic Code Generation

➤ Project A

- ✓ *Develops models using well defined Simulink Coding standards*
- ✓ *The Model is the source and the generated “C” is treated like object code*

➤ Project B

- ✓ *Core Simulink Building Blocks are validated and all models are required to use only the validated building blocks*
- ✓ *The Model is the source and the generated “C” is treated like object code*

➤ Project C

- ✓ *Develops models using well defined Simulink Coding standards*
- ✓ *Using Mathworks 178B guidelines from Bill Potter*
- ✓ *Uses scripts and configuration files to enhance readability of generated “C” source*
- ✓ *The Model is the detailed design and the “C” code is the source*
- ✓ *Follows a more standard software development process*

Long Range Land Attack Projectile

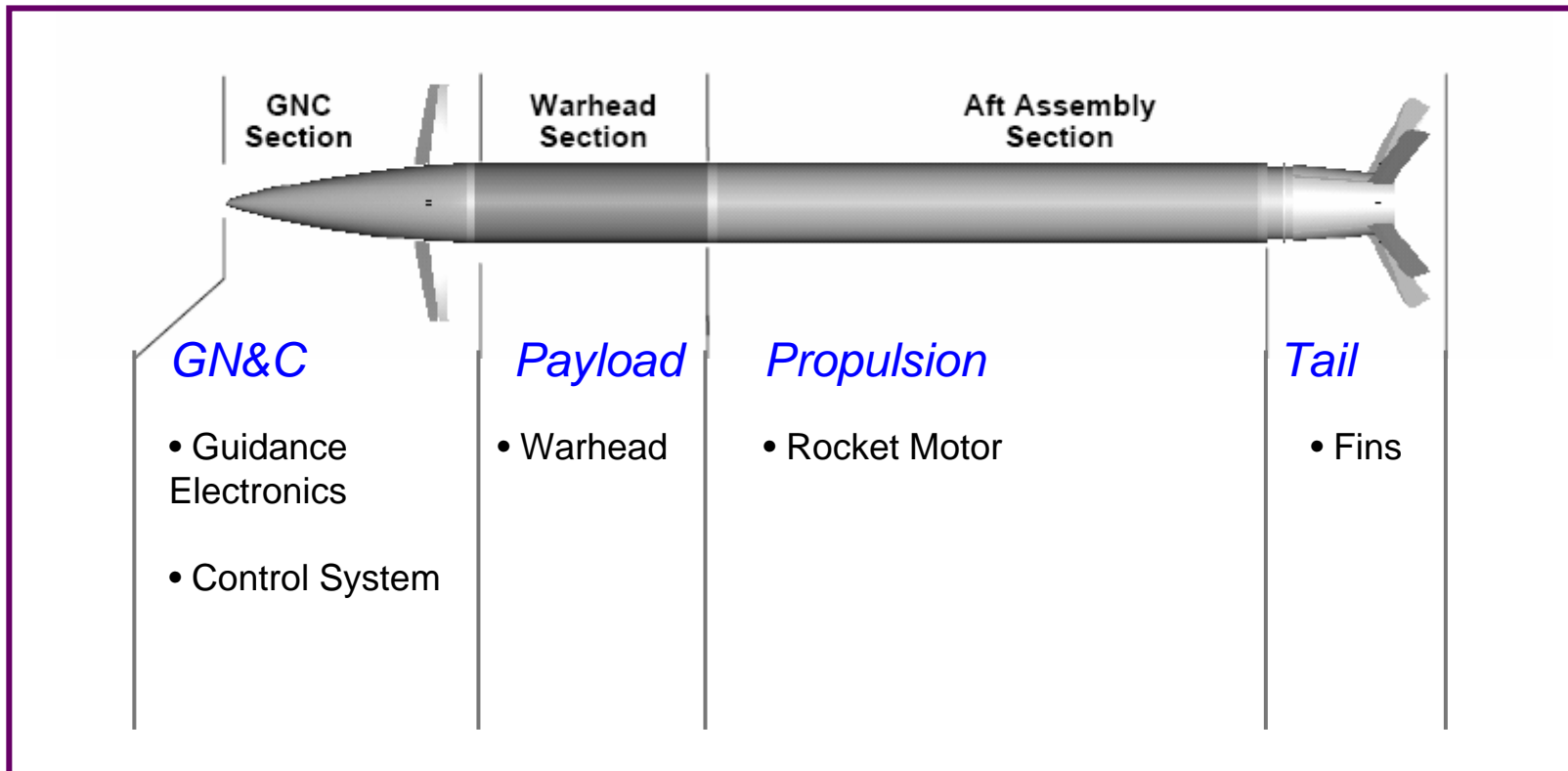


SAIC is subcontracted to Lockheed-Martin for the GNC subsystem

- LRLAP is part of a family of 155mm projectiles for the Advanced Gun Systems on the U.S. Navy's next-generation DDG-1000 destroyer
- ✓ *Provides single-strike lethality from offshore against a wide range of targets*
- ✓ *Multiple payloads and multiple guidance approaches*
- ✓ *Initial concept focused on long-range land attack requirement*

Tactical Design Overview

LRLAP gives DDG-1000 warships the ability to provide interdiction, suppression and other fire support missions to support ground and expeditionary forces.




[Video - Navy Advanced Gun System \(AGS\) Non-combatant Evacuation Simulation Scenario](#)

GNC Applications

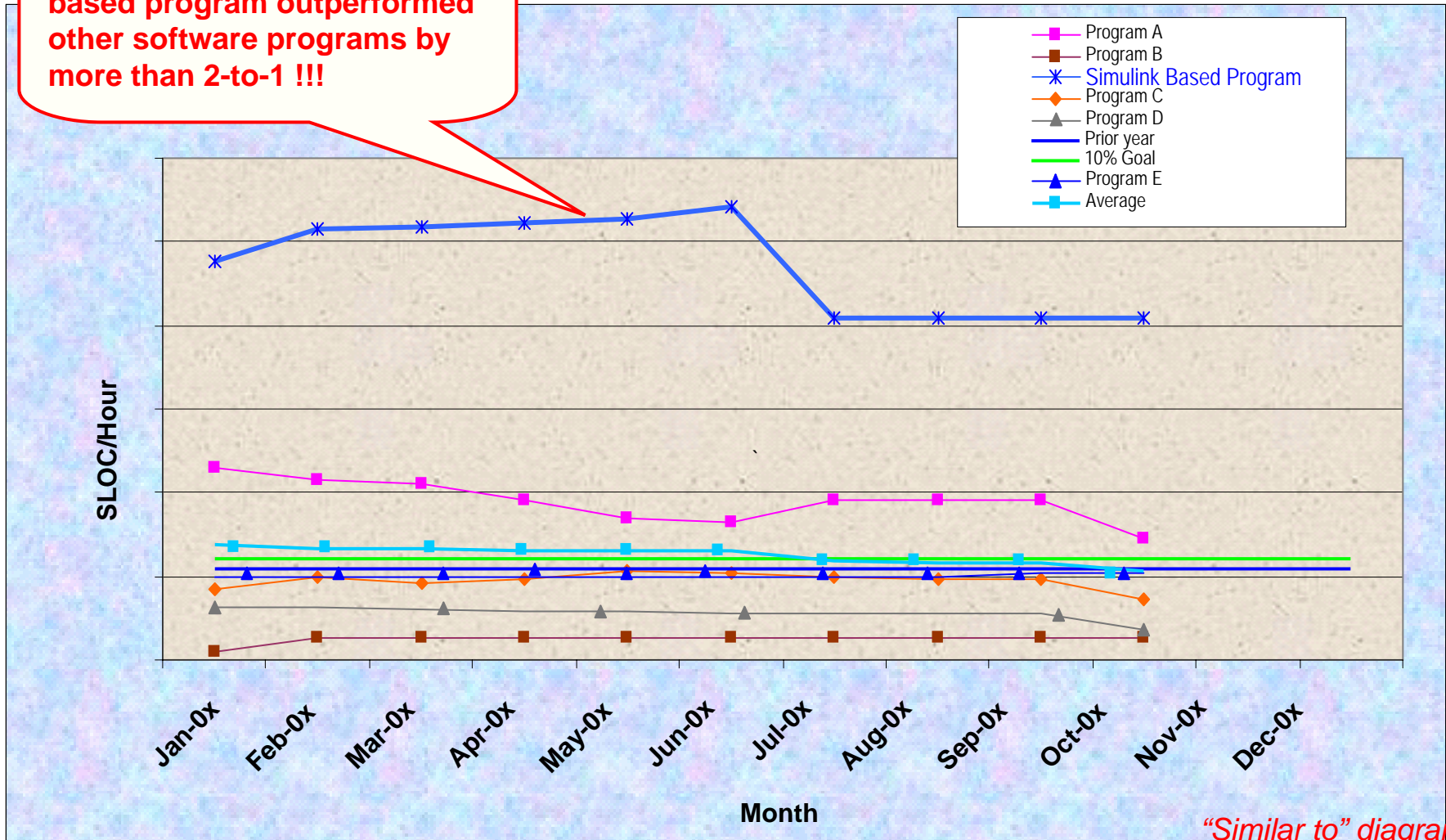
- *Guidance, Navigation and Control applications are prime candidates for Simulink modeling and simulation*



IMU Subsystem	Provides pitch, roll and yaw rates
GPS Subsystem	Detects current position based on GNSS constellation
Autopilot	Provides automated vehicle guidance and control
Navigation Algorithms	Plans and records position compared to known locations
Guidance Laws 	Evaluates sensor readings and course data to determine speed and heading
Control Subsystem	Flight control surfaces used to stabilize and direct the vehicle
Wind Models	Provides mach speed and dynamic pressure

Software Productivity Using Simulink®

With Simulink®, the model-based program outperformed other software programs by more than 2-to-1 !!!



"Similar to" diagram

Overall LRLAP Experience with MBD

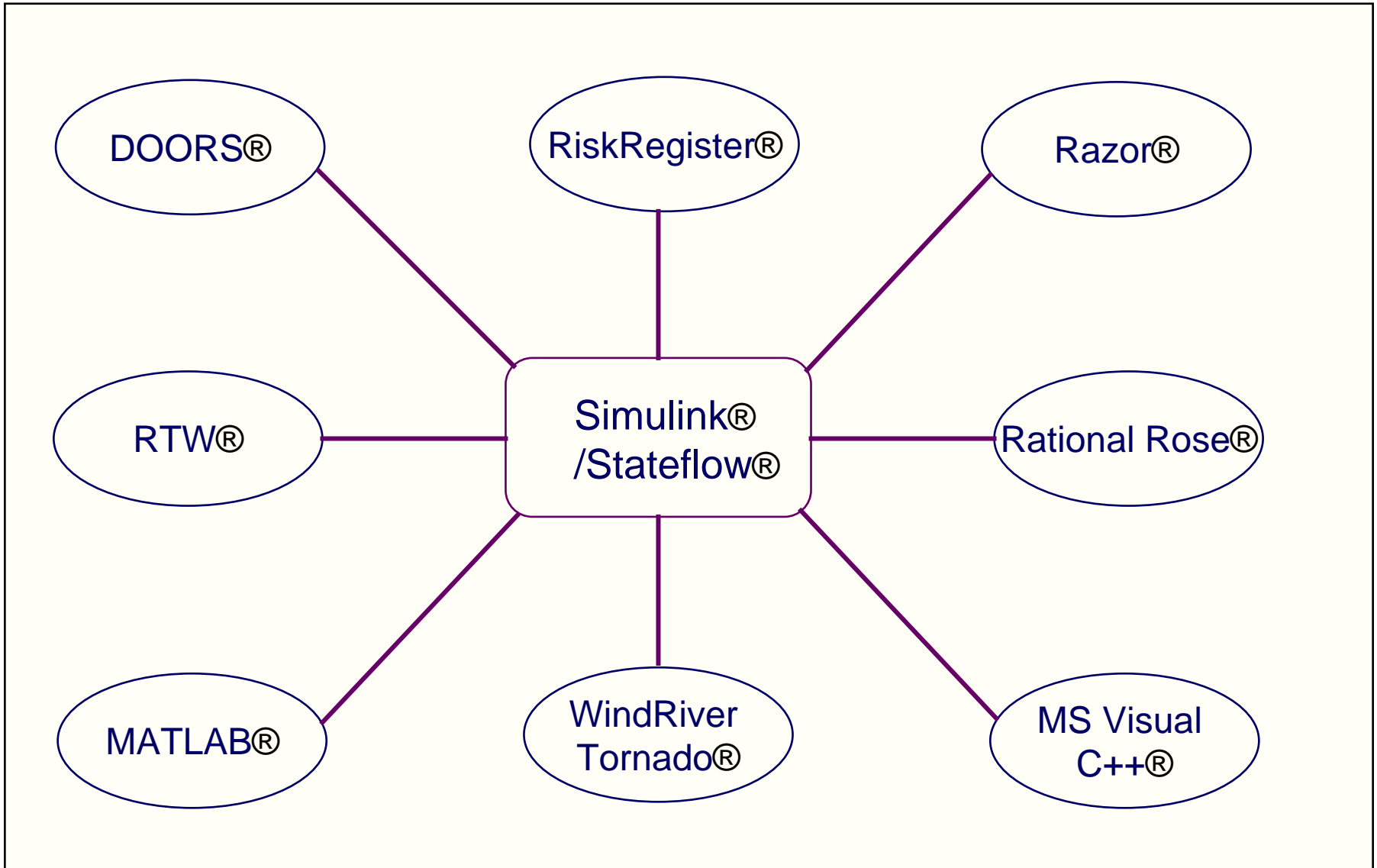
➤ Successes

- ✓ *LRLAP is the longest-range guided projectile in U.S. history*
- ✓ *Nine Successful Flight Tests with No Software Errors*
- ✓ *Cost & Safety: Reduced Software Defects (Early Checkout in Engineering Simulations)*
- ✓ *Verification: Rapid Prototyping to Analytical and Real-Time Simulators*
- ✓ *Verification: Reduced Testing (Unit Test and Standalone)*
- ✓ *Cost: Overall Reduction in Manhours/SLOC*

➤ Challenges

- ✓ *Process: Handcode to Auto Generated Code Integration*
- ✓ *Tool: Interface Control and Management on Large-Scale Model*
- ✓ *Resources: Auto Code Efficiency (Memory, Throughput)*
- ✓ *Training: Turning Control Law Designers Into Software Engineers*
- ✓ *Optimization of automatic code generation (readability, standards, CM procedures)*

Tools That We Use



Decision Making for Auto Code Generation

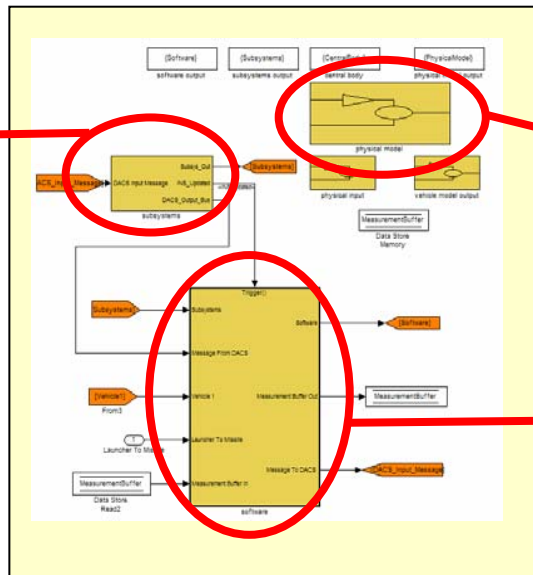
Category	Examples	Method/Tool
Algorithms	Guidance, Navigation, and Control algorithms, math models	Simulink® auto code generation
Structural	Capsules, Classes, Interfaces, Protocols	Rose®
Operating Systems	VxWorks®, Nucleus®, etc	COTS or hand-coded
Device Drivers	SDLC drivers, FireWire drivers, etc	COTS or hand-coded
Internal Logic	Data movement, arithmetic operations, etc	Hand-coded
Message Data	Assembly of message, calculation of message data, etc	Hand-coded

- Challenges for autocoding certain categories of software
 - ✓ *If a tool lacks the features to adequately model software category*
 - ✓ *Sometimes less efficient to model versus hand-coding*
 - ✓ *If modeling causes negative side effects on system simulation run time*
 - ✓ *If resultant auto generated software executes inefficiently*

Simulink® and Auto Generated Code

Auto Generated Code

Simulink®



Continuous
Physics
Models

GNC
Software
Models

```
/* Stateflow Block: '<S1>/GNC Executive' */  
{  
  
/* Stateflow Block: '<S1>/GNC Executive' */  
  
if (software5_DWork.GNCExecutive.is_active_c19_software5 == 0) {  
    software5_DWork.GNCExecutive.is_active_c19_software5 = 1;  
    software5_DWork.GNCExecutive.is_c19_software5 =  
        (uint8_T)software5_IN_prelaunch;  
} else {  
    switch (software5_DWork.GNCExecutive.is_c19_software5) {  
        case software5_IN_postlaunch:  
            if ((int32_T)software5_B.Memory[0] > 0) {  
  
                software5_reset_tracker();  
  
                software5_clear_reset();  
            }  
  
            software5_save_inputs();  
  
            software5_ownership();  
  
            software5_tracker();  
  
            software5_MKV_guidance();  
  
            software5_autopilot();  
  
            software5_output();  
  
            break;  
        case software5_IN_prelaunch:  
            if (software5_U.signal1 != 0.0) {  
                software5_DWork.GNCExecutive.is_c19_software5 =  
                    (uint8_T)software5_IN_postlaunch;  
            }  
    }  
}
```

➤ Simulation

✓ Tool System performance evaluation

✓ Requirements definition support

➤ Automatic code generation

✓ Model blocks translated to comparable code constructs

✓ Embedded software & real-time simulation software can be generated

Effective Simulink® Usage Produces Software From Model-based Design

Simulink® Auto Code Quality

- LMMFC Simulink® Modeling Style Guide ensures readable, maintainable software is generated
- Characteristics of Simulink® generated software
 - ✓ *Generated software structurally matches Simulink® model*
 - ✓ *Comment-to-Lines of Code ratio is developer controlled*
 - Simulink® comment blocks
 - Comment fields with model blocks
 - ✓ *Developers can control variable names*
 - Unique model block names
 - Unique model block input/output port name
- Well-styled Simulink® models become part of the Software Design Document (SDD) and Algorithm Description Document (ADD)

Summary

- Significant Reduction in Software Anomalies Through Early Prototyping and Evaluation
- Significant Reductions in Manhours/Source Lines of Code with Model-Based Software and Automatic Code Generation
- Produced Excellent Flight Test Results in Very Complex Development Effort with NO Compromises to Flight Safety
- More Requirements-Focused Development Process
- Leveraging Off Heritage Relationships with Mathworks to Mature Modeling Environment and Code Generation
- CMMI Process More Ingrained into Graphical Model Development and Review

Future Plans

➤ Code Readability

- ✓ *Configuration Parameters*
- ✓ *Auto coding Standards*
- ✓ *Templates (Code Headers)*

➤ Software Configuration Management

- ✓ *Managing Release Issues*
- ✓ *Model Revisions (Check-in, Check-out)*

➤ Design Issues

- ✓ *Linking Simulink® with Rose®*
- ✓ *SysML, DODAF*
- ✓ *System of Systems*