

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY)		2. REPORT TYPE		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)



Digital Engineering Shift Left (MBSE and DevSecOps)

Allan Dianic OUSD(R&E)
Supported by
Shane Smith Mitre
Dr Bruce Douglass Mitre

Lets Get Digital Webinar

22 June 2023



Purpose

- Shift Left: Provide the Software Community best practices and guidance to support the implementation of Digital Engineering (Model Based System Engineering) into the DevSecOps pipelines. This approach seeks to shift engineering left in the software domain by describing more advanced methods of planning and modelling large integrated software products concurrent with code and system integration and operation takes place.



What we have learned

- While there has been some Academic research in this subject but limited practice within the DOD
 - There is a lot of MBSE work in the DOD at the system level just not much in Agile and DevSecOps or Software intensive space
- There is some work in the commercial space in combining Models/MBSE with software development
 - IBM uses the MBSE tool Rhapsody to build part of the Rhapsody tool itself
 - There are several cases where companies build models to identify boundaries for guiding security testing
- The major limiting factor is model update to code change cycle time; i.e., how fast does the model and code diverge
 - This is shown how the loss of favor in the creation of UML models
- There are some potential use cases to consider:
 - Model Driven Pipeline
 - Security
 - Certifications
 - AutoCode Generation (to include API management)
 - Embedded Systems
 - Reverse Engineering



What Have We Done

- Generated a dynamic process model (displayed as a set of interconnected webpages) of the DOD defined DevSecOps process with a “Best Practice” set of MBSE artifacts
- Identified programs currently executing (or planning to) MBSE with DevSecOps
- Have established a library of literature on the subject of DevSecOps and MBSE
- Coordinated a 3hr CIO DOD DSO COP meeting for MBSE on January 12, 2023
- Created DSO/MBSE Shift Left Playbook



What do we have left to do

- Finalize Draft MBSE/DSO Playbook
- Update current DSO/MBSE model to enable simulation to perform an independent value to divergence analysis
- Piloting 1-2 plays from the playbook with programs by August 2023
- Final MTR for the project by September 2023



DSO Process Model



MDD DevSecOps Process Content – Front Page

The screenshot displays the Eclipse Process Framework Composer interface. The main content area is titled "Guideline: MDD Dev Sec Ops Splash Page" and features a "Main Description" section. The description includes a "Welcome to Model-Driven Dev Sec Ops" heading, four icons representing "Model-Driven DevSecOps Delivery Process", "Relations Among Models", "Work Products", and "Roles", and an "Overview" section. The "Overview" section contains the following text:

Overview

This content depicts a DevSecOps process workflow - complete with worker roles and work product definitions - for performing DevSecOps that incorporates models of various kinds in the development of the systems of concern. This content is captured in the Systems and Software Process Engineering Metamodel (SPEM) language using the Eclipse Process Framework Composer (EPF Composer) tool. The workflows are rendered as activity diagrams which contain either task descriptors or nested activities. Most elements on the diagram are hyperlinked to specification content.

Purpose

The purpose of this content is to provide process guidance - what to do, when to do it, who does it, task definitions, and work products produced or consumed by the tasks. Many of the work products are digital engineering work products (models) of various kinds. Click [Relations Among Models](#) for a view of the various kinds of models and their interconnection to other models.

What is Dev Sec Ops?

DevSecOps is an approach to delivering software functionality that integrates software development, systems operations, and security practices together. It is characterized by extremely rapid (often daily) deployments of new software into the business environment, highly integrated and automated testing and security assessments, operations of the application - often in a cloud environment, and monitoring of the deployed application that provides feedback into the development part of the cycle. It is based on the notion of software factories implementing multiple software pipelines of features for rapid development and delivery.

What is Digital Engineering?

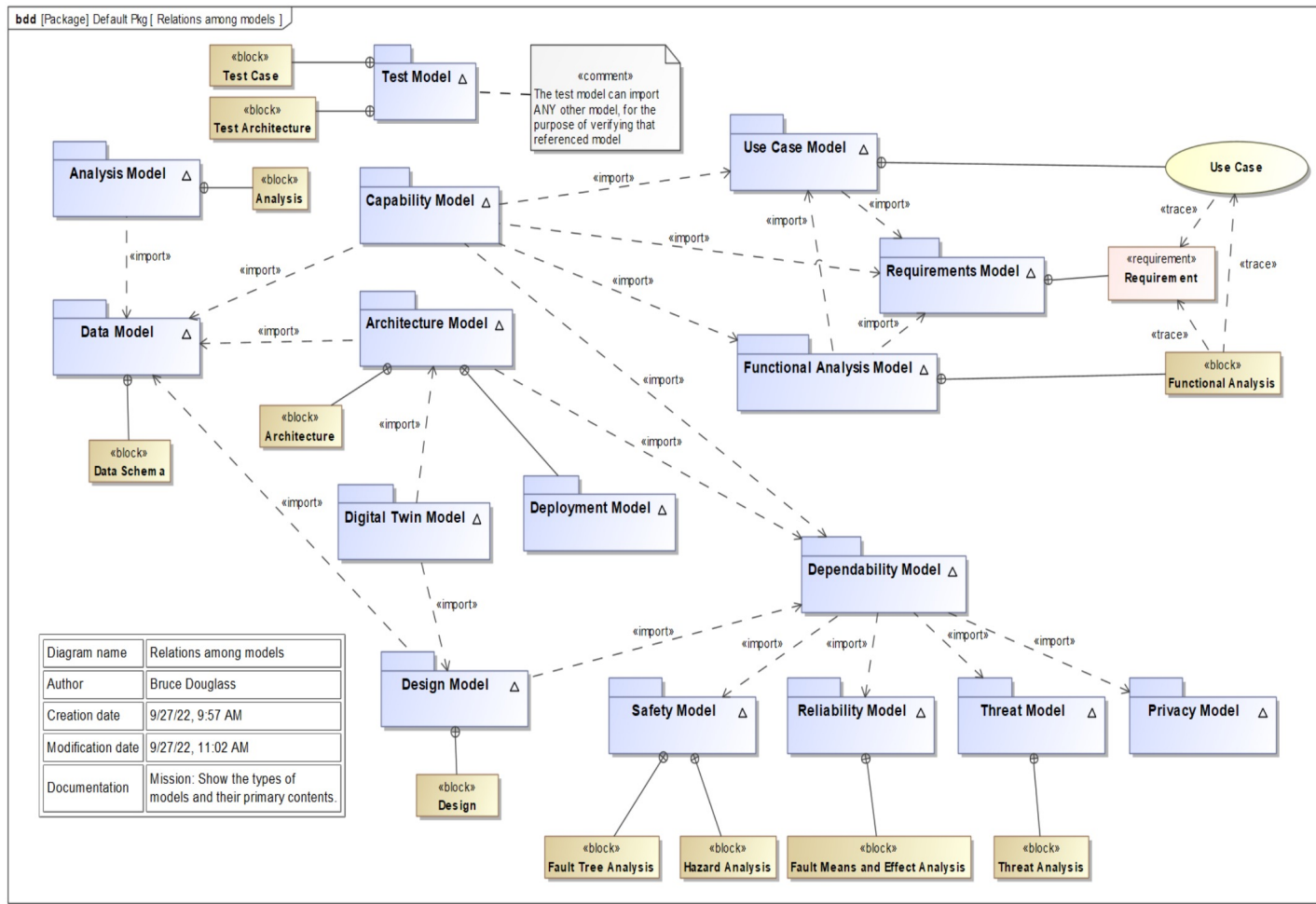
Digital Engineering is an approach to engineering that focuses on the storage and federation of engineering data as digitally-accessible information. This information is held in models of various kinds. Each datum has a single authoritative source of truth and can be accessed by other systems and workflows as needed. Digital engineering for systems engineering is known as *model-based systems engineering* (MBSE) while digital engineering for software is usually known as *model-driven development* (MDD).

What is a model?

The left sidebar of the interface shows a tree view of the project structure, including "Model-Driven DevSecOps", "MDD Dev Sec Ops Splash Page", "Relations Among Models", "Model-Driven DevSecOps Delivery Process", "Plan Program", "Plan Pipeline", "Develop", "Build", "Test", "Release & Deliver", "Deploy", "Operate", "Monitor", "Work Products", "Digital Engineering Work Products", "Analysis Model", "Capability Model", "Data Model", "Dependability Model", "Functional Analysis Model", "Modeling Guidelines", "Privacy Model", "Project Risk Model", "Reliability Model", "Requirements Model", "Safety Model", "Software Architecture Model", "Software Design Model", "Threat Model", "Use Case Model", "Other DevSecOps Work Products", "Roles", "Guidance", and "Term Definitions".

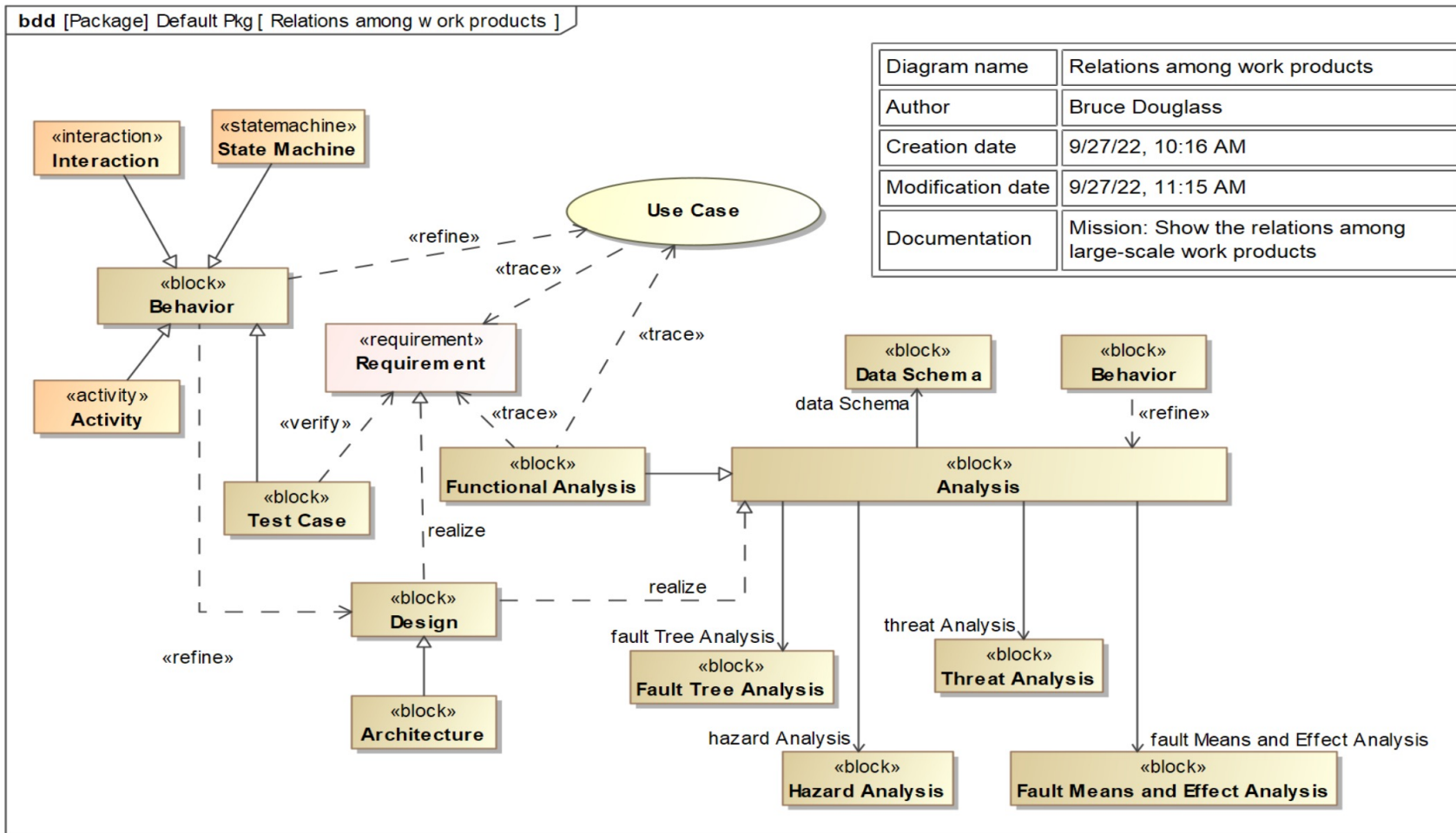


Types and Relations of Models in MDDSO





Relations Among Key MDD Work Products





MDD DevSecOps Process Content – Main Page

Eclipse Process Framework Composer

Delivery Process: Model-Driven DevSecOps Delivery Process

This process is created to give guidance for applied modeling in the DevSecOps environment.

Description | Work Breakdown Structure | Team Allocation | Work Product Usage

Expand All Sections | Collapse All Sections

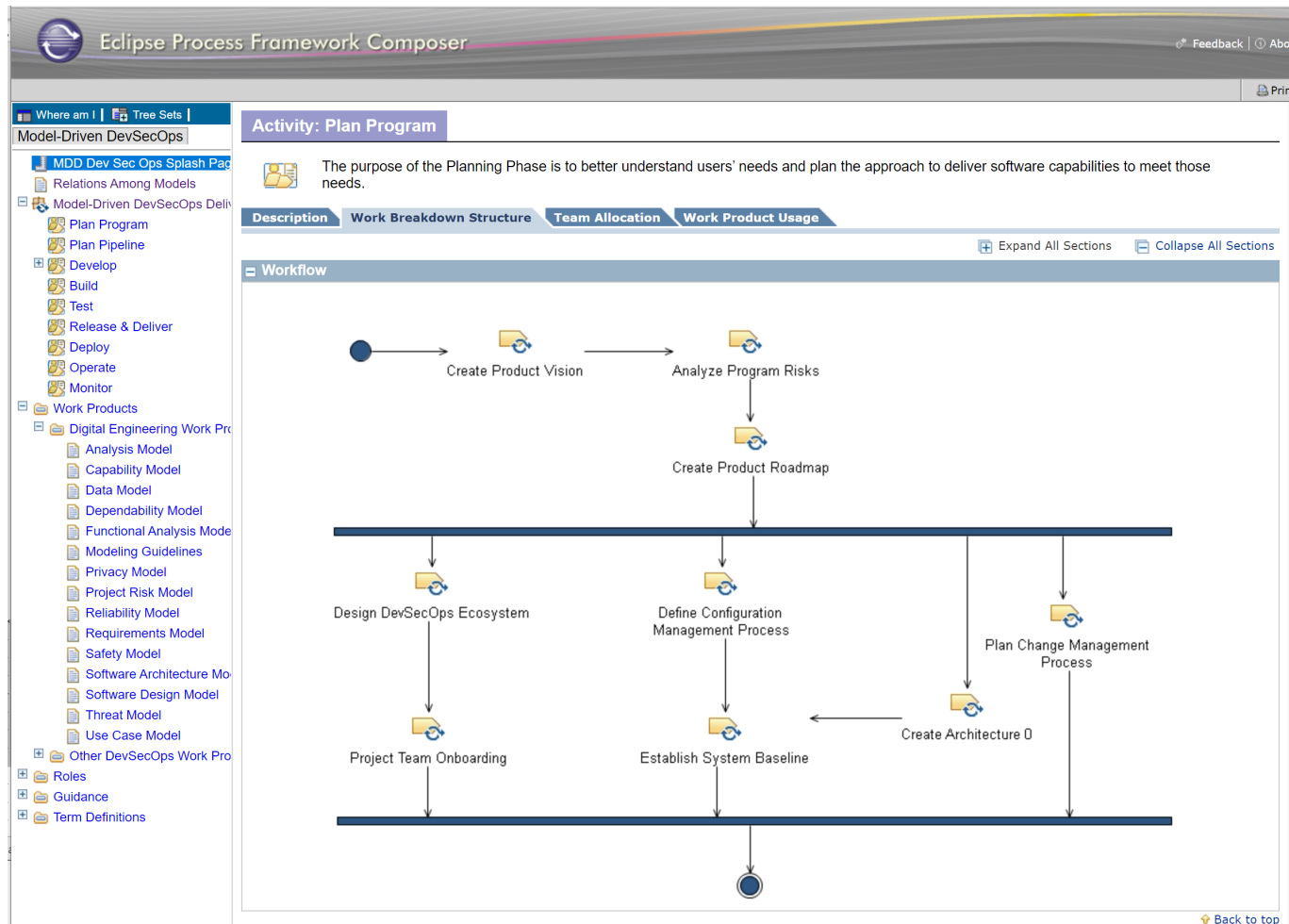
Workflow

The diagram illustrates the overall workflow of Model-Driven DevSecOps. It begins with a start node leading to 'Plan Program'. A decision diamond follows, with one path leading to 'Plan Pipeline' and another leading to 'Operate'. 'Plan Pipeline' leads to a decision diamond with two paths: one to 'Develop' and another to 'Build'. 'Develop' leads to a decision diamond with a path to 'Build' (labeled '[dev gate ok]') and another to 'Operate' (labeled '[else]'). 'Build' leads to a decision diamond with a path to 'Test' (labeled '[build gate ok]') and another to 'Operate' (labeled '[else]'). 'Test' leads to a decision diamond with a path to 'Release & Deliver' (labeled '[release tests pass]') and another to 'Operate' (labeled '[else]'). 'Release & Deliver' leads to a decision diamond with a path to 'Deploy' (labeled '[deployment successful]') and another to 'Operate' (labeled '[else]'). 'Deploy' leads to a decision diamond with a path to 'Operate' (labeled '[deployment successful]') and another to 'Release & Deliver' (labeled '[else]'). 'Operate' leads to a decision diamond with a path to 'Monitor' and another to 'Plan Program'.

Back to top



MDD DevSecOps Process Content – Workflow (Activity)





MDD DevSecOps Process Content – Workflow (Activity)

Eclipse Process Framework Composer Feedback | About Print

Where am I | Tree Sets |
Model-Driven DevSecOps

- MDD Dev Sec Ops Splash F
- Relations Among Models
- Model-Driven DevSecOps D
 - Work Products
 - Digital Engineering Work
 - Other DevSecOps Work I
 - Roles
 - Guidance
 - FMEA Example
 - FMEA Template
 - Model-Based Testing
 - Modeling Guideline Exarr
 - Risk Management Plan S
 - The Six Dimensions of Sc
 - Metrics
 - Defect Density Metric**
 - Escaped Defects Metric
 - Application Deployment F
 - Automatic Code Generati
 - Burndown Rate Metric
 - Design Requirements Co
 - Failure Rate Metric
 - Mean Attack Success Ra

Activity: Operate

This activity sequences the steps involved in running and operating an application in the DevSecOps environmetn. Click on the task descriptors to the task definitions and its metadata.

Description | **Work Breakdown Structure** | **Team Allocation** | **Work Product Usage**

[Expand All Sections](#) [Collapse All Sections](#)

Relationships

Parent Activities
<ul style="list-style-type: none">Model-Driven DevSecOps Delivery Process

[Back to top](#)

Description

Click on the task descriptors to the task definitions and its metadata.

[Back to top](#)

Properties

More Information

Supporting Materials
<ul style="list-style-type: none">Downtime MetricMean Attack Success Rate MetricMean Time to Detect MetricServer Availability MetricUptime Metric

[Back to top](#)



MBSE DSO Playbook



DSO/MBSE Playbook

- Plays
 - Model Driven Pipeline
 - Security
 - Certifications
 - Autocode Generation (to include API management)
 - Embedded Systems
 - Reverse Engineering
- Structure
 - *Overview*
 - *Goals and Objectives*
 - *Preconditions / Assumptions*
 - *Key tools or technologies in use in this play (if appropriate) (I'm thinking model-based threat analysis, code generation, ...)*
 - *Key Models and their value*
 - *Specific workflow*
 - *Play-specific specializations*
 - *Process Flow Variants (if appropriate)*
 - *Measuring success (metrics)*



Playbook Example – Automated Code Generation Play

- Overview
- The UML can be used to create software designs. Given appropriate tooling, high-quality source code can be generated from appropriately specified designs. The Rhapsody tool from IBM is especially well-known in the real-time and embedded development space for this. There are many benefits from this including:
 - Strong coupling and synchronization between the software design and the software implementation
 - The ability to verify a system at the design level as well as at the code level
 - Enhanced capability to visualize complex designs
 - Improved maintenance of development work products
 - Ease of tracing among requirements, design, and code
 - Design-level verification
 - Ability to use Model-Based Testing methods
 - Faster development of high-quality software implementations



Playbook Example – Automated Code Generation Play

- *Goals and Objectives*
- Specific goals and objectives of this play include:
 - Capture the design independently from the implementation to facilitate design understanding
 - Efficiently create software implementation from designs
 - Verify the design independent of software implementation detail
 - Ensure consistency between design and software implementation
- *Preconditions / Assumptions*
- To use this play, you must create high-quality designs with adequate precision to support the automatic generation of code. Additionally, your development environment must have a modeling tool capable of high-quality code generation. The developers must be sufficiently facile in the selected modeling language as well as in the target implementation language.
- *Key tools or technologies in use in this play*
- Modeling must be done in tools that can represent the selected design language and generate high-quality code in the target software implementation language. Rhapsody from IBM is a UML compliant tool that meets this need; Simulink from Mathworks is a control-theoretic modeling tool that is also commonly used.



Playbook Example – Automated Code Generation Play

- *Key Models and their value*
- **Specification Model**
 - The specification model represents the capabilities and requirements for the system of concern. Although not directly used to generate the software implementation, it drives the development of the design model. The key value of this model is that it supports requirements analysis and results in higher-quality requirements.
- **Architecture Model**
 - *Architecture* may be thought of as the set of strategic design decisions that organize and optimize the system as a whole. The largest scale parts of the architecture – often referred to as subsystems or components – hold the lower-level software implementations. These architectural elements serve as a scaffolding to support the application.
- **Design Model**
 - The design model identifies and specifies the software structures, including classes, functions, data, and interfaces. A design model typically provides structural, functional, behavioral, and parametric views of the software in an integrated fashion, from which the software may be generated. Since the design model abstracts away much of the implementation detail, it is far easier to focus on the correctness and completeness of the design.



Playbook Example – Automated Code Generation Play

- *Play-specific specializations*
- Tools vary widely in their ability to support full code generation. It is common to require some or even a great deal of manual touch up to the implementation after code generation.
- Some tools support “round-trip engineering.” This refers to the ability of the tool to accept modification to a generated software implementation to update the design. This differs from reverse engineering in that the overall design and visualizations of the design are maintained as changes to the generated code are incorporated back into the design.



Playbook Example – Automated Code Generation Play

Specific workflow

In principle, little is changed about the process flow itself (see Figure 1), but some of the tasks themselves are performed differently. Almost all of the differences occur in the Develop activity of the MDD DevSecOps Process

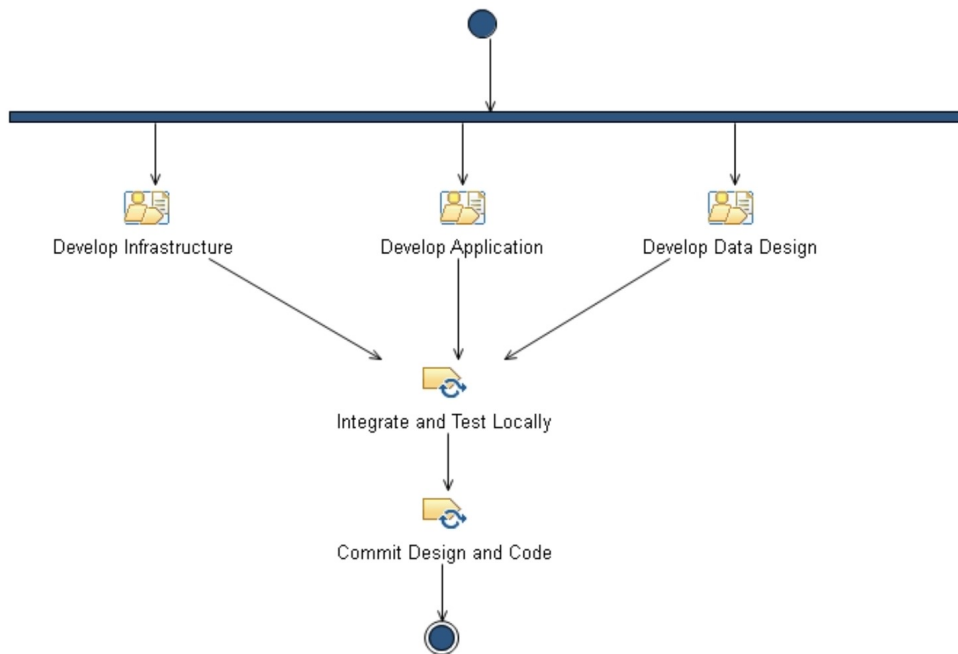


Figure 5: Develop Activity process flow



Playbook Example – Automated Code Generation Play

The details of the three development subactivities are shown in the next three figures with the modified tasks highlighted in red.

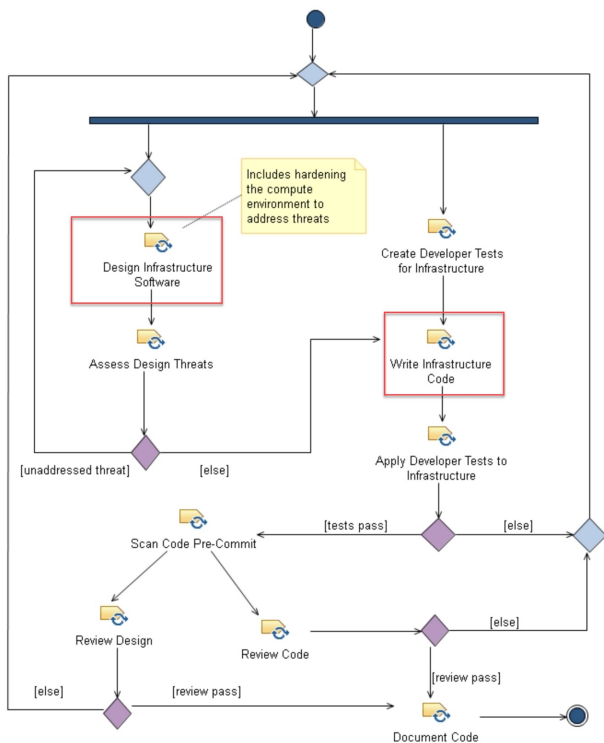


Figure 6: Develop Infrastructure Activity



Playbook Example – Automated Code Generation Play

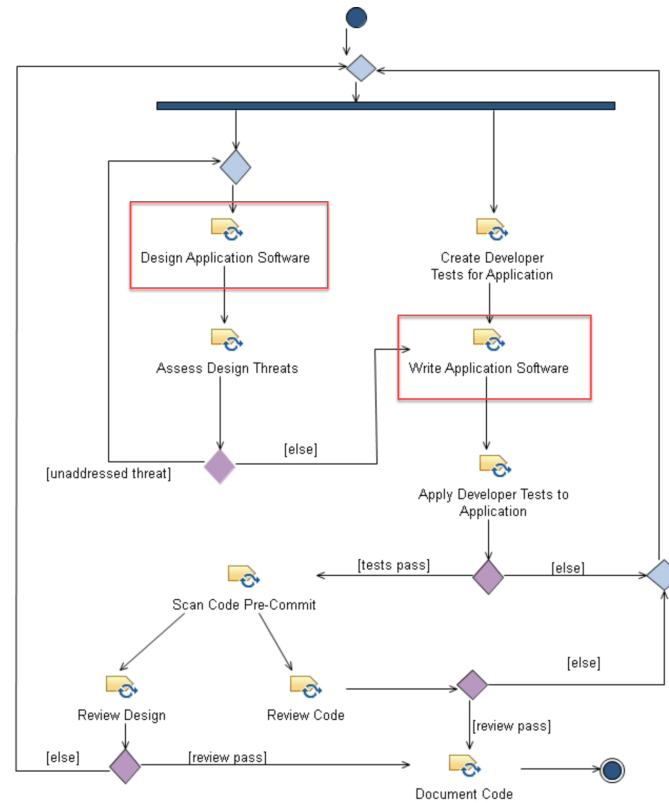


Figure 7: Develop Application Activity



Playbook Example – Automated Code Generation Play

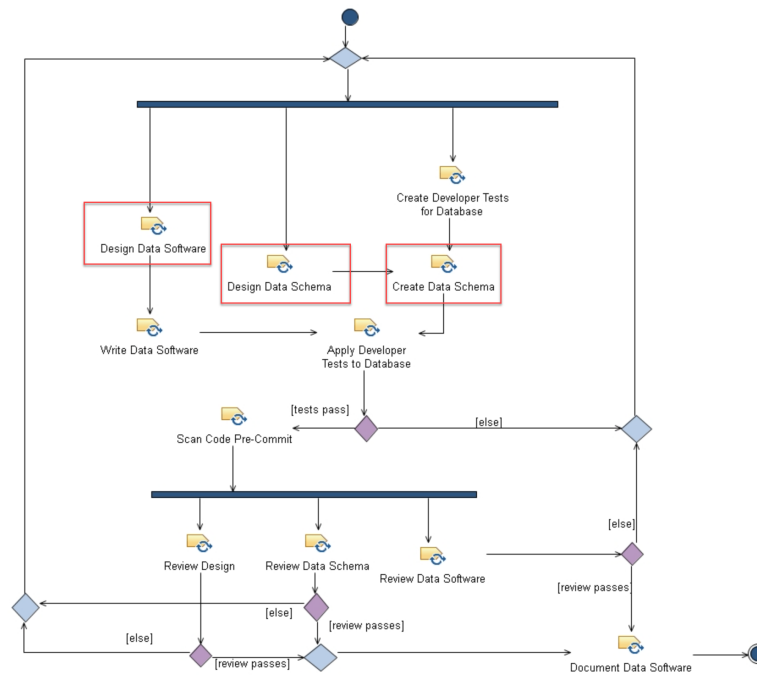


Figure 8: Develop Data Design Activity



Playbook Example – Automated Code Generation Play

Process Flow Variants (if appropriate)

- For data-centric models, it may be possible to create APIs for data access directly from the data models. Some tools support this generally or for specific middleware such as the Data Description Service (DDS) or Common Object Request Broker Architecture (CORBA) standards.

Measuring success (metrics)

- Metrics for this play include Total Generated Code (generally stated in Kilo-lines of source code generated) and Percentage of Generated Code (the ratio of the number of generated source code lines divided by the total number of lines of code).



Questions



Contact:

Allan Dianic allan.v.dianic.civ@mail.mil

Shane Smith sesmith@mitre.org

Dr. Bruce Douglass bdouglass@mitre.org