

# Performing Verification and Validation in Reuse-Based Software Engineering

## Software Engineering Workshop

Edward A. Addy

NASA/WVU Software Research Laboratory

NASA/Goddard Space Flight Center

Software Engineering Laboratory

December 3, 1998

# Impact of Reuse-Based Software Engineering on the Software Process

- Implementation of reuse-based software engineering (software product lines) impacts all areas of the software development process

→ Domain Engineering

→ Component Development

→ Reuse Library Maintenance

→ Requirements Analysis

→ Design and Implementation

→ System Integration

→ Testing

→ Configuration Management

→ Quality Assurance

→ Verification and Validation

## Framework for V&V in Reuse-Based Software Engineering

- **Current V&V work geared toward specific application systems, throughout the lifecycle of the system**
- **Reuse-based software engineering (software product lines) introduces software engineering at a level outside the individual application system**
  - **Common requirements**
  - **Architecture for multiple systems**
  - **Components used in multiple systems**
  - **Unit testing on shared components**
  - **Documentation**

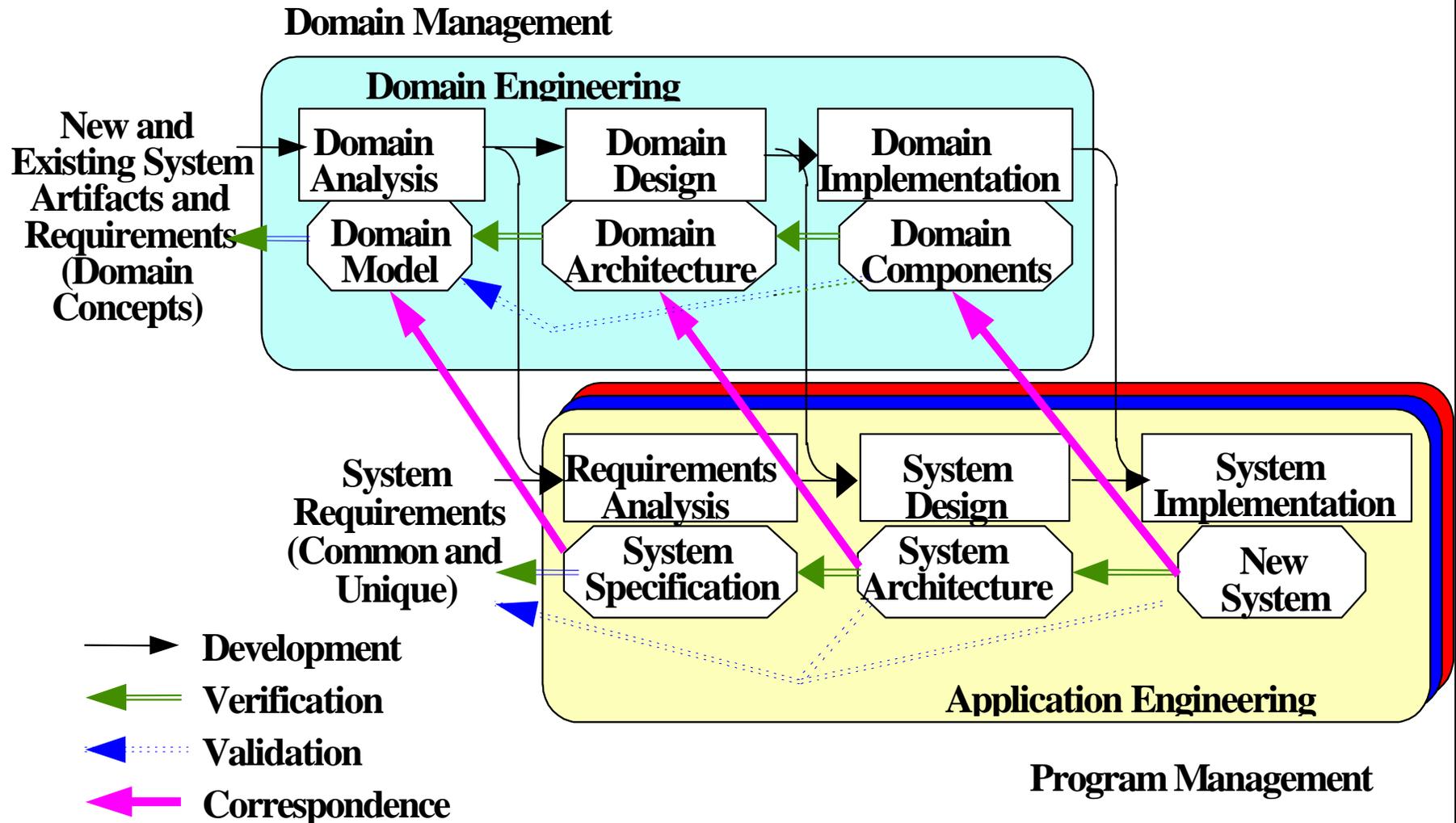
## The Context for V&V

- **“Software V&V is a systems engineering discipline that evaluates software in a systems context”**

**Delores Wallace and Roger Fujii, Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards, NIST Special Publication 500-165**

- **In reuse-based software engineering, the context for V&V is provided by the domain model and the generic architecture.**

# General Framework



## V&V Tasks

- **Domain-level:** tasks performed to ensure that domain products fulfill the requirements established during earlier phases of domain engineering.
- **Transition-level:** tasks to provide assurance that an application artifact correctly implements the corresponding domain artifact.
- **Application-level:** traditional V&V tasks to ensure the application products fulfill the requirements established during previous application lifecycle phases.

## Domain-Level V&V Tasks

PHASE	TASKS
<b>Domain Analysis</b>	<b>Validate Domain Model</b> <b>Model Evaluation</b> <b>Requirements Traceability Analysis</b>
<b>Domain Design</b>	<b>Verify Domain Architecture</b> <b>Design Traceability Analysis</b> <b>Design Evaluation</b> <b>Design Interface Analysis</b> <b>Component Test Plan Generation</b> <b>Component Test Design Generation</b>
<b>Domain Implementation</b>	<b>Verify and Validate Domain Components</b> <b>Component Traceability Analysis</b> <b>Component Evaluation</b> <b>Component Interface Analysis</b> <b>Component Documentation Evaluation</b> <b>Component Test Case Generation</b> <b>Component Test Procedure Generation</b> <b>Component Test Execution</b>

## Transition-Level V&V Tasks

PHASE	TASKS
Requirements	Correspondence Analysis between System Specification and Domain Model
Design	Correspondence Analysis between System Architecture and Domain Architecture
Implementation	Correspondence Analysis between System Implementation and Domain Components

- Map the application artifact to the corresponding domain artifact.
- Ensure that the application artifact has not been modified from the domain artifact without proper documentation.
- Ensure that the application artifact is a correct instantiation of the domain artifact.
- Obtain information on testing and analysis on a domain artifact to aid in V&V planning for the application artifact.

## **Criticality Analysis in Reuse-Based Software Engineering**

- **Criticality analysis is performed in order to allocate V&V resources to the most important (i.e., critical) areas of the software**
- **Question - How should criticality analysis be extended from V&V in application software engineering to V&V in reuse-based software engineering?**
  - **Component more critical in one system than in another**
  - **Component not critical in any individual system, but used in many systems (and hence critical to the organization)**
  - **Component will be replaced quickly in most systems, but will remain for a long period in a few systems**

## **Economic Considerations for Reusable Components**

- **Span of application** – the number of components or systems that depend on the component
- **Criticality** – potential impact due to a fault in the component
- **Marketability** – degree to which a component would be more likely to be reused by a third party
- **Lifetime** – length of time that a component will be used

**Source - Working Group on Component Verification, Validation, and Certification, WISR8, March 1997**