N91-22725

Synthesis: Intertwining Product and Process

David M. Weiss

SOFTWARE PRODUCTIVITY CONSORTIUM

Keynote Address 1:45 -2:45 Nov. 7, 1990

SYNTHESIS: INTEGRATING PRODUCT AND PROCESS

David M. Weiss

Software Productivity Consortium

Abstract

A current trend in manufacturing is to design the manufacturing process and the product concurrently. The goal is to make the product easy to produce by the manufacturing process. Although software is not manufactured, the techniques needed to achieve the goal of easily producible software exist. The problem is how to organize the software production process and the products to eliminate rework. The solution lies in viewing system production as creating different members of a family, rather than creating a new system each time requirements change. Engineers should be able to take advantage of work done in previous developments, rather than restating requirements, reinventing design and code, and redoing testing.

Synthesis is a proposed systematic process for rapidly creating different members of a program family. Family members are described by variations in their requirements. Requirements variations are mapped to variations on a standard design to generate production quality code and documentation. The approach is made feasible by using principles underlying design for change. Synthesis incorporates ideas from rapid prototyping, application generators, and domain analysis. This talk wll be a discussion of the goals of Synthesis and the Synthesis process. The technology needed and the feasibility of the approach will be briefly discussed. Finally, the status of current efforts to implement Synthesis methodologies will be given .

Topics

- Synthesis vision
- Components of a Synthesis process
  - The application engineering and domain engineering processes
- Application Engineering: Building the application
- Domain Engineering: Building the application engineering environment
- Summary

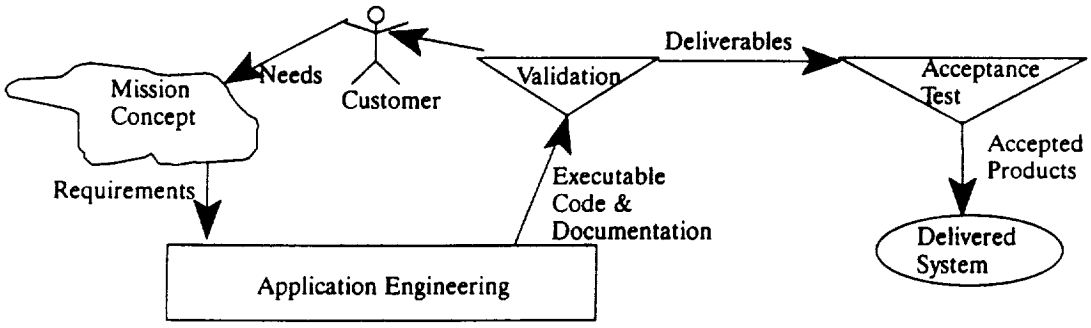
Typical Problems

- Ill–defined and changeable requirements
- Confusion of requirements, design, code
- Transformational barriers
  - Requirements –>Design –> Code
  - Requirements –> Test
- Rediscovery and reinvention

Goals

- Bring the customer into the production loop for validation
- Separate the concerns of requirements determination and validation from design, coding, and testing
- Respond rapidly to changes in requirements
- Rapidly generate deliverable products
  - generate code and documentation
  - achieve high productivity
  - achieve high quality
- Achieve systematic reuse
  - capture and leverage expertise
  - reuse systems

The Synthesis Application Development Process: Idealized



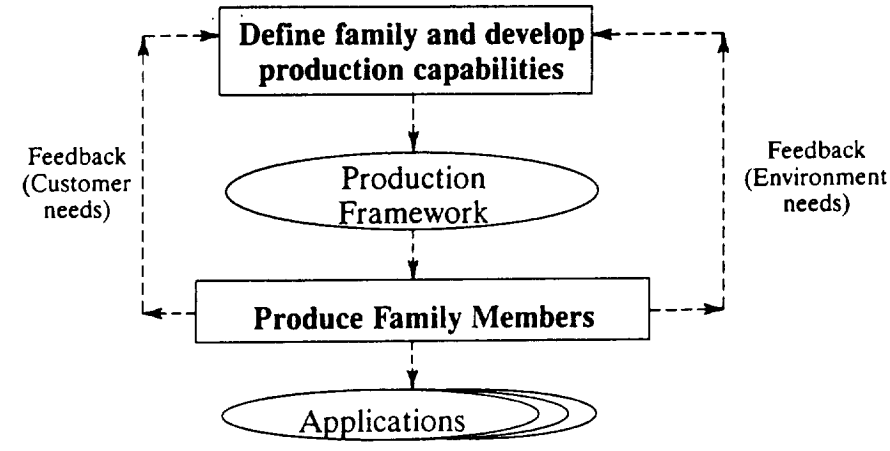
Key:

Process | Product | Customer Process

Approach

- Integrate the development process and the product
  - Design for producibility
  - Concurrent engineering for software
- Reorganize the software development process
  - Evolve a family rather than build single systems
- Develop systematic approach to building flexible application generators
- Use existing technology

A Synthesis Process



Examples of Similar Approaches

- YACC, LEX—parser/compiler applications
- Tedium—flexible application generator (MIS orientation)
- Systematica—generation of CASE tools
- Toshiba software factory
  - Generation of power plant software
  - Standardized design
  - Automated generation of 70%–80% of delivered code
- Spectrum
  - Prototype application engineering environment
  - Standardized design
  - Automated generation of code

Synthesis

Any methodology for constructing software systems as instances of a family of systems

- Process + Methods + Workproducts

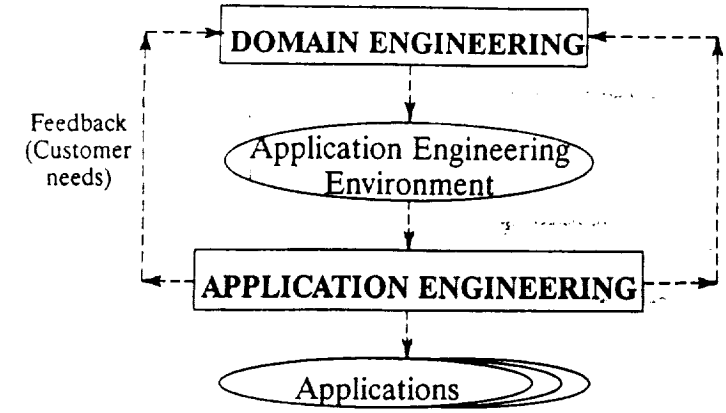
Components of Synthesis

- *Application Engineering*: An iterative process for constructing application systems.
  - Requirements are described by an *application model*
  - Rapidly determine requirements and generate deliverable software
- *Application Engineering Environment*: A framework (automated or manual) that supports a prescribed application engineering process.
  - Rapid prototyping and generation of systems
  - Automated generation of members of program families
  - Automated reuse of systems

Components of Synthesis (continued)

- *Domain Engineering*: A repeatable, iterative process for the design and development of both a family of systems and an application engineering process for the family.
  - Systematic development of product families for member company domains
  - Missing step in current processes
- *Domain Model*: A specification for an application engineering environment.
  - Conceptual framework (Language for specifying application models)
  - Reuse architecture (Standard, adaptable design)
  - System composition mapping (Map from language to reuse architecture)
- *Domain*: (1) A business area
  (2) A family of applications to be created within a business area

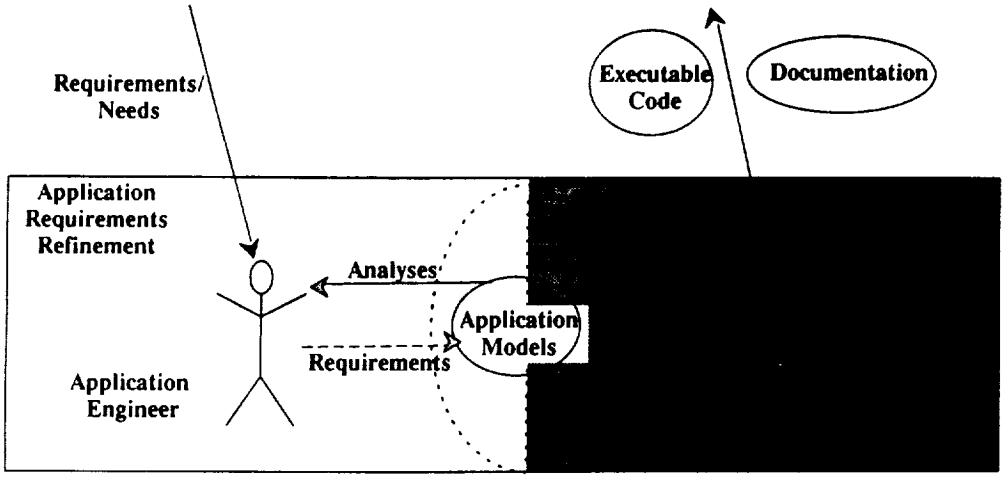
A Synthesis Process



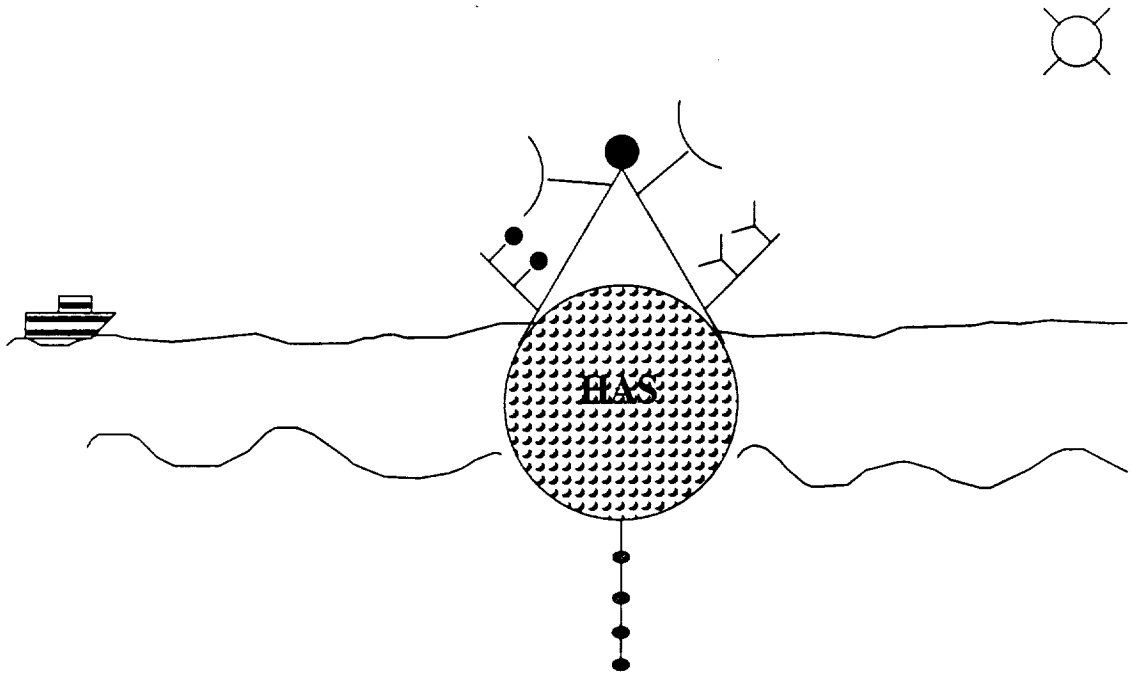
Key:

Product | Process

The Application Engineering Process



Application Engineering for Host-at-Sea (HAS) Buoy Systems



The Host-at-Sea (HAS) Buoy System

HAS Buoys drift at sea and monitor and report on environmental conditions. A typical HAS Buoy:

- Is equipped with a set of *sensors* that monitor *environmental conditions*, such as air and water temperature and wind speed. The value of a particular condition at a given time is determined by averaging sensor readings.
- Can determine its *location,* using the *Omega* or some other navigation system.
- Maintains a *history* of the environmental data it has collected, a history of its location, and a correlation between the two.
- Is equipped to transmit and receive *messages* via radio.
- Periodically transmits *messages* containing current weather information.
- Responds to requests that it receives, via radio, to transmit more detailed reports and to transmit weather *history* information.

The Host-at-Sea (HAS) Buoy System (continued)

HAS Buoys drift at sea and monitor and report on environmental conditions. A typical HAS Buoy:

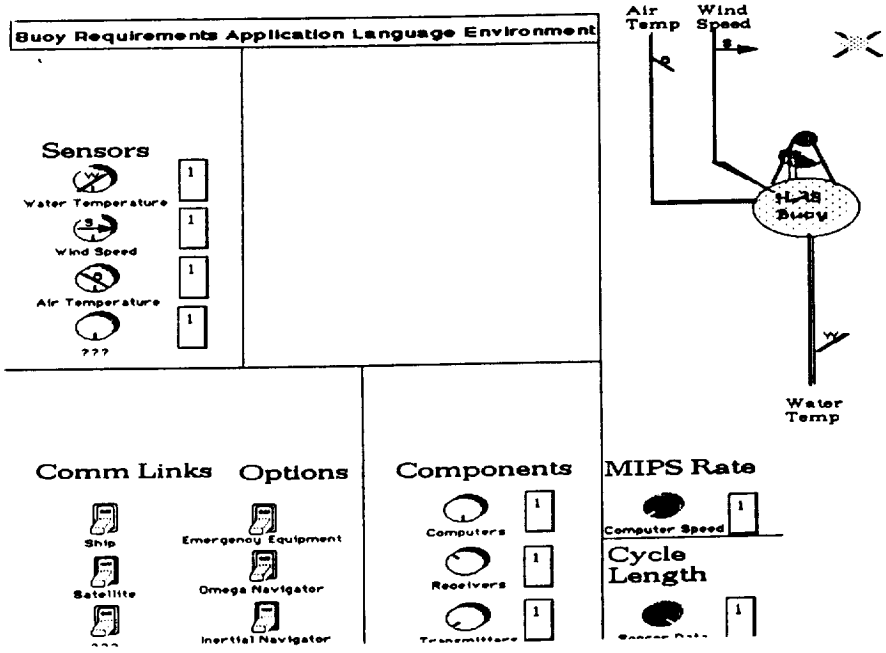
- Is equipped with an *emergency switch*, which, when flipped, causes the buoy to transmit an SOS signal in place of its periodic wind and temperature reports.
- Has a red light that it can turn on to be used in emergency rescue operations.
- Can accept location data from passing ships, via radio messages.
- Performs *built-in tests* (BIT) to determine if its computer and sensors are operating properly.

HAS as a Real Time System

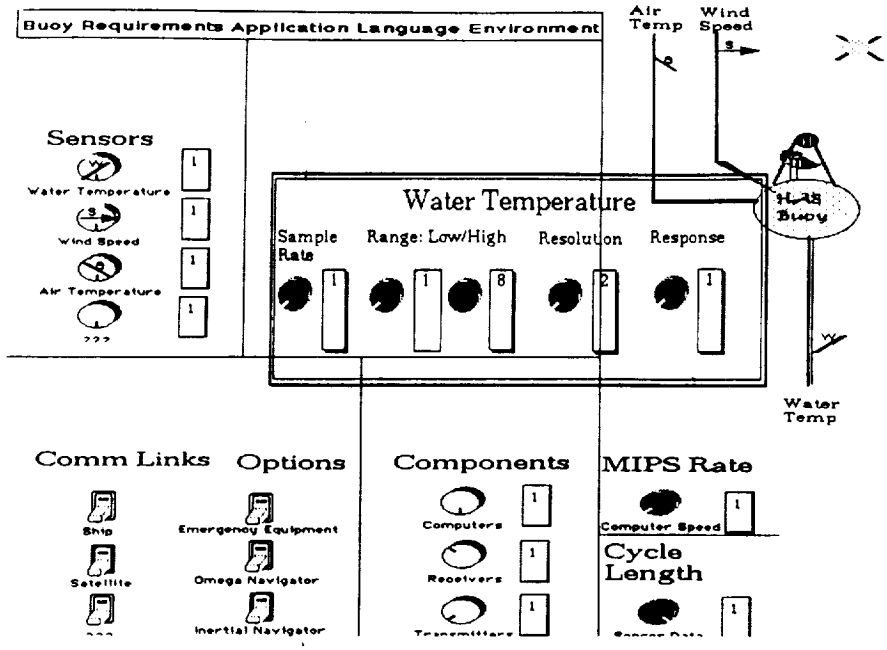
- Meet real-time deadlines for sampling sensors, sending messages
- Perform several functions concurrently, i.e., message broadcasting, data collection
- Reorder processing priorities based on occurrence of external events
- Receive and respond to requests from other systems
- Maintain history data base
- Handle exceptions, such as resource failures, without human intervention
- Missing: human-computer interface

SOFTWARE PRODUCTIVITY CONSORTIUM

The Buoy Application Engineering Environment



The Buoy Application Engineering Environment

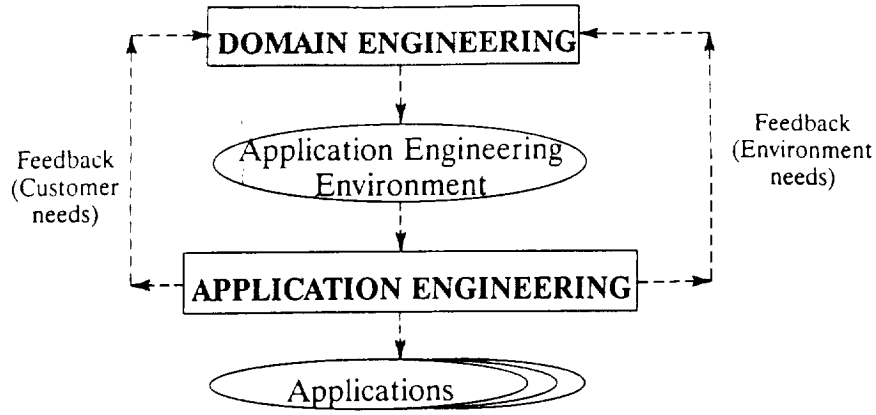


SOFTWARE PRODUCTIVITY CONSORTIUM

Characteristics of the Buoy Application Engineering Environment

- Focuses on requirements decisions, independent of design and implementation
- Focuses on variabilities used to describe members of the Buoy family
- Generates code and documentation for the application engineer (transparently)
- Simulates Buoy operations in a form meaningful to the customer
- Analyzes consistency, completeness, cost, and performance of the application model
  - -- Do I have enough MIPS to do the job?
  - -- Have I specified unnecessary redundancy?
  - -- How much will it cost?

A Synthesis Process

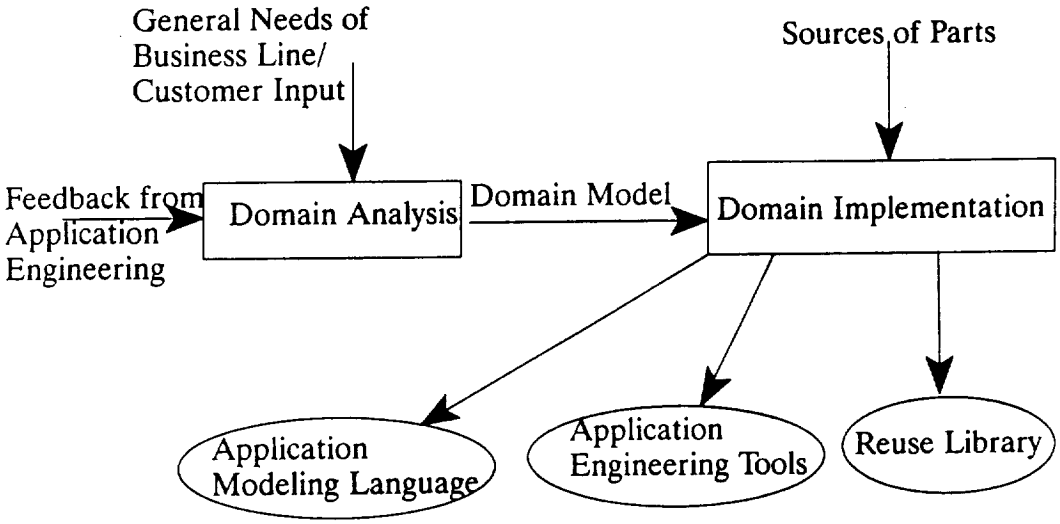


Key:

Product | Process

SOFTWARE PRODUCTIVITY CONSORTIUM

Domain Engineering: Building the Environment



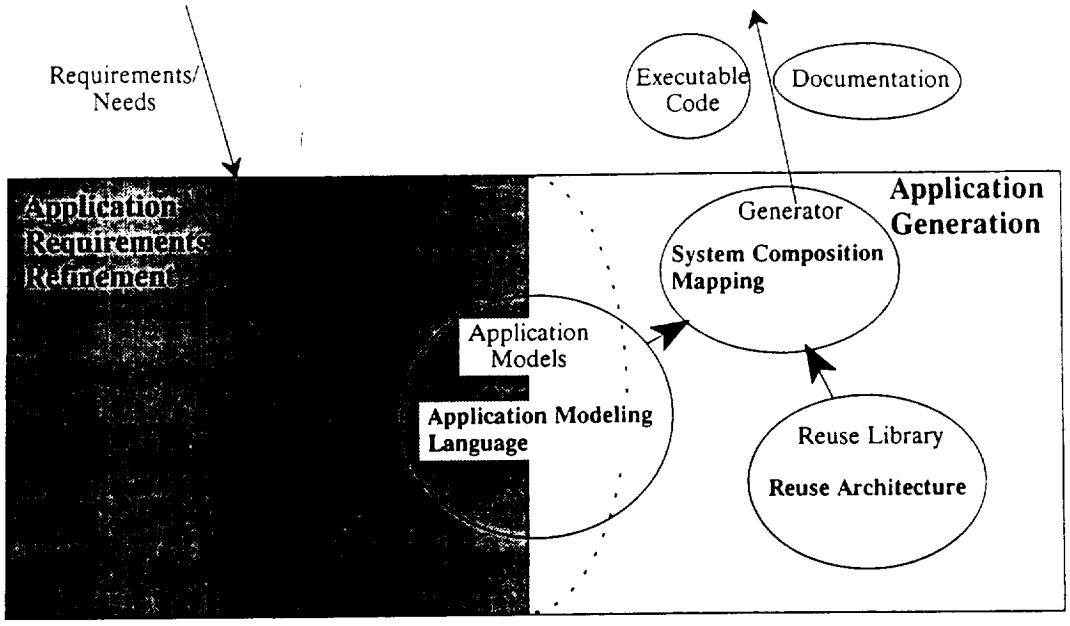
Domain Analysis

A Process For Refining/Creating Specifications For An Application Engineering Environment

The Domain Model

- Conceptual Framework
  - Application Modeling Language: Language for stating requirements
- Reuse Architecture
  - System software architecture organized for ease of reuse through adaptation
  - Specifications for parts for reuse library
- System Composition Mapping
  - Process for selecting and adapting parts for generation of code and documentation
  - Mapping from the modeling language to corresponding entities in the reuse architecture

Application Engineering Using The Domain Model



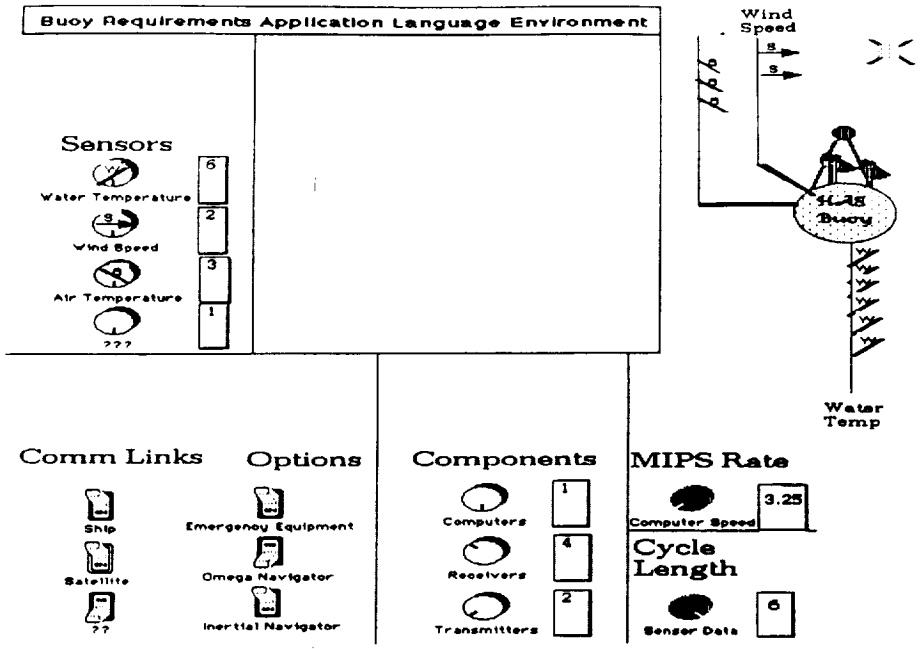
SOFTWARE PRODUCTIVITY CONSORTIUM

Application Modeling Language

Goal: Representation of requirements in application terms

- Support engineering decision making
- Specialize for the domain
- Separate representation, semantics, presentation
- Permit representation of expected variations

Application Modeling Language for The HAS Buoy Domain



Reuse Architecture

Goal: Define mechanically adaptable products for the domain

- Create information hiding class hierarchy to manage changeability and guide design decomposition
  - Divide software components into classes according to:
    1) Relative likelihood of change
    2) Origin of change (functional requirements, environment, software design decisions)
    -- Each class is an abstraction
  - Write specifications for components to be stored in reuse library
- Create process structure (for real-time systems) to support reconfigurability
- Identify dependencies among components to support systematic reuse

The System Composition Mapping

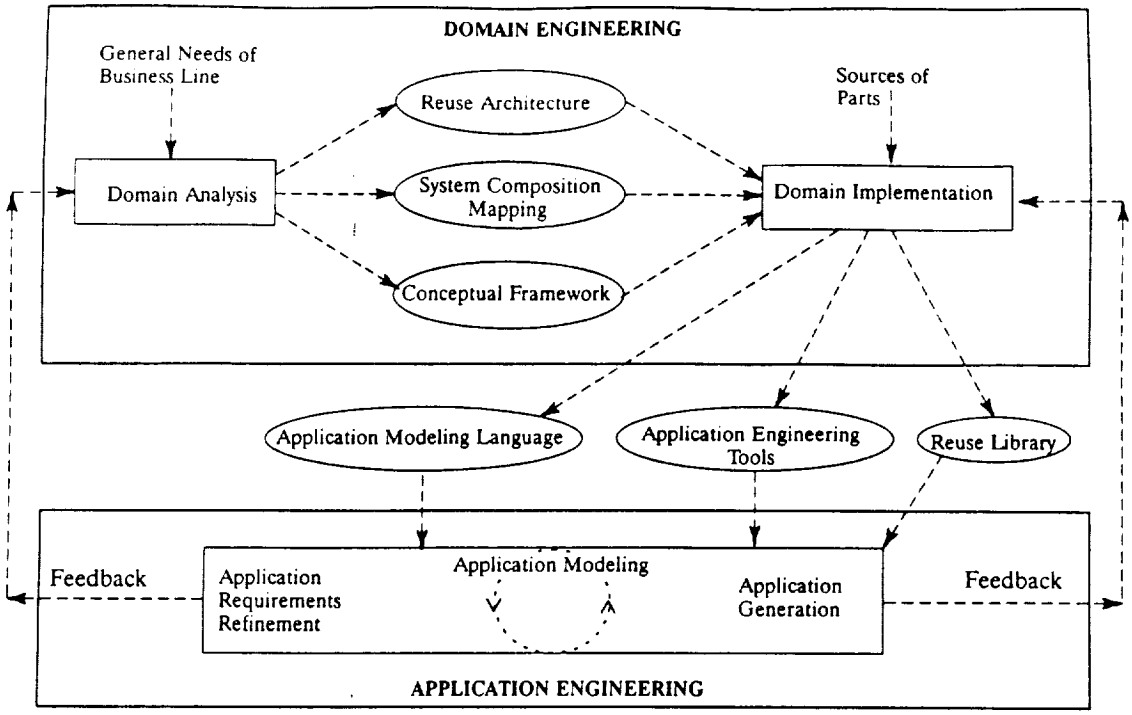
Goal: Select and adapt classes to compose applications that satisfy application models

- For each set of variations described in an application model, select appropriate parts from the reuse library
- Instantiate (Adapt) parts as determined by the application model

Domain Implementation

A Process for Refining/Creating An Application Engineering Environment

Synthesis: Major Products and Processes



Synthesis

- **Synthesis**: Any methodology for the construction of software systems as instances of a family of systems having similar descriptions.
- **Synthesis process**: Any systematic process for producing a reuse architecture, application modeling language, and system composition mapping within an application domain.

Key Synthesis Concepts

- ***Families of Systems***

  Domains are formalized as families of systems that share many common features. Software systems are derived as instances of a family,
  not as single unique systems.

- ***Model-Based Specification and Analysis***

  Specify requirements and system-building decisions precisely in an application model suitable for analysis,
  not constantly rework solution–specific representations.

- ***Reuse Architecture Designed for Adaptation***

  Creation and pre–planned reuse of mechanically adaptable subsystems based on engineering decisions,
  not opportunistic search and match with "reusable" parts.

- ***System Composition Mapping***

  Mapping from variations in an application model to adaptations in all deliverables for the implementing subsystems,
  not just tracing to possibly affected components.

Summary

- The technology to improve the software production process exists
- Reorganizing software production to take advantage of the family viewpoint is the key to improvement
  - One organization that concentrates on continually improving production of family members (process oriented)
  - One organization that concentrates on determining requirements for family members (project oriented)
- Similar reorganizations are happening in engineering fields
  - customer involvement
  - shorter time to market
  - more variation across product line

Session 1

Lessons Learned in Software Engineering

Chair: **Gary Raines**, *Manager, Avionics Systems Development Office, NASA/JSC*

Report from NASA Ada User's Group

John R. Cobarruvias
Flight Data Systems Division, NASA/JSC

Paper not available at time of printing.

Software: Where We Are & What is Required in the Future

Jerry Cohen
Boeing Aerospace and Electronics

Paper not available at time of printing.

Managing Real-Time Ada

Carol A. Mattax

Hughes Aircraft Corp., Radar Systems Group

Paper not available at time of printing.