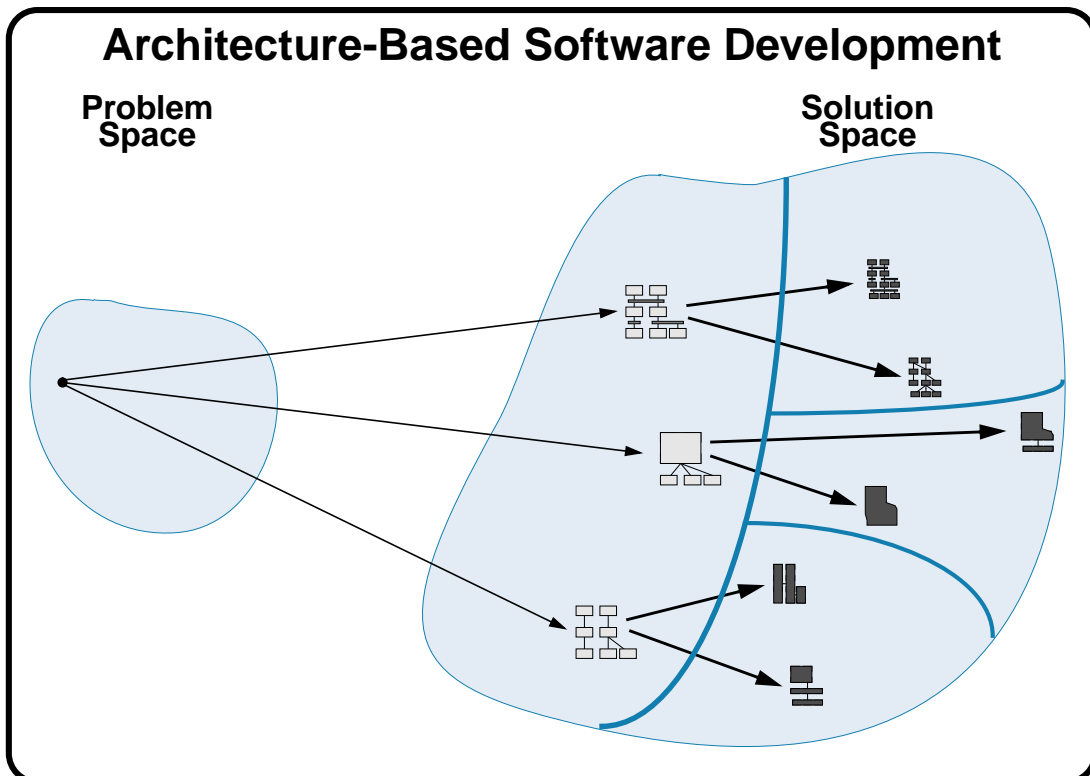
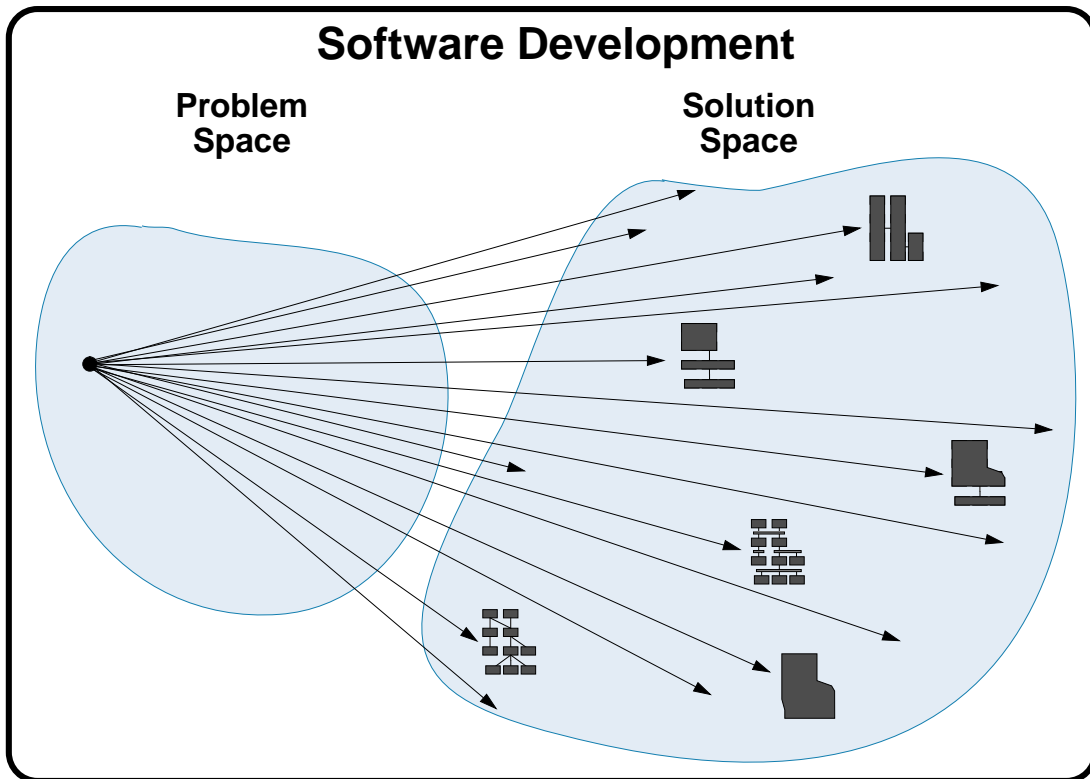


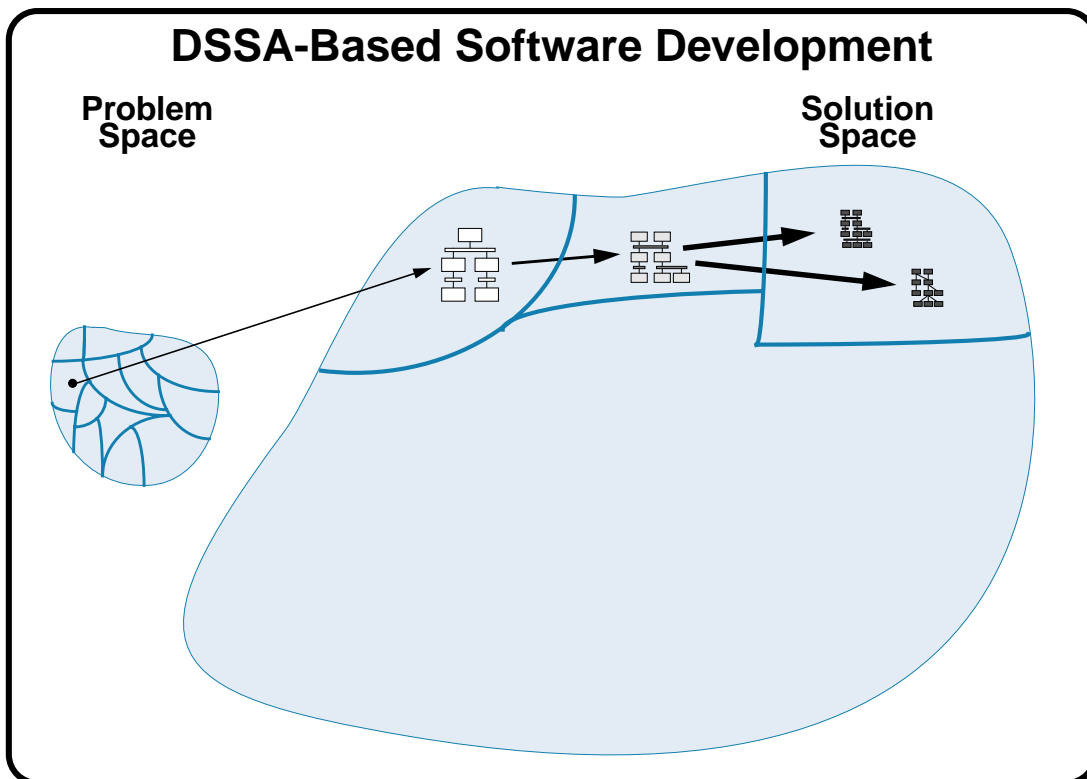
Review — Sources of Architecture

- Main sources of architecture
 - black magic
 - architectural visions
 - intuition
 - theft
 - method
- Routine design vs. innovative design
 - good designers vs. great designers
 - cost vs. optimality vs. repeatability
- “Architecting” consists of
 - *decomposition* of a system into constituent elements
 - *composition* of (existing) elements into a system
- Process-driven vs. checklist-driven architecting
 - heuristic-based vs. rule-based design space navigation

Why Domain-Specific?

- Development in specific domains can be optimized
 - maximize method and theft
 - minimize intuition
- Reuse in specific domains is most realistic
 - reuse in general is too difficult to achieve
 - focus on particular classes of applications with similar characteristics
- Criteria for successful reuse [Biggerstaff]
 - well-understood domain
 - slowly changing
 - has intercomponent standards
 - provides economies of scale
 - fits existing infrastructure





What Is DSSA?

- DSSA is an assemblage of software components
 - specialized for a particular type of task (domain)
 - generalized for effective use across that domain
 - composed in a standardized structure (topology) effective for building successful applications

– Rick Hayes-Roth, 1994

- DSSA is comprised of
 - a domain model,
 - reference requirements,
 - a reference architecture (expressed in an ADL),
 - its supporting infrastructure/environment, and
 - a process/methodology to instantiate/refine and evaluate it.

– Will Tracz, 1995

What Is a Domain Model?

- “All models are wrong; some are useful.”
 - *unknown, SEI*
- A domain model is a representation what happens in a domain
 - functions being performed
 - data, information, and entities flowing among the functions
- It deals with the *problem space*
- Fundamental objective:
 - standardize problem-domain terminology and semantics
 - terminology + semantics = ontology
 - provide basis for standardized descriptions of problems to be solved in the domain

Domain Analysis

- Domain model is a product of domain analysis
- Domain analysis entails
 - identifying, capturing and organizing objects and operations
 - their description using a standardized vocabulary
 - in a class of similar systems
 - in a particular problem domain
 - to make them reusable when creating new systems
- “Domain analysis is like several blind men describing an elephant”

Elements of a Domain Model (1)

- Customer needs statement
 - identifies functional requirements
 - high-level
 - informal
 - ambiguous
 - incomplete
- Scenarios
 - functional requirements
 - data flow
 - control flow
 - elicited from the customer
- Domain dictionary
 - definitions of terms used in the domain model
 - updated over time

Elements of a Domain Model (2)

- Context (block) diagram
 - high-level data flow between the major components in the system
- Entity/relationship diagrams
 - aggregation (“a-part-of”) and generalization (“is-a”) relationships
- Data flow models
- State transition models
- Object model
 - first phase of component interface design

What Are Reference Requirements?

- Requirements that apply to the entire domain
- Reference requirements contain
 - *defining* characteristics of the problem space
 - functional requirements
 - *limiting* characteristics (constraints) in the solution space
 - non-functional requirements (e.g., security, performance)
 - design requirements (e.g., architectural style, UI style)
 - implementation requirements (e.g., platform, language)

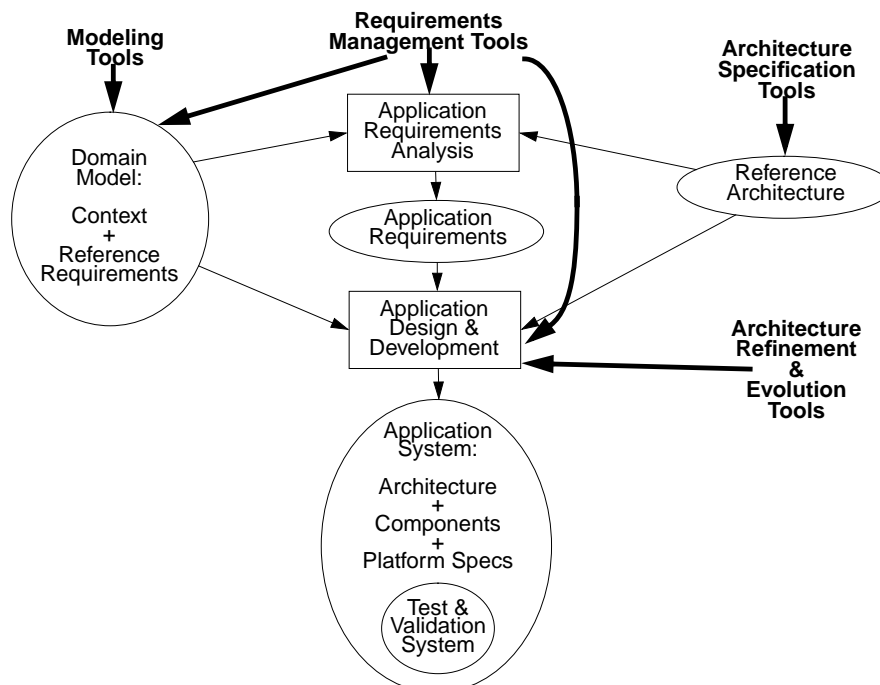
What Is a Reference Architecture?

- A standardized, generic architecture describing all systems in a domain
 - focuses on fundamental abstractions of the domain that expose a hierarchical, compositional nature
 - *a set* of reference architectures may be needed to satisfy all reference requirements
- Based on the constraints in reference requirements
- Syntax and semantics of constituent high-level components are specified
- It is reusable, extendable, and configurable
- A reference architecture is *instantiated* to create a design architecture for a specific application system

Elements of a Reference Architecture

- Reference architecture model
 - topology based on the architectural style
- Configuration decision tree
 - to instantiate a system from the reference architecture
- Architecture schema or design record
 - information about components and design alternatives
 - domain- and implementation-specific
- Reference architecture (component) dependency diagram
- Component interface descriptions
- Constraints
 - parameter value ranges and relationships
- Rationale

DSSA Process with Principal Supporting Tool Types



Primary DSSA Support Tool Types

- Domain modeling tools
- Requirements management tools
 - describe new application objects, attributes, and relationships
 - detect inconsistency, incompleteness, ambiguity
- Architecture specification tools
 - refine reference architectures
 - describe and constrain components and connectors
- Architecture refinement and evolution tools
 - configure reference architecture to meet application-specific requirements
 - specify and instantiate components
 - compose configurations into an executable form

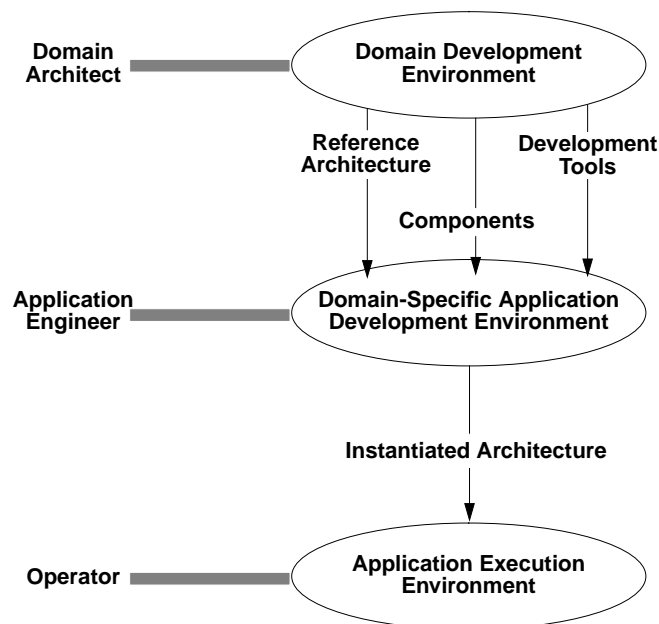
Secondary DSSA Support Tool Types

- Repository tools
 - store and retrieve components
- Component selection tools
- Component generators
- Architecture package, load, and exercise tools
 - aid in the integration and configuration of components
- Requirements validation tools
 - testing and analysis
- Performance validation tools
 - observe system in execution or simulation

Tertiary DSSA Support Tool Types

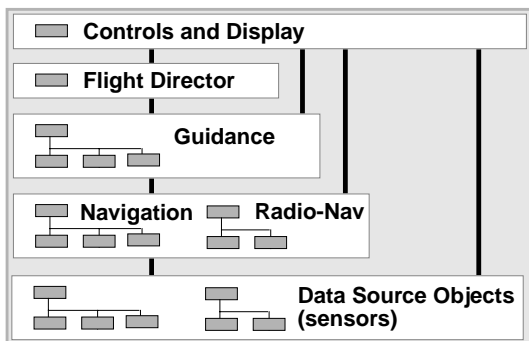
- Language processing and compiling
- User interfaces
- Databases
- Distributed and real-time computing
- Documentation
- Configuration management
- Role
 - used to develop tools at the previous two levels
 - generic rather than DSSA-specific

Three Layers of DSSA Environments



Avionics DSSA

- **Avionics Domain Application Generation Environment**
- **Layered reference architecture**
 - subsystems decomposed into primitive components with standardized interfaces
 - over 40 different realms with over 350 distinct components
 - $\text{realm} \equiv \{ x : \text{component} \mid (\forall i,j)(x_i.\text{interface} = x_j.\text{interface}) \}$
- **ADAGE reference architecture model:**



- reference architecture is defined by component realms and domain-specific composition constraints
- even simple avionics systems often require over 50 distinct components stacked 15 layers deep

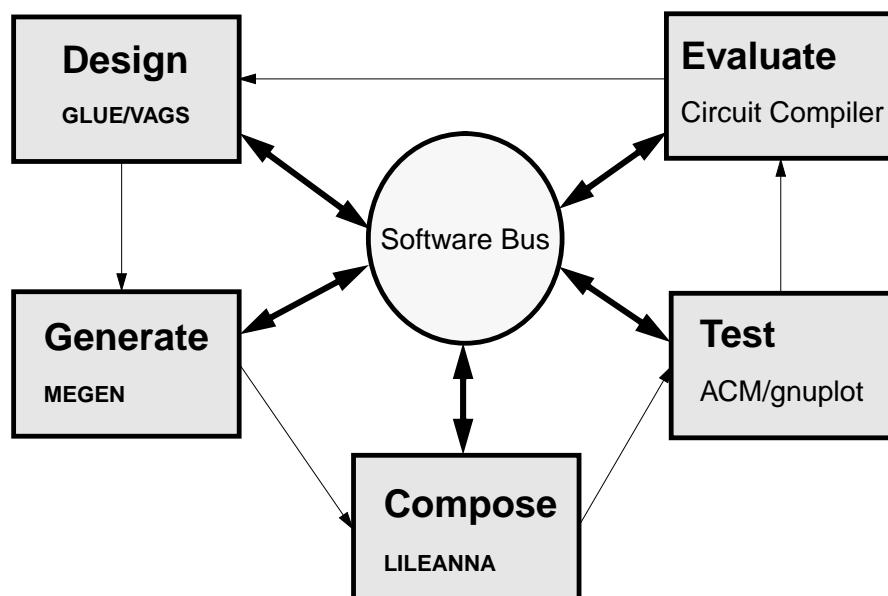
ADAGE Tool Suite (1)

- **DOMAIN (DOmain Models, All Integrated)**
 - a hypermedia scenario-based requirements capture and modeling tool
- **Avionics Example Test Bed**
 - simulation and analysis tools (ACM and gnuplot)
- **Software Circuit Compiler**
 - timing analysis tool
- **Realm Component Builder and Compiler**
 - tool for describing components in the reference architecture
- **GLUE (Graphical Layout User Environment)**
 - GUI for configuring and viewing the reference architecture
- **VAGS (Variational Attribute Grammar System) - constraint checker**

ADAGE Tool Suite (2)

- **MEGEN** (Module Expression GENERator)
 - an application generator
- **LILEANNA**
 - an ADA-specific high-level system description language
- **Scientist Work Bench**
 - a data/program organization/launching tool
- **Software Component Design Record Editor**
 - an information store for organizing reusable software artifacts

ADAGE Development Environment Architecture

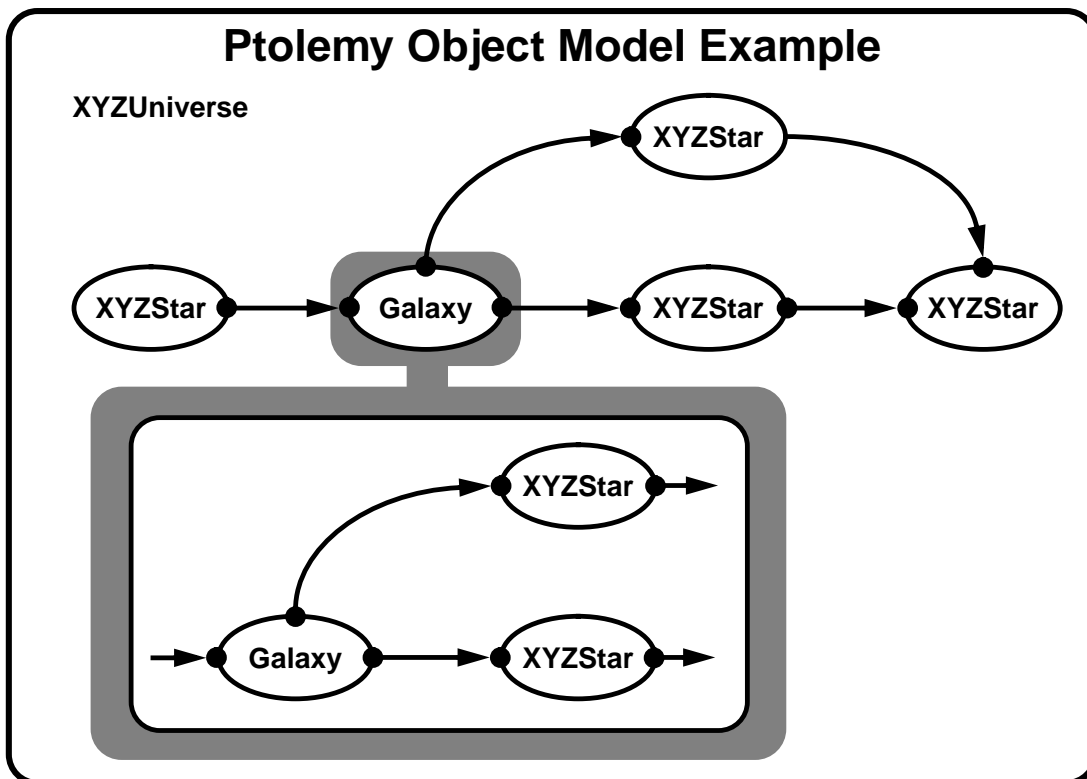


Signal Processing DSSA

- Ptolemy
 - a framework for simulation and rapid prototyping of heterogeneous systems
 - focus on signal processing and communication systems
- Comprehensive scope
 - algorithms and communication strategies
 - simulation
 - hardware and software design
 - parallel computing
 - generating real-time prototypes
- Domain (new telecom services) requires joint design of
 - control software
 - signal processing
 - transport
 - hardware elements

Ptolemy Object Model

- *Block* — reusable S/W component
- *Star* — atomic Block
- *Galaxy* - composite Block
- *Universe* — a complete Ptolemy application
- *PortHoles* — standard interface for Block communication
- *Particles* — Blocks communicate using streams of these
- *Scheduler* — determines operational semantics of a network of blocks by ordering their invocations
- *Target* — controls execution
 - by invoking a Scheduler for a simulation-oriented application
 - by synthesizing code, invoking an assembler, downloading code into attached hardware, etc. for synthesis-oriented applications



Ptolemy Object Model (cont.)

- *Domain* - a set of Blocks, Targets, and associated Schedulers conforming to common computational model
 - computational model is operational semantics governing Block interactions
 - example domains:
 - SDF - synchronous dataflow; for signal processing
 - DDF - dynamic dataflow; for signal processing
 - MQ - message queue; for telecom switching software
 - DE - discrete event; network or high-level system modeling
- *Wormhole* - mechanism enabling different Domains to coexist in a Ptolemy application
 - externally - a Star that obeys the operational semantics of the external Domain
 - internally - an entire foreign Universe, with Scheduler and Stars for that Domain

