

Interoperability Conflict Analysis

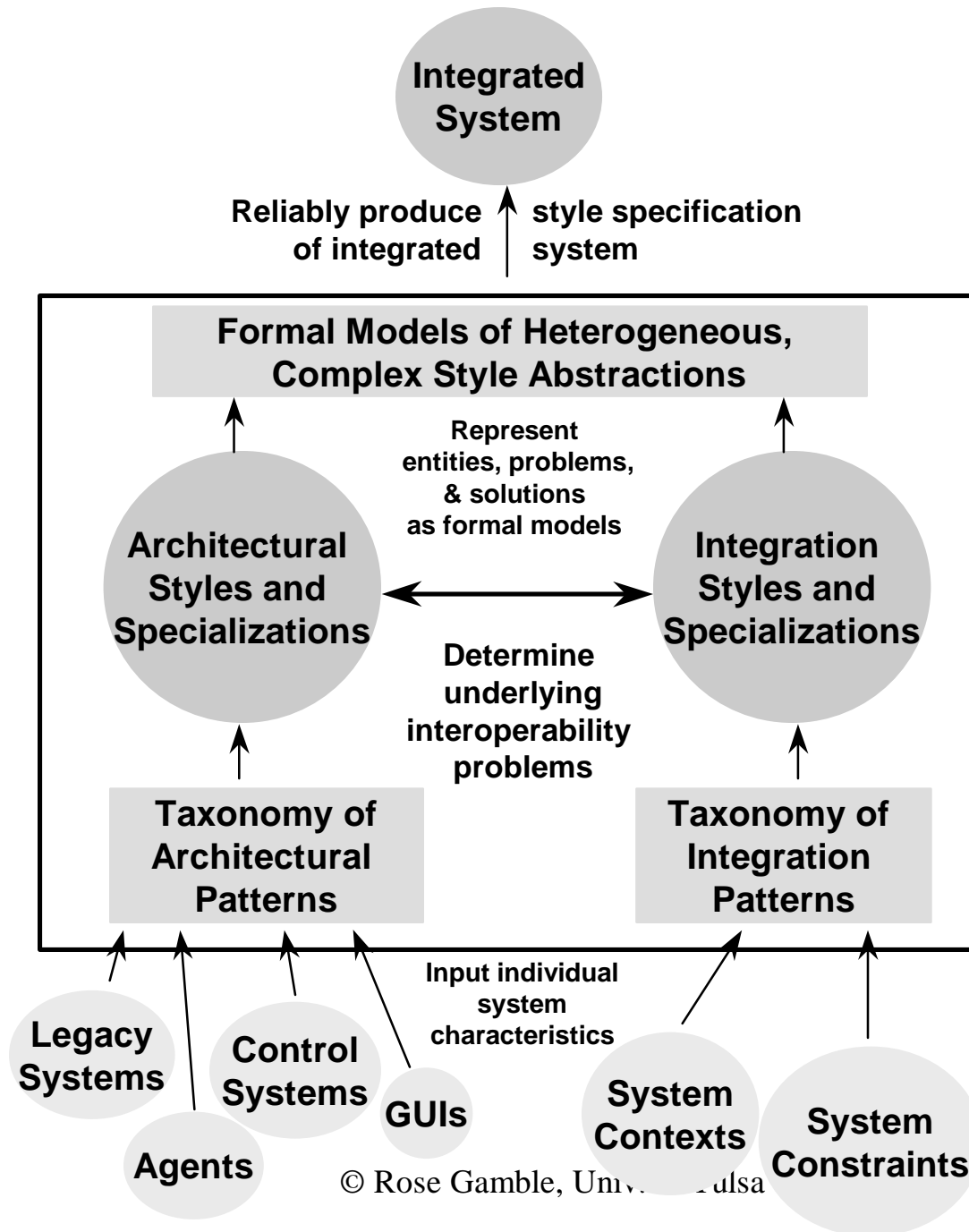
Rose Gamble
University of Tulsa

Approach

- Formal modeling and analysis
 - architectural styles
 - integration styles
- Experimental platform for interoperability analysis
- Taxonomy of common abstractions and specializations
- Extend the foundation to multi-agent architectures
 - configure agents to communicate with legacy systems
 - provide agents with architectural integration strategies to conduct dynamic integration with other systems

Architectural Style Analysis

- allows quick abstraction of software module characteristics
 - common characteristics can be defined, modeled, and compared
 - aids in predicting module interoperability conflicts
- provides a mechanism for choosing most interoperable modules
 - elevates integration to style level
 - allows integration to be an early design decision

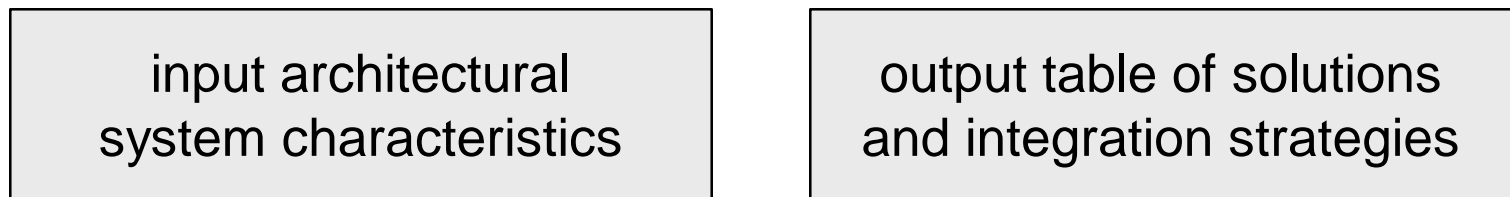


Style Characteristics

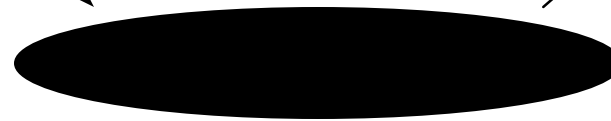
Characteristic Category	Event-Based	Pipe & Filter	Main/ Subroutine	Object Oriented
Components	Methods that signal events <ul style="list-style-type: none"> • Access control • Access protocol 	Filters <ul style="list-style-type: none"> • Filter specialization 	Procedures	Instantiated abstract data types <ul style="list-style-type: none"> • Object sharing
Component Identity	Named or not named	Not named	Named	Named
Connectors	Event Broadcast <ul style="list-style-type: none"> • Event definition • Event representation • Event handling • Parameter passing • Message delivery • Invocation 	Pipes <ul style="list-style-type: none"> • Pipe specialization 	Procedure Call <ul style="list-style-type: none"> • Invocation 	Message Passing <ul style="list-style-type: none"> • Composite object handling
Output Production	One-step	Incremental	One-step	One-step
Control Structure	Decentralized <ul style="list-style-type: none"> • Method of communication 	Data streams	Single thread	Decentralized <ul style="list-style-type: none"> • Control flow
Data Storage	Common store, data with events	Data streams	Global source	Hidden and distributed
Data Representation	Appl. dependent	Appl. dependent	Appl. dependent	Appl. dependent

Experimental Platform

Netscape Browser

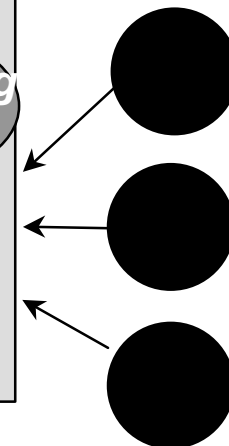
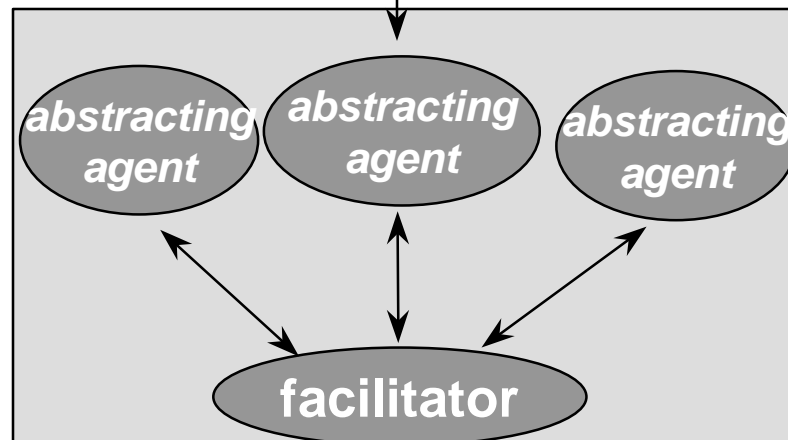


Java server at Univ. of Tulsa Site



Formal Models

Intelligent agent system using KQML with embedded KBS



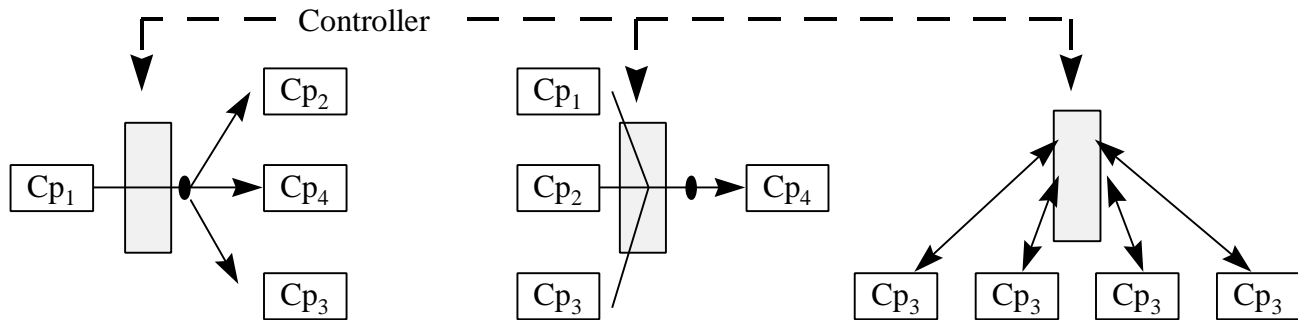
Event System Integrated with Main/Subroutine System

Characteristic	System 1	System 2	Solution	Strategies
Components	method	procedure	Relate components in the systems	Translator or Controller
Component Identity	named	named	Handle name clashes	Developer
Connectors • Invocation	Event Broadcast explicit	Procedure Call implicit	Handle mismatch in the methods of invocation	Translator or Controller
Output Production	one-step	one-step	No interoperability problems	
Control structure	decentralized	single thread	Coordinate control	Broker
Data Storage	common	global	No interoperability problems	
Data Representation	format 1	format 2	Change one system's format	Translator
Final result			Translator, Controller, Developer	

Toward Integration Styles

- Influenced by
 - Initial integration patterns definition
 - wrapper, broker, repository, & work-flow manager
 - Design patterns research
- High-level classification
- Taxonomy goal
 - property sharing integration patterns
 - combination integration patterns

Example Integration Style

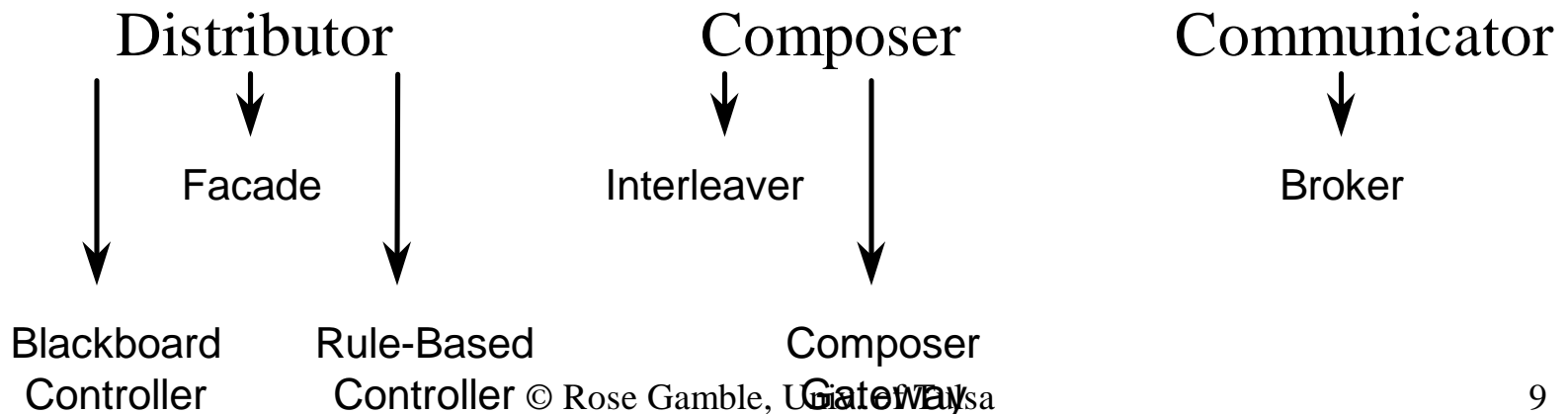


Controller selects between possible components to pass information

Controller selects what is passed from multiple inputs

Controller communicates globally with all components

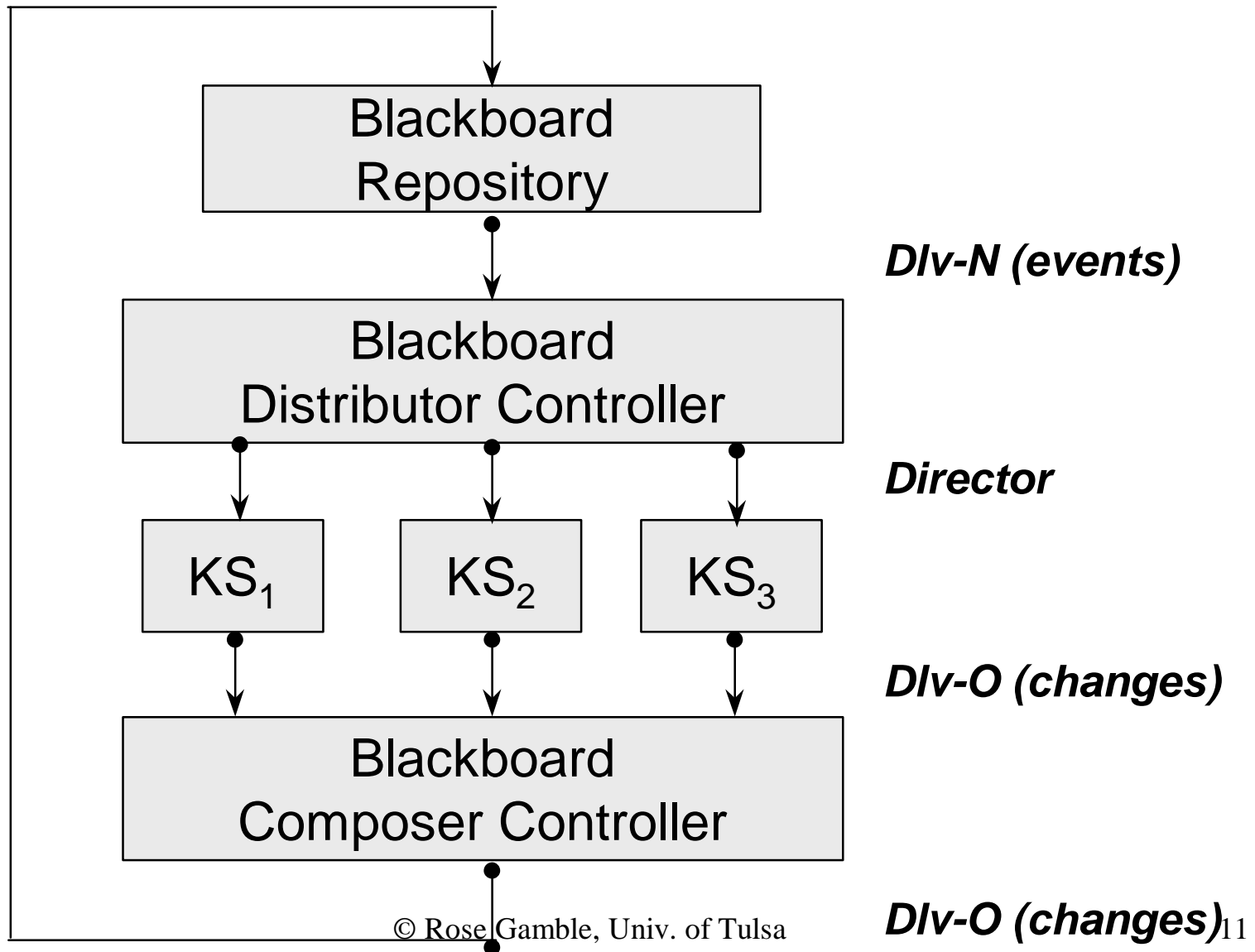
Potential Taxonomy



Formal Modeling

- Provides a consistent representation and analysis mechanism
 - architectural styles
 - integration styles
 - taxonomy of style abstractions and properties
 - overall system architecture

Blackboard Architectural Style



Formal Models of Component and Specialization

Distributor Controller

indata: ALLDATA outports: set CID alphabet: $CID \rightarrow ALLDATA$ states: set STATE start: STATE transitions: $(STATE \times ALLDATA) \leftrightarrow (STATE \times (CID \rightarrow ALLDATA))$
start \in states # outports $> 1 \wedge$ outports = dom alphabet $\forall s_1, s_2: STATE; intran : ALLDATA;$ outtran: $CID \rightarrow ALLDATA;$ $ ((s_1, intran), (s_2, outtran)) \in transitions$ <ul style="list-style-type: none"> • $s_1, s_2 \in states \wedge related(intran, indata)$ $\wedge dom outtran \subseteq outports \wedge outtran \subseteq alphabet$

Distributor Controller State

c: Distributor_Controller curstate: STATE state_vars: DCSTATE strategies: set STRATEGY instate: ALLDATA outstate: $CID \rightarrow ALLDATA$
$curstate \approx (state_vars, strategies) \wedge curstate \in c.state$ $related(instate, c.indata)$ dom outstate $\subseteq c.outports \wedge outstate \subseteq c.alphabet$

Distributor Controller Step

Δ Distributor_ControllerState $c' = c$ $\exists in: ALLDATA; out: CID \rightarrow ALLDATA$ <ul style="list-style-type: none"> • $((curstate, in), (curstate', out)) \in c.transitions$ $\wedge out = compute(in, strategies) \wedge instate' = remove(in, instate)$ $\wedge outstate' = add(out, outstate)$

Blackboard_Distributor_Controller

Distributor_Controller indata: set EVENT outports: set KSID alphabet: $KSID \rightarrow set ACTIVATION$ transition: $(STATE \times EVENT) \leftrightarrow (STATE \times (KSID \rightarrow set ACTIVATION))$
$\forall s_1, s_2: STATE; intran: EVENT;$ outtran: $(KSID \rightarrow set ACTIVATION)$ $ ((s_1, intran), (s_2, outtran)) \in transitions$ <ul style="list-style-type: none"> • intran $\in indata$

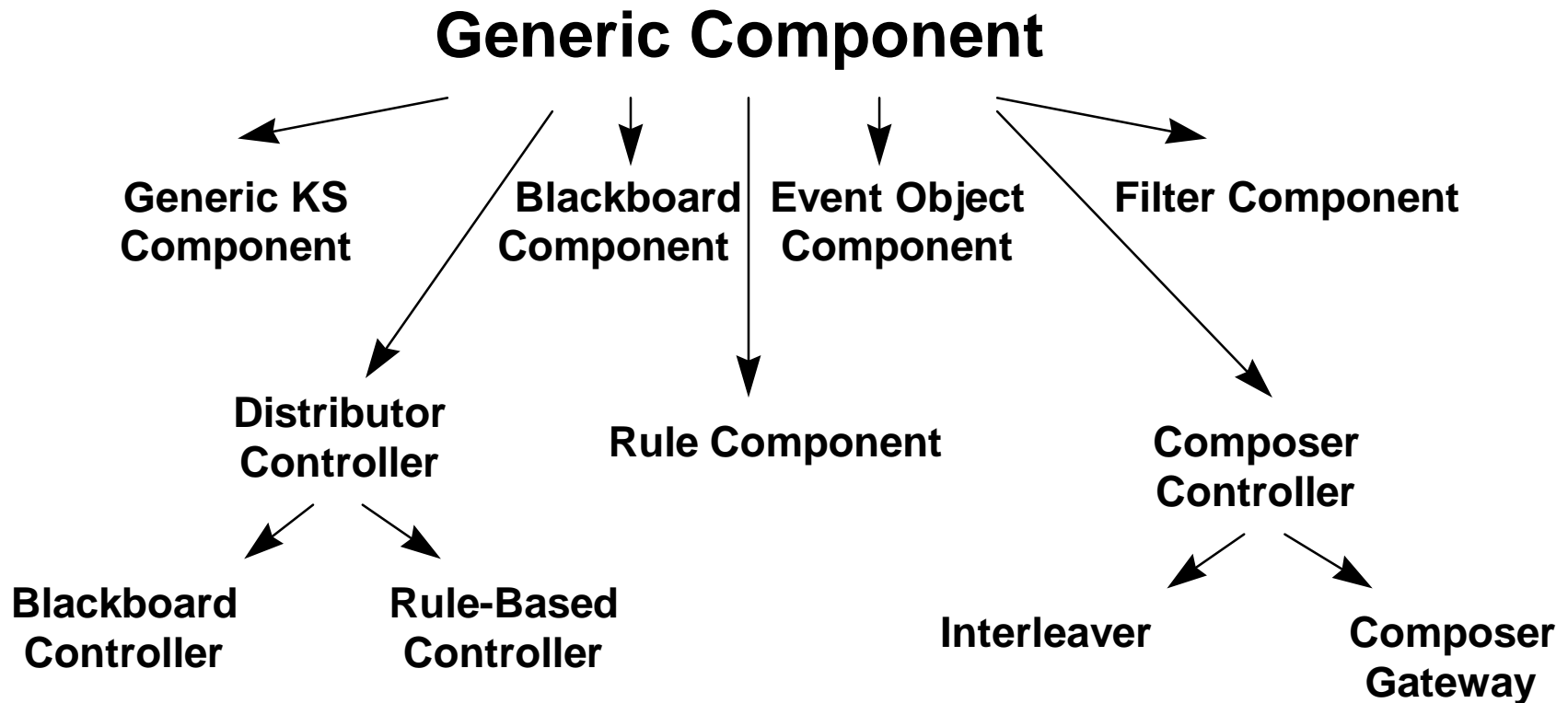
Blackboard_Distributor_Controller_State

c: Blackboard_Distributor_Controller curstate: STATE state_vars: BCSTATE precondition: set PRECONDITION strategies: set STRATEGY instate: seq EVENT outstate: $KSID \rightarrow set ACTIVATION$
$curstate = (state_vars, strategies, precondition) \wedge$ $curstate \in c.state$ ran instate $\subseteq c.indata \wedge$ dom outstate $\subseteq c.outports \wedge outstate \subseteq c.alphabet$

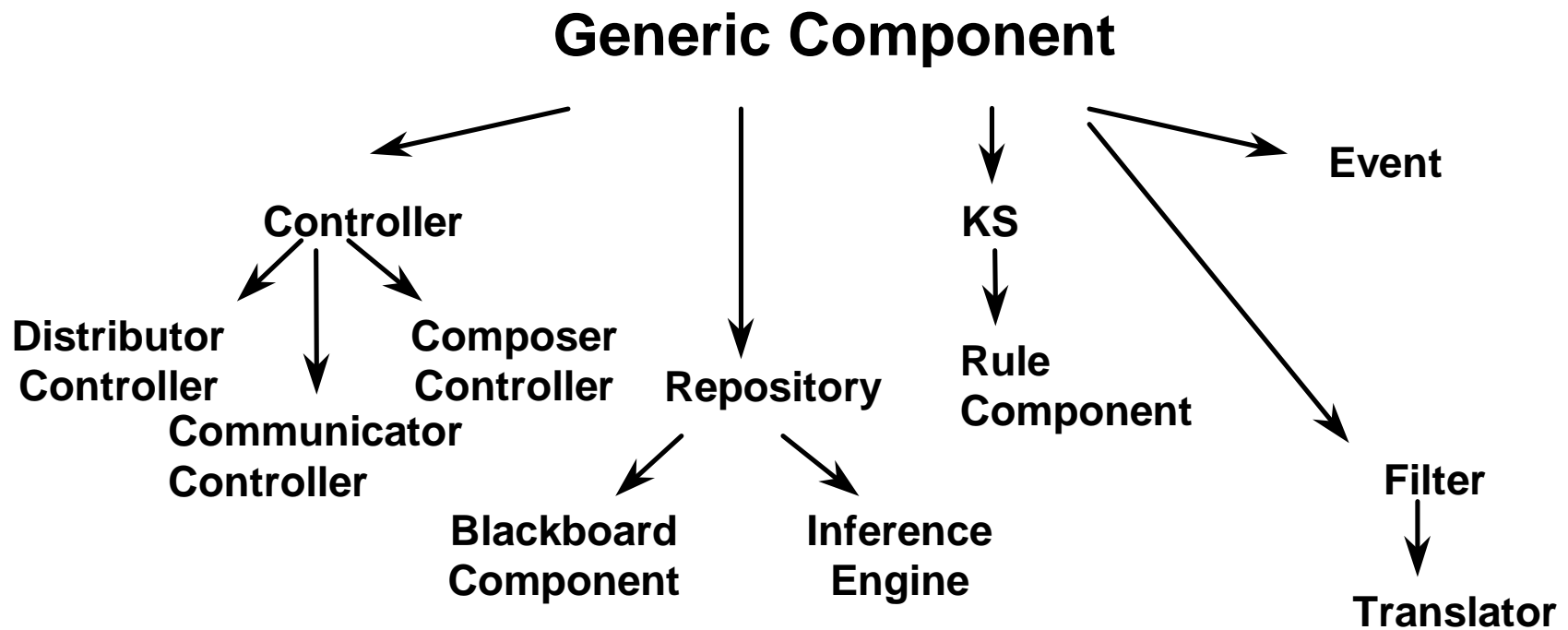
Blackboard_Distributor_Controller_Step

$\Delta BB_ControllerState$ $c = c'$ $\exists in: EVENT; out: KSID \rightarrow set ACTIVATION$ $ in = head instate$ <ul style="list-style-type: none"> • $(curstate, in), (curstate', out) \in c.transitions \wedge$ $out = determine(in, strategies, precondition) \wedge$ $instate' = tail instate \wedge outstate' = outstate \cup out$
--

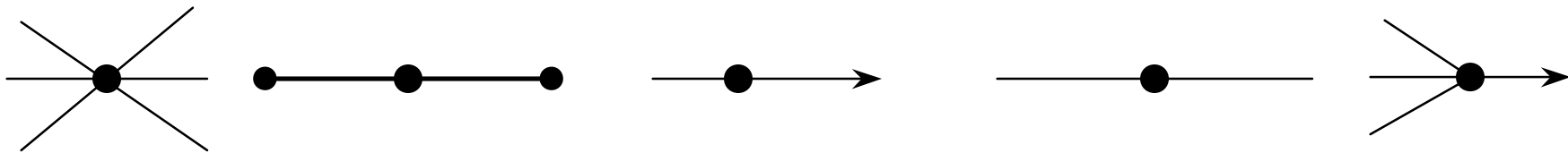
Taxonomy of Components and Connectors



A Better Taxonomy



Specialization of Connectors



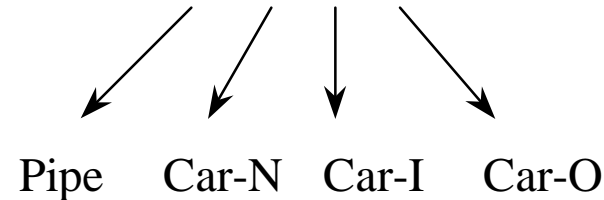
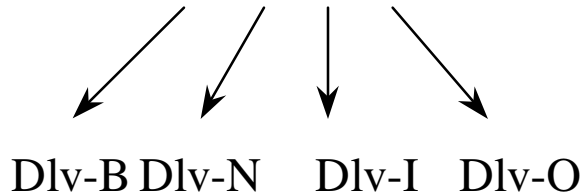
Distributor:
Collects and distributes data to multiple components

Delivery: Point-to-point connector that transmits discrete increments of ordered data

Director: A point-to-point connector that transforms discrete increments of unordered data

Carrier : Point-to-point connector that carries a continuous stream of ordered data

Dispatcher:
Collects and focuses data to single component



Conflict Analysis Potential

- Consistent framework for formal representation of style abstractions
 - allows for reliable determination of interoperability problems
- Elevating integration to style level
 - incorporates potential solutions directly into analysis
 - consistent representation provides more complete integration system analysis
- Taxonomy of style abstractions
 - provides more detail with respect to characteristics, conflicts, and solutions