

## Classifying Interoperability Conflicts



L. Davis      D. Flagg      R. Gamble      C. Karatas

Dr. R.F. Gamble, Director  
Software Engineering & Architecture Team  
Department of Math & Computer Science  
University of Tulsa

[gamble@utulsa.edu](mailto:gamble@utulsa.edu)  
[www.seat.utulsa.edu](http://www.seat.utulsa.edu)

## Component-Based Software Engineering (CBSE)

- Techniques to aid IT implementation
  - Modularizes software functionality
  - Allows systematic construction of software Reduces time to market
  - Increases manageability



## The Motivation for Integration

- Heterogeneous components need principled methodologies for CBSE techniques to provide promised value-added
- Fuse existing COTS and Legacy systems
- Eliminate manual information flow
- Combine redundant processes & offer better service
- Insert COTS components



## Software Architecture

- Describes set of architectural elements that have a particular form
  - processing elements
  - data elements
  - connecting elements
  - behavior expectations
- Incorporates
  - High-level software description
  - Established software styles
  - Principles of design patterns



## Interoperability Problems

- One independent computer program cannot communicate with another
  - Large, complex systems in use
    - Healthcare management systems
    - C2ISR tools
  - Systems have unique information infrastructures
    - Patient records
    - Satellite data
  - Replacing systems not an option
    - Purchasing them very costly
    - Transitioning them is prohibitive



## Software Architecture Analysis

- Characterizes a software component architecturally
- Identifies behavioral mismatches among interacting components
- Provides functional solutions for interoperability conflicts

## The Set of Architecture Characteristics

CHARACTERISTICS	TYPE	DEFINITION	VALUES
Blocking	C	Whether or not the thread of control is suspended.	Blocking, Non-Blocking
Control Structure	A, C	The structure that governs the execution in the system.	Single-Thread, Multi-Thread, Concurrent
Control Topology	A, C	The geometric form control flow takes in a system.	Hierarchical, Star, Arbitrary, Linear
Data Format Difference	A	Describes whether or not data being passed within the application is in the same format.	Yes, No
Data Storage Method	C	How data is stored within a system.	Local, Global, Distributed
Data Topology	A, C	The geometric form data flow takes in a system.	Hierarchical, Star, Arbitrary, Linear
Identity of Components	C	Awareness of other components in the system.	Aware, Unaware
Supported Control Transfer	C	The method supported to achieve control transfer.	Explicit, Implicit, None
Supported Data Transfer	C	The method supported to achieve data transfer.	Explicit, Shared, Implicit-Discrete, Implicit-Continuous, None
Synchronization	A	The level of dependency of a module on another module's control state.	Synchronous, Asynchronous

## Causes of Architecture Conflicts

- Component systems are inhibited from interacting by certain characteristic values
  - Component-component interaction
- Application configuration & coordination requirements impose demands that participating components cannot satisfy
  - Application-component interaction
- Integration solution does not comply with the overall application requirements.
  - Application – Integration interaction

## Problematic Architecture Interactions (PAI)

---

- Definition
  - An interoperability conflict that is predicted through the comparison of architecture interaction characteristics and requires intervention via external services for its resolution
- Structure

*CS.Concurrent(A)@{1} → CT.Hierarchical(B)*

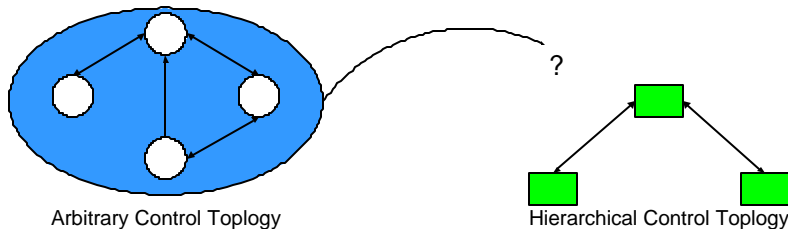
## Classifying Architecture Conflicts

---

- Architecture characteristic values compared
  - Within characteristics
  - Across characteristics
- Conjectured conflicts assessed for similarity and a set resolved
- Set compared to characteristic literature for justification and categorization
  - Control transfer
  - Data transfer
  - Interaction initialization

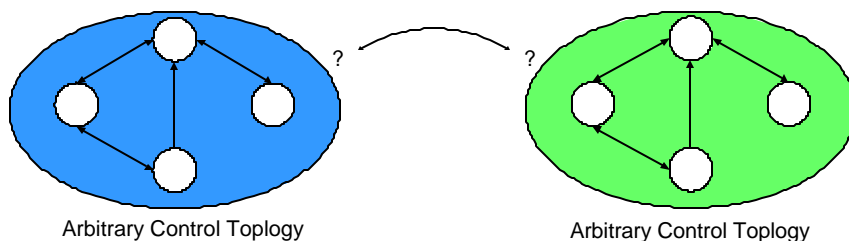
## Restricted Points of Control Transfer

- Control transfer major part of integrating component communication
  - Must agree on the direction of the transfer
- Control passed according to component's assumptions about exchange.
  - Assumptions may be erroneous



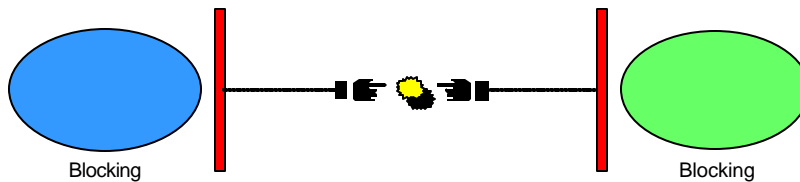
## Unspecified Control Destination

- Direct communication between components often necessary to transfer control.
- If component cannot be transferred control, it cannot be reused in an integration as readily
  - A component's methods may be encapsulated and private
  - A component's interface may not be specified until runtime due to dynamic binding



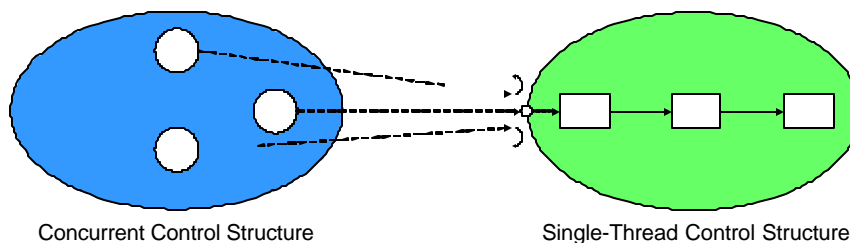
## Inhibited Rendezvous

- Component can rendezvous or handshake to exchange control
- Deadlock can occur if the threads of communicating components are blocked or have failed
- Rendezvous problems also occur when components involved in exchange have different control transfer assumptions
  - different calling structures
  - independent and non-terminating execution that cannot be pre-empted
  - several seats of control



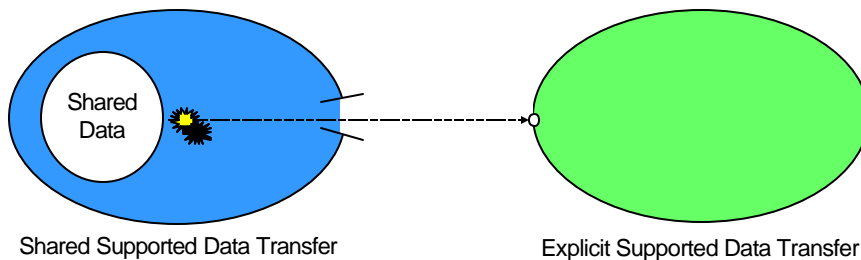
## Multiple, Unsequenced Control Transfers

- Integrating control between components is not just facilitating transfer of communications
- Preserving the intent of the transfers is necessary
- Sequencing between components must be managed



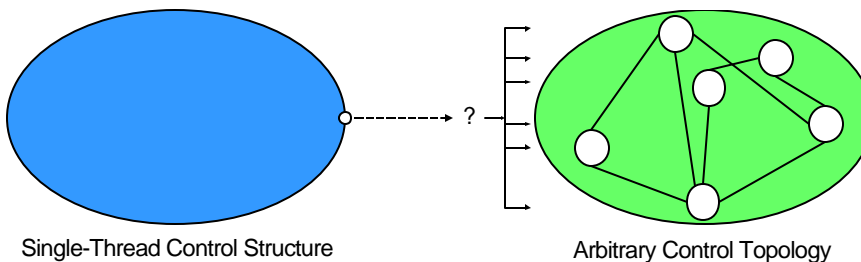
## Restricted Points of Data Transfer

- Correct transfer of data is essential for a usable integrated system.
- Maintenance of data communication transparency guiding motivation for integration architectures
- Inaccessible data hinders data transfer.
  - Accessibility issues emerge when transfer mechanisms of connectors are different or inactive



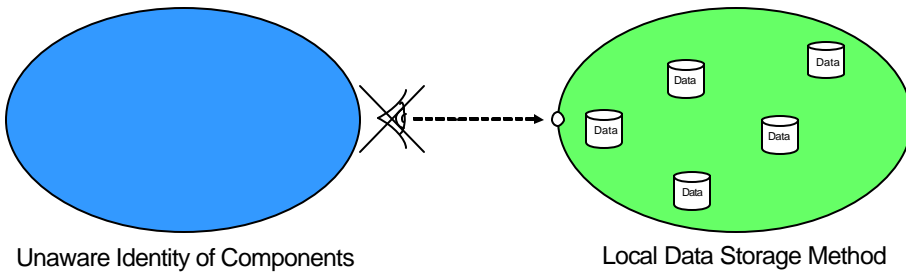
## Unspecified Data Destination

- Direct communication of data common data transfer mechanism
- Components should have similar data exchange assumptions
- Components must have equivalent knowledge about other components with which they wish to communicate
- Problems surface when
  - component's thread expects to directly pass data but cannot find either a waiting thread or the interface address
  - component dynamically bound



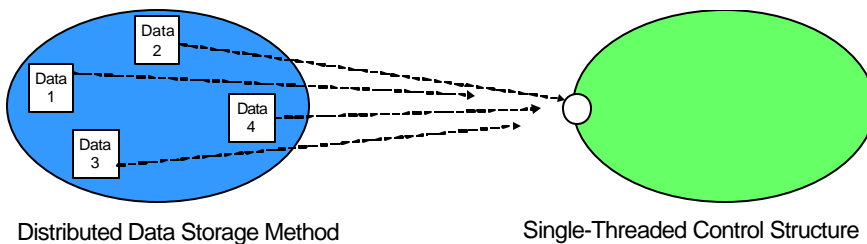
## Unspecified Data Location

- Buffering mechanisms of the various components must be complimentary. D
- Data layouts must be obvious to all components for data transfer
- Data location can be obscured and transmission impeded when
  - components attempt to access private data
  - components wish to communicate data to an object whose data store is dynamically bound



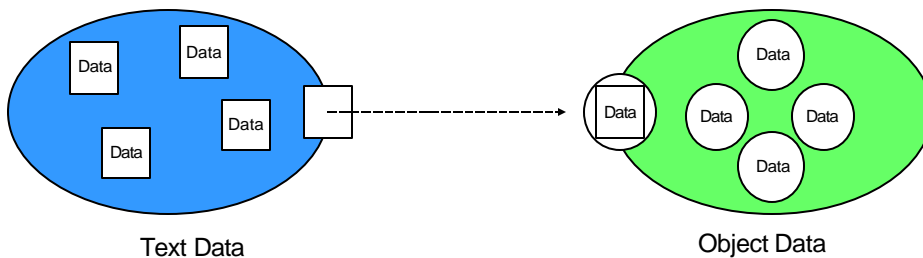
## Inconsistent Data

- Inconsistently communicated data can be permute data content of a communication
- To exchange data, synchronization of the exchange must be maintained
- Data can be corrupted or lost if care is not taken to manage data exchange



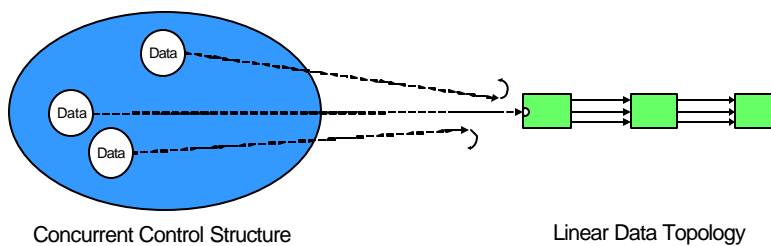
## Invalid Data

- Invalid data most widely recognized conflict
- Data format transparency a pre-requisite for integrating data properly
- Translation and marshalling well understood



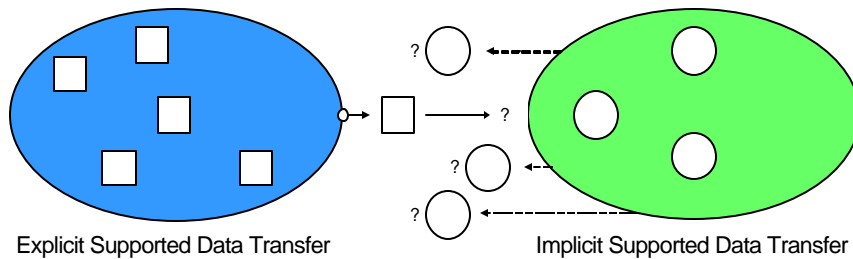
## Multiple, Unsequenced Data Transfers

- Appropriate sequencing of data transfers necessary to guarantee content
  - No guarantee that data received by the sink component was first packet of sequence of communications.
- Disparate data transfer mechanisms are problematic
- Data can be lost at a receiving component that does not embody concurrency control.



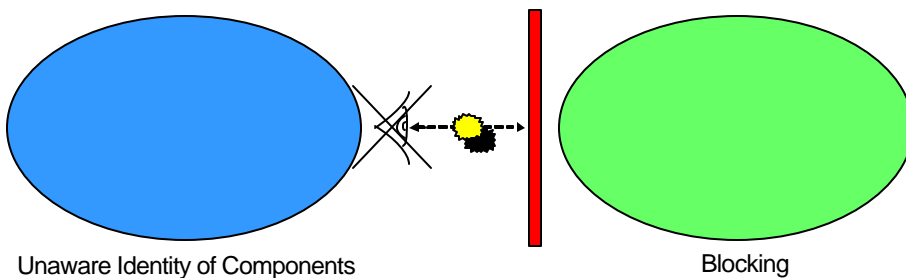
## Mismatched Data Transfer Assumptions

- Component data transfer method determines mode of data communication
- Incompatible mechanisms impede data exchange as mode of transfer gets less direct



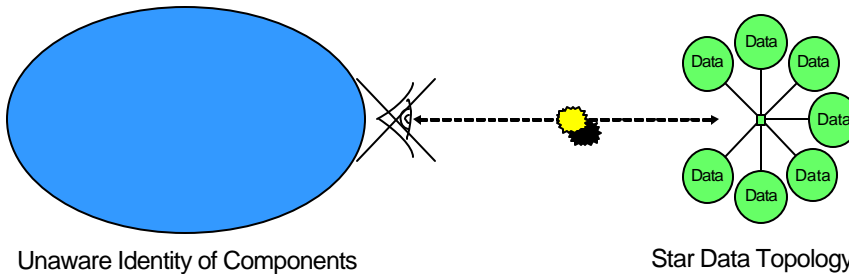
## Uninitialized Control Transfer

- Unrecognizable control transfer modes obstruct communication between participating components
- Components may be unable to pass control
  - Lack awareness of communication partners
  - Lack awareness of other component's connectors
  - Lack push communication capability to and beyond its interface



## Uninitialized Data Transfer

- Unrecognizable data transfer modes obstruct communication between participating components
- Components may be unable to pass data
  - Lack awareness of communication partners
  - Lack awareness of other component's connectors
  - Lack push communication capability to and beyond its interface



## Categorized Architecture Conflicts

- Category 1: Control Transfer
  - 1. Restricted points of control transfer
  - 2. Unspecified control destination
  - 3. Inhibited rendezvous
  - 4. Multiple, unsequenced control transfers
- Category 2: Data Transfer
  - 6. Unspecified data destination
  - 5. Restricted points of data transfer
  - 7. Unspecified data location
  - 8. Inconsistent data
  - 9. Invalid data
  - 10. Multiple, unsequenced data transfers
  - 11. Mismatched data transfer assumptions
- Category 3: Interaction Initialization
  - 12. Uninitialized control transfer
  - 13. Uninitialized data transfer

## Conclusions

---

- Software architecture provides essential design information
- Property assessment & satisfaction can be obtained through
  - Formal means where properties are critical
  - Patterns-based where experience is decisive enough

