# Analysis of Bridges & Wormholes Paper

## Part 2 – Assessment and Improvement of Control

First Delivered: 03 Feb 2026

Last Updated: 03 Feb 2026

Michael M Lee

# Introduction

## Purpose of Analysis Effort

- ✓ Understand the wormhole approach to bridging (1st control, then data)
- ✓ Provide an overview of the approach that makes it more accessible
- Assess the approach's viability (Can it work?)
- Identify constraints and propose improvements

## Key Terms

- HOME (requesting service) domain
- AWAY (providing service) domain
- MX = Model Execution/Software Architecture domain

# When We Last Visited… Bridging Behaviors

## Are these behaviors viable?

**Can the SW Architecture support them?**

**CRP Behavior**

**Synchronous Bridge Scenario**

**Asynchronous Bridge Scenario**

**Announcement Bridge Scenario**

# B&W Allocation of Responsibility to SW Architecture

| Item | Type | Page | Paragraph | Section | SW Architecture Responsibility |
|---|---|---|---|---|---|
| 1 | Data | 3 | 2 | Assumptions & Rationale | Establish instance correspondence across domains |
| 2 | Data | 3 | 3 | Assumptions & Rationale | Support semantic shifts |
| 3 | Control | 3 | 4 | Transfer Vectors | Structure/Type of Transfer Vector |
| 4 | Control | 4 | 4 | Transfer Vectors | Construction, population & propagation of a return event |
| 5 | Control | 5 | 2 | Return Coordinates… | Structure/Type of Return Coordinates |
| 6 | Control | 5 | 5 | Return Coordinates… | Manage fork in control between Home & Away |
| 7 | Control | 5 | 8 | Return Coordinates… | Construction & population of Return Coordinates |
| 8 | Control | 6 | 1 | Domain Interface (CRP) | *Transfer control to Away via an event or invocation |
| 9 | Control | 7 | 6 | Request Wormhole | *Transfer control from Home to Away via a Request Action |
| 10 | Control | 9 | 4 | Return Wormhole | *Transfer control from Away to Home via a Return Action with a Transfer Vector or Return Coordinates |
| 11 | Control | 14 | 3 | Resolving Wormholes | Construction & population of a return event |
| | | | | | |
| | | | | | * = Responsibility implied by wormholes being MX actions |

# References for Implied Responsibilities

| Item | Type | Page | Paragraph | Section | SW Architecture Responsibility | Bridges & Wormholes Reference |
|---|---|---|---|---|---|---|
| **8** | Control | 6 | 1 | Domain Interface (CRP) | *Transfer control to Away via an event or invocation | "When a domain receives control from across a domain boundary, the receiver sees the control as either the arrival of an event from outside the domain or as the invocation of a synchronous service." |
| **9** | Control | 7 | 6 | Request Wormhole | *Transfer control from Home to Away via a Request Action | "Control is transferred from one domain to another by invocation of a wormhole process on an ADFD or SDFD." |
| **10** | Control | 9 | 4 | Return Wormhole | *Transfer control from Away to Home via a Return Action with a Transfer Vector or Return Coordinates | * "If there is an input to the wormhole of type transfer vector, the return will be asynchronous."<br><br>* "If there is an input of type return coordinate, the return will be synchronous." |

# Viability of a Synchronous Bridge Scenario?

| Item | Type | Page | Paragraph | Section | SW Architecture Responsibility |
|---|---|---|---|---|---|
| 1 | Data | 3 | 2 | Assumptions & Rationale | Establish instance correspondence across domains |
| 2 | Data | 3 | 3 | Assumptions & Rationale | Support semantic shifts |
| 3 | Control | 3 | 4 | Transfer Vectors | Structure/Type of Transfer Vector |
| 4 | Control | 4 | 4 | Transfer Vectors | Construction, population & propagation of a return event |
| 5 | Control | 5 | 2 | Return Coordinates… | Structure/Type of Return Coordinates |
| 6 | Control | 5 | 5 | Return Coordinates… | Manage fork in control between Home & Away |
| 7 | Control | 5 | 8 | Return Coordinates… | Construction & population of Return Coordinates |
| 8 | Control | 6 | 1 | Domain Interface (CRP) | *Transfer control to Away via an event or invocation |
| 9 | Control | 7 | 6 | Request Wormhole | *Transfer control from Home to Away via a Request Action |
| 10 | Control | 9 | 4 | Return Wormhole | *Transfer control from Away to Home via a Return Action with a Transfer Vector or Return Coordinates |
| 11 | Control | 14 | 3 | Resolving Wormholes | Construction & population of a return event |
| | | | | | |
| | | | | | * = Responsibility implied by wormholes being MX actions |

## How Scenario Works

1. A Request Wormhole action executes in **Home** which passes control to MX [#9] which then…
2. Suspends the Request Wormhole Action [#6], creates a Return Coordinate [#5, #7] and initiates activation of the associated CRP activity in **Away** [#8] which stores the Return Coordinate and executes until…
3. A Synchronous Return Wormhole action executes in the CRP Activity [#10] in **Away** which has retrieved and passed the Return Coordinate and any return data to MX which causes it to…
4. Resume the Request Wormhole action in **Home** [#6].

**Viable?**

Consistent with published SW Architecture responsibilities.

# Viability of an Asynchronous Bridge Scenario?

| Item | Type | Page | Paragraph | Section | SW Architecture Responsibility |
|---|---|---|---|---|---|
| 1 | Data | 3 | 2 | Assumptions & Rationale | Establish instance correspondence across domains |
| 2 | Data | 3 | 3 | Assumptions & Rationale | Support semantic shifts |
| 3 | Control | 3 | 4 | Transfer Vectors | Structure/Type of Transfer Vector |
| 4 | Control | 4 | 4 | Transfer Vectors | Construction, population & propagation of a return event |
| 5 | Control | 5 | 2 | Return Coordinates… | Structure/Type of Return Coordinates |
| 6 | Control | 5 | 5 | Return Coordinates… | Manage fork in control between Home & Away |
| 7 | Control | 5 | 8 | Return Coordinates… | Construction & population of Return Coordinates |
| 8 | Control | 6 | 1 | Domain Interface (CRP) | *Transfer control to Away via an event or invocation |
| 9 | Control | 7 | 6 | Request Wormhole | *Transfer control from Home to Away via a Request Action |
| 10 | Control | 9 | 4 | Return Wormhole | *Transfer control from Away to Home via a Return Action with a Transfer Vector or Return Coordinates |
| 11 | Control | 14 | 3 | Resolving Wormholes | Construction & population of a return event |
| | | | | | |
| | | | | | * = Responsibility implied by wormholes being MX actions |

## How Scenario Works

1. A Request Wormhole action executes in **Home** which passes control to MX [#9] which then…
2. Returns control to the Request Wormhole action [OOA] and initiates activation of the associated CRP activity in **Away** [#8] which stores the Transfer Vector and executes until…
3. An Asynchronous Return Wormhole action executes in the CRP Activity [#10] in **Away** which retrieves and passes the Transfer Vector and any return data to MX which causes it to…
4. Construct a "Populated Return Vector" event [#3, #4] and post that event in **Home.** [#11, #4]

**Viable?**

Consistent with published SW Architecture responsibilities.

# Viability of an Announcement Bridge Scenario?

Announcement Bridge scenario is just a subset of the Asynchronous Bridge scenario.

**Viable?**

Consistent with published SW Architecture responsibilities.

# Control Responsibilities Used in Scenarios

| Item | Type | Page | Paragraph | Section | SW Architecture Responsibility |
|---|---|---|---|---|---|
| 1 | Data | 3 | 2 | Assumptions & Rationale | Establish instance correspondence across domains |
| 2 | Data | 3 | 3 | Assumptions & Rationale | Support semantic shifts |
| ✓ 3 | Control | 3 | 4 | Transfer Vectors | Structure/Type of Transfer Vector |
| ✓ 4 | Control | 4 | 4 | Transfer Vectors | Construction, population & propagation of a return event |
| ✓ 5 | Control | 5 | 2 | Return Coordinates… | Structure/Type of Return Coordinates |
| ✓ 6 | Control | 5 | 5 | Return Coordinates… | Manage fork in control between Home & Away |
| ✓ 7 | Control | 5 | 8 | Return Coordinates… | Construction & population of Return Coordinates |
| ✓ 8 | Control | 6 | 1 | Domain Interface (CRP) | *Transfer control to Away via an event or invocation |
| ✓ 9 | Control | 7 | 6 | Request Wormhole | *Transfer control from Home to Away via a Request Action |
| ✓ 10 | Control | 9 | 4 | Return Wormhole | *Transfer control from Away to Home via a Return Action with a Transfer Vector or Return Coordinates |
| ✓ 11 | Control | 14 | 3 | Resolving Wormholes | Construction & population of a return event |
| | | | | | |
| | | | | | * = Responsibility implied by wormholes being MX actions |

# When We Last Visited… Matching Requests and CRPs

**Request Behavior**

**CRP Behavior**

**What are the constraints on matching?**

**Can these constraints be reduced?**

**Matching Request with CRP**

# Matching Wormholes with Control Reception Points

In order to identify the CRP that corresponds to each wormhole, it is helpful to bring the definitions of all request wormholes from one domain together with all CRPs of the other domain. One way of doing this is shown in the following table:

| Domain | Ident | Meaning | Inputs | Synch outputs | Asynch return | Asynch outputs |
|---|---|---|---|---|---|---|
| Application | W1 | Get magnet current | Magnet ID | current | -- | NA |
| Application | W2 | Move robot | x, y, θ, Robot ID | -- | R2: (Robot ID) | none |
| Application | W3 | Position slider | Slider ID, position | delta | -- | NA |
| | | | | | | |
| PIO | S1 | Read analog input point | AIP ID, return coordinate | scaled value | -- | NA |
| PIO | AIOP1 | Set analog input point | AIP ID, setpoint | -- | -- | NA |
| PIO | AOP1 | Set analog output point | AOP ID, setpoint | -- | -- | NA |
| PIO | AIP3 | Iterate to setpoint | AIP ID, setpoint, return coordinate | discrepancy | -- | NA |
| PIO | TAG10 | Three-axis move | TAG ID, x setpoint, y setpoint, z setpoint, notification | -- | notification | none |

# Constraints on Matching Behavior

**HOME Request**
w/o Transfer Vector
and w/outputs
(Sync)

Only Matches

**AWAY CRP**
w/Return Coordinates
and Sync Return
(Sync)

**HOME Request**
w/Transfer Vector
w/ or w/o outputs
(Async)

Only Matches

**AWAY CRP**
w/Transfer Vector
and Async Return
(Async)

**HOME Request**
w/o Transfer Vector
and w/o outputs
(Announce)

Only Matches

**AWAY CRP**
w/o Transfer Vector
or Return Coordinates
And w/o any Return
(Announce)

**NOTE**: I/O Data must also match

# Constraints and Proposed Improvements

## Bridges & Wormholes v1

- 1 Request Wormhole to address:
  - synchronous,
  - asynchronous and
  - announcement behavior
- 2 Return Wormholes to address:
  - synchronous and
  - asynchronous behavior

## Constraints

- A CRP can only provide 1 behavior based on Return Wormhole it uses
- Home must match CRP behavior
- Hard to determine Home's desired behavior from the Request in AL (Transfer Vector & Outputs)

## Bridges and Wormholes v2

- 2 Request Wormholes, Sync and Async
- New data type, Return Mechanism w/subtypes:
  - Transfer Vector, Return Coordinates, Neither
- 1 Return Wormhole
  - CRP provides stored Return Mechanism
  - MX uses subtype to provide requested behavior

## Improvements

- A CRP can provide both sync and async behavior based on subtype of Return Mechanism
- Home does not need to match CRP behavior
- Easy to determine Home's desired behavior from the Request in AL

# When We Last Visited… Concurrency

Understanding and modeling concurrency correctly is essential to building models that can successfully execute in multiple software architectures.

# Aspects of Concurrency when Modeling a Bridge Request

**Behavior ➔ Synchronous or Asynchronous for HOME**

- A synchronous bridge request blocks the request action in HOME while the service is being provided
- An asynchronous bridge request, does not block the request action in HOME while the service is being provided.

**Service Completion Notification ➔ HOME knows requested service is completed**

- A synchronous bridge request always gets an implicit service completion notification by virtue of the fact that it gets unblocked when the service is completed.
- An asynchronous bridge request can choose to get a completion notification or not, in the form of a Return Event, depending on whether or not the HOME domain needs one for control or data purposes.

# Clarity on Concurrency when Modeling a Bridge Request

# Lack of Constraints on Matching Behavior

**HOME**
**Synch Request**
w/Return Coordinates

**HOME**
**Async Request**
w/Transfer Vector

**HOME**
**Async Request**
w/Neither

**AWAY CRP**
w/Single Return Type
and Return Mechanism

**NOTE**: I/O Data must also match

# Providing Both Synchronous and Asynchronous Behavior

- A CRP is an activity in AWAY that provides a requested service.
- A CRP can be activated by an event or an invocation.
- A CRP will be provided any input data and a Return Mechanism on activation.
- A CRP has some amount of work to do to provide the requested service and may or may not have data to return to HOME.
- When that work is done and any return data is available, the CRP invokes the Return wormhole, providing the Return Mechanism and any return data to the MX.
  - If the Return Mechanism is a Return Coordinate, the data (if any) and control can be returned to the Sync Request action using the provided Return Coordinates
  - If the Return Mechanism is a Transfer Vector, the data (if any) can be returned to the HOME domain via the Return Event created from the Transfer Vector. Control has already been returned to the Async Request action earlier.
  - If the Return Mechanism is Neither, there is nothing to do. Control has already been returned to the Async Request action earlier and HOME doesn't require a notification or data.

# Synchronous Bridge Scenario

1. A Sync Request Wormhole action executes in **Home** which suspends the Wormhole action and causes MX to…
2. Initiate activation of the associated CRP activity (w/input data + Return Mechanism = Return Coordinates) in **Away**
3. The CRP activity executes until, a New Return Wormhole action executes in the CRP Activity in **Away** , providing a Return Mechanism which causes MX to…
4. Recognize that the Return Mechanism subtype is a Return Coordinates and…
5. Resume the Request Wormhole action in **Home** (w/requested return data, if any).

Red = New

# Asynchronous Bridge Scenario w/Notification

1. An Async Request Wormhole action executes in **Home** which suspends the Wormhole action and causes MX to…
2. Initiate activation of the associated CRP activity (w/input data + Return Mechanism = Transfer Vector) in **Away** and resume the Request Wormhole action in **Home**.
3. The CRP activity executes until a New Return Wormhole action executes in the CRP Activity in **Away** , providing a Return Mechanism which causes MX to…
4. Recognize that the Return Mechanism is a Transfer Vector and…
5. Post the "Populated Return Vector" event in **Home**

Red = New

# Asynchronous Bridge Scenario w/o Notification or Data

1. An Announce Request Wormhole action executes in **Home** which suspends the Wormhole action and causes MX to…
2. Initiate activation of the associated CRP activity (w/input data + Return Mechanism = Neither ) in **Away** and resume the Request Wormhole action in **Home**.
3. The CRP activity executes. If a New Return Wormhole action is invoked in the CRP Activity in **Away**, it provides a Return Mechanism which causes MX to…
4. Recognize that the Return Mechanism is a Neither and does nothing.

Red = New

# Viability of a  New Bridge Scenarios?

**Viable?**

As viable as the Bridging & Wormholes v1 Approach

# Summary

## Results of Analysis of B&W Control

- Focusing on the types of behavior helped clarify
  - matching HOME and AWAY and
  - the constraints behavior placed on the matching.
- Providing step-by-step diagrams of the scenarios helped
  - understand how the behaviors worked and
  - assess the approach's viability.
- Considering the concurrency of execution between Home and Away was more to the point than considering execution to be synchronous or asynchronous.

## Possible Benefit of Proposed B&W Improvement

- Removing the constraints behavior places on the matching HOME requests with AWAY services would improve the usefulness of the AWAY services.

But… there is still (much) more work to be done.

# Questions? Comments?? Suggestions???

# Challenges & Questions w/Bridges & Wormholes Approach

- Need to handle a request wanting different return events based on results of service invocation (e.g., Fail or Succeed) PLUS can this map to a synchronous behavior?
- Generic, table-based bridge code is allowed by B&W. How about Project/Bridge specific code as K-C does?
- Implied in B&W is a granularity of a class instance to initiate a CRP or receive a response event. What about Class state models?
- Why aren't synch requests w/o data not allowed to block until completion in B&W?
- Can you have a mismatch between Home & Away wrt the number of return parameters as long as Away provides more that Home wants?

# Matching Request⇔CRP Possibilities with B&W/v1

AWAY/Control Reception Point

HOME/ Request

| | Announce | Synch w/Data | Synch w/o Data | Asynch w/Data | Asynch w/o Data |
|---|---|---|---|---|---|
| Announce | OK | | | | |
| Synch w/Data | | OK | | | |
| Synch w/o Data | | | | | |
| Asynch w/Data | | | | OK | |
| Asynch w/o Data | | | | | OK |

# Constraints with this Matching

## Request Wormhole (Home)

- ID
- Meaning
- Data items to be sent
- Data items to be returned
- Transfer Vector (optional)

## Control Reception Point (Away)

- ID
- Meaning
- Data items to be sent
- Data items to be returned
- Return information expected, one of:
  - Transfer Vector
  - Return Coordinates
  - Nothing

**Request Wormhole with Transfer Vector**

- CRP must have Transfer Vector specified AND an Asynchronous Return

**Request Wormhole without Transfer Vector and with outputs required**

- CRP must have Return Coordinates specified AND a Synchronous Return

**Request Wormhole without Transfer Vector and with no outputs required**

- CRP must have neither Return Coordinates nor Transfer Vector specified AND no Return

# Matching Request⇔CRP Possibilities with B&W/v2

AWAY/Control Reception Point

HOME/ Request

| | Announce | V1-Synch w/Data | V1-Synch w/o Data | V1-Asynch w/Data | V1-Asynch w/o Data |
|---|---|---|---|---|---|
| **Announce** | OK | | Ok'ish | | OK |
| **Synch w/Data** | | OK | | OK | |
| **Synch w/o Data** | | | OK | | OK |
| **Asynch w/Data** | | OK | | OK | |
| **Asynch w/o Data** | | | OK | | OK |

# Benefits of Bridges and Wormholes v2

## B&W v1

AWAY/Control Reception Point

HOME/ Request

| | Announce | Synch w/Data | Synch w/o Data | Asynch w/Data | Asynch w/o Data |
|---|---|---|---|---|---|
| Announce | OK | | | | |
| Synch w/Data | | OK | | | |
| Synch w/o Data | | | | | |
| Asynch w/Data | | | | OK | |
| Asynch w/o Data | | | | | OK |

## B&W v2

AWAY/Control Reception Point

HOME/ Request

| | Announce | V1-Synch w/Data | V1-Synch w/o Data | V1-Asynch w/Data | V1-Asynch w/o Data |
|---|---|---|---|---|---|
| Announce | OK | | Ok'ish | | OK |
| Synch w/Data | | OK | | OK | |
| Synch w/o Data | | | OK | | OK |
| Asynch w/Data | | OK | | OK | |
| Asynch w/o Data | | | OK | | OK |

- There still needs to be an agreement between Home and Away on data
- Allowing a CRP to provide both synchronous and asynchronous behavior
  - is possible,
  - makes a CRP more useful and
  - reduces the need to "negotiate interfaces".

# Matching Request⇔CRP Possibilities with B&W/v2

AWAY/Control Reception Point

HOME/ Request

| | Announce | CRP w/Data | CRP w/o Data |
|---|---|---|---|
| Announce | OK | | Ok'ish |
| Synch w/Data | | OK | |
| Synch w/o Data | | | OK |
| Asynch w/Data | | OK | |
| Asynch w/o Data | | | OK |