DARPATECH 2000

ISO

Dynamic Assembly for System Adaptability, Dependability and Assurance (DASADA)

The software revolution requires dynamic gauges

But gauges made it happen

And an ability to dynamically update and use models

John Salasin, PhD
Information Systems Office

DARPA

DARPATECH 2000 ISO

Problem

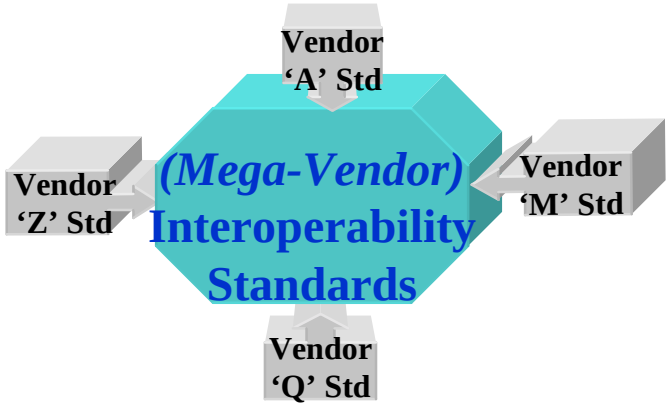
"Software glitches leave Navy Smart Ship dead in the water" (GCN July 13, 1998)

"Glitch in combat systems software knocks out weapons capability" (The Virginian-Pilot, July 8, 1998)

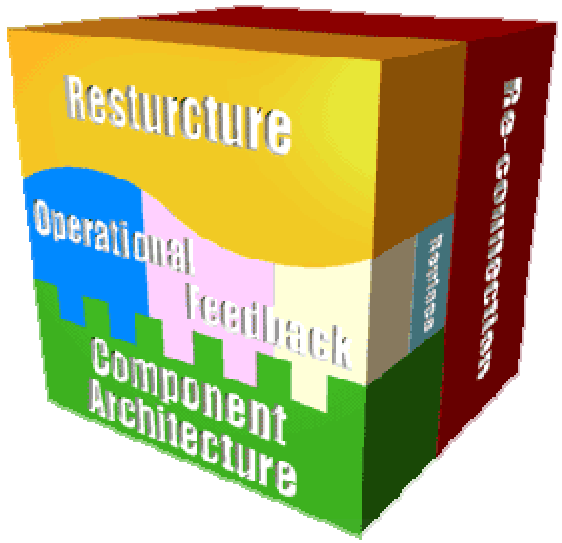
We can't completely model today's complex systems. Therefore, we can't: Understand them; Predict them; Control them; Automatically compose or adapt them.

Why DASADA

Industry hasn't done it, DoD needs it



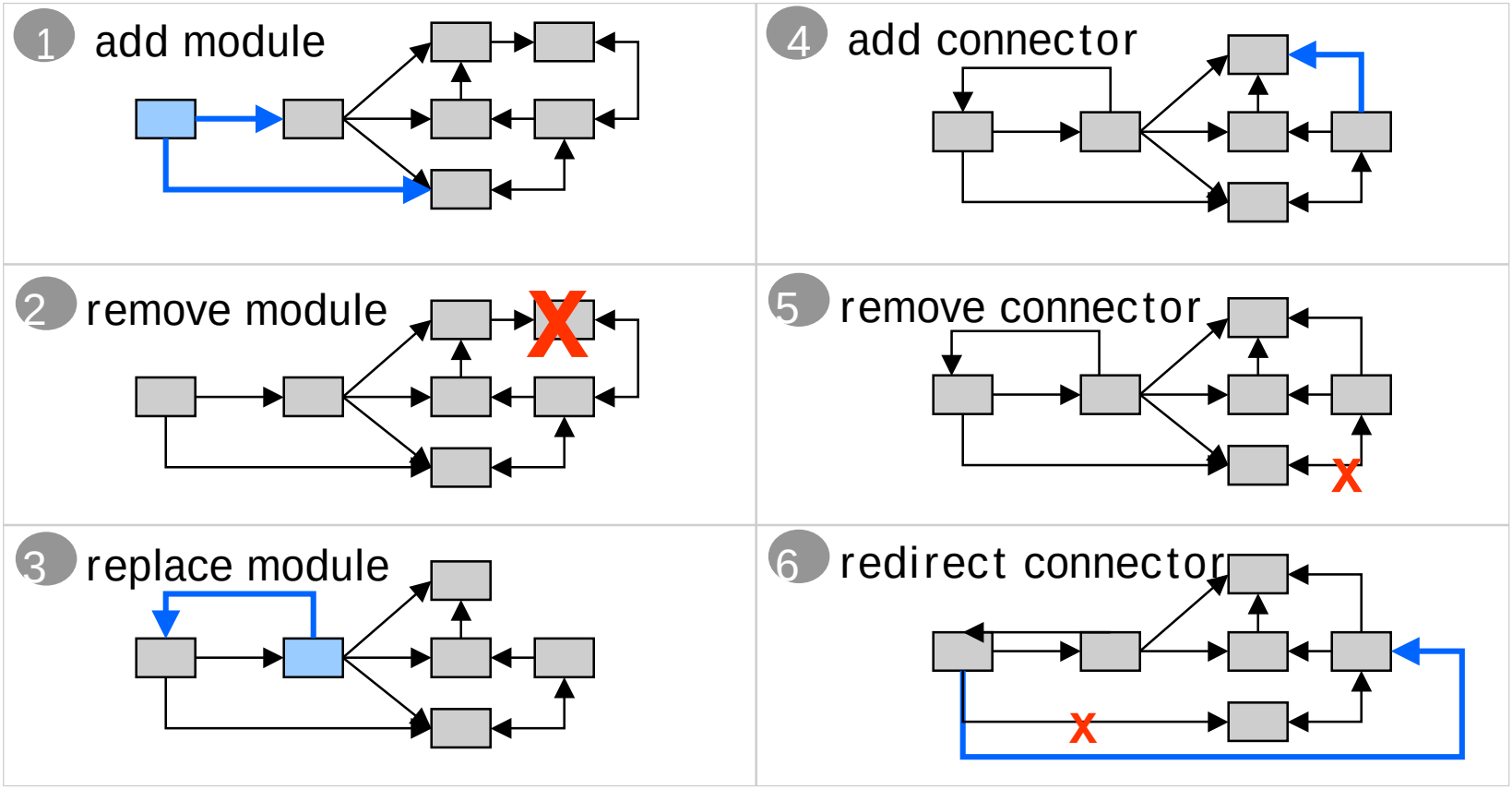
Software Fix(s), re-connector(s), glue and gauges



Predictable composition is key to reduced cycle time

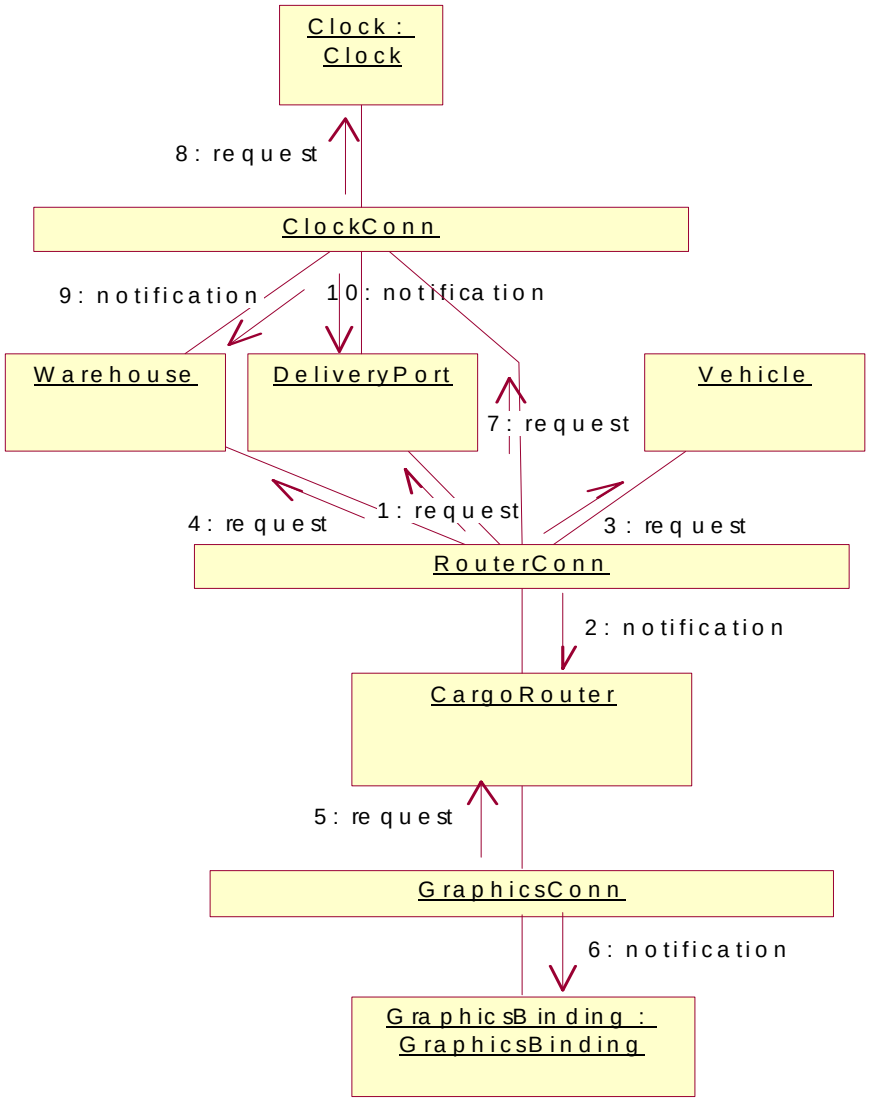
- Dynamically assemble, reconfigure, and evolve systems
- Easily introduce new components to add functionality
- Adaptively and dynamically scale systems
- Continuously upgrade components

Transformation Based Architecture



Assess suitability before, during, and after architecture transformations

Software Architecture



Components

Connectors

Events

Constraint rules

Gauges

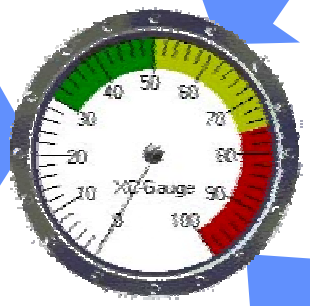
Gauges Are Central

Gauges assess suitability before, during, and after software architecture transformations.

Continual Coordination

- **Architecture Conformance**
- **Composability Wrapper**


Connectivity



Continual Design

- **Functional Similarity**
- **Content Similarity**


Example: Constraint-based Gauge

Continual Validation


Evolution and Integration Command Center

- **Runtime Event Monitoring**
- **Connectivity Gauges**

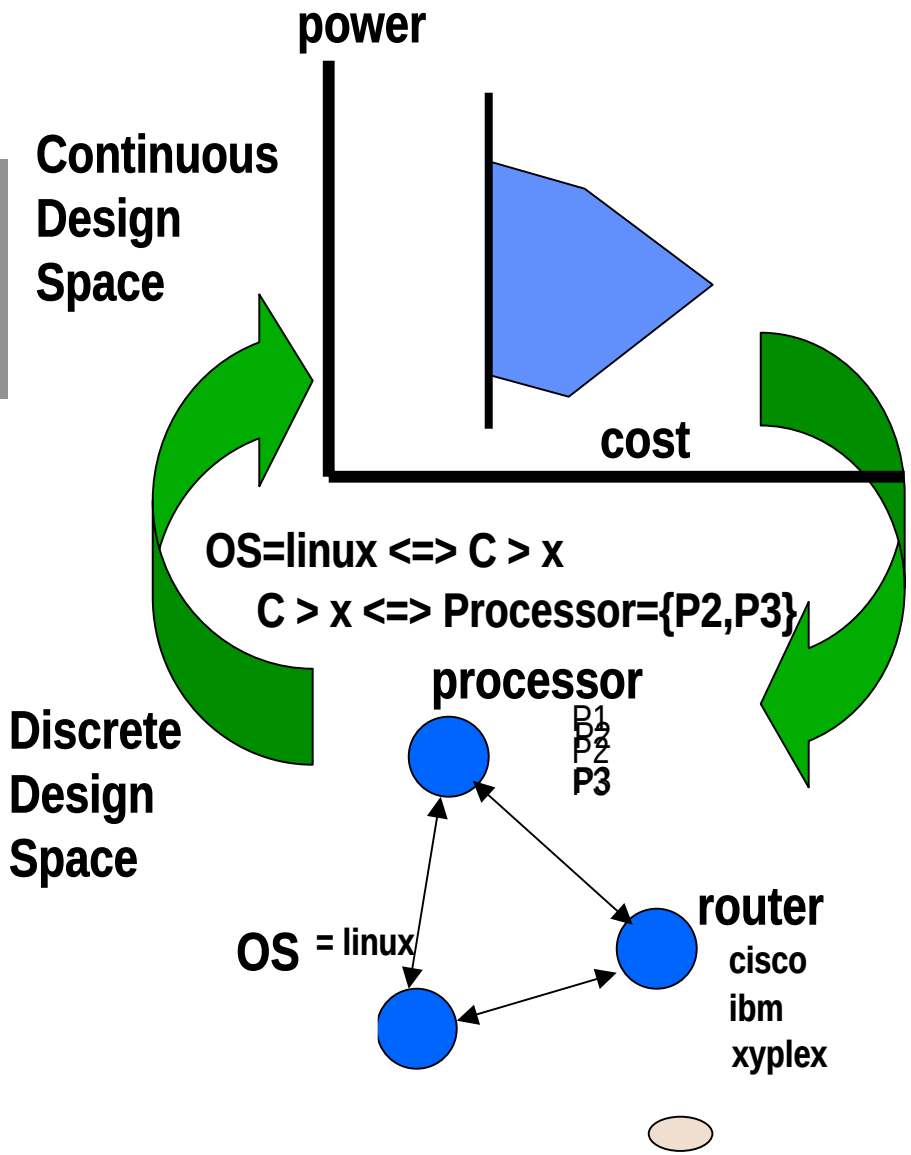
Automatic, dynamic (re)configuration is a key element of the DASADA software vision.

Constraint Gauges

Goals — Verify system architecture

Method — Incremental invocation of constraint based gauges

Technical Basis — Formal definition of system level gauges



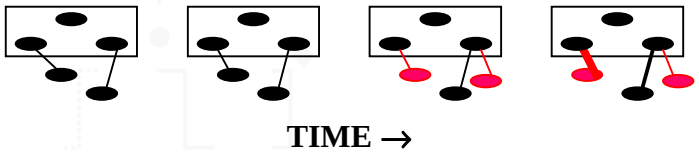
Connectivity Gauges

Goals
- Determine runtime component
- Determine how connections were made

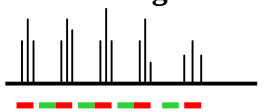
Method
- Insert gauges at a variety of probe points
- Monitor creation and change of component bindings
- Monitor flow of requests

Technical Basis
- Ubiquitously insert runtime gauges into component-based applications

Determining actual runtime configurations & usage.



Monitoring interaction patterns to determine "windows of opportunity" for reconfiguration.



Detecting "dead code" that increases code footprint, may harbor viruses, and complicates evolution & testing. The gauge may provide the following advice:

> **"The MathPak library is bound to your application, but has not been used for 3 months by any user of your application. The library may represent dead code. Library size is 5MB. The library was bound to your application by the gLinker tool on 12/13/99 using the file myMake."**

Gauge Infrastructure: Evolution and Integration Command Center

Goal

- **Enable "go/no-go" decision for re-configuration tasks**
- **Monitor "live" evolution of systems**

Method

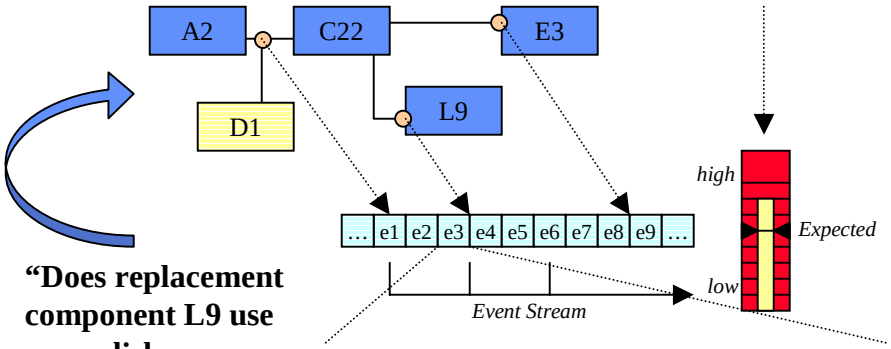
- **Rapid hypothesis testing**
- **FlexML event description**
- **Dynamic probing of components**

Technical Basis

- **Architectural models show how to develop testing regiment**

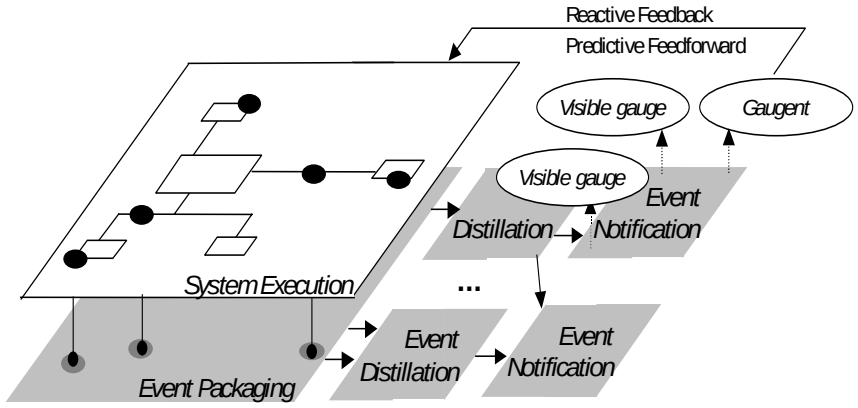
"monitor average server HTTP response time as the new database back-end engine is brought on-line"

Insert probes based on Architectural Model Generate gauges



"Does replacement component L9 use more disk resources than its predecessor?"

FlexML Event Representation

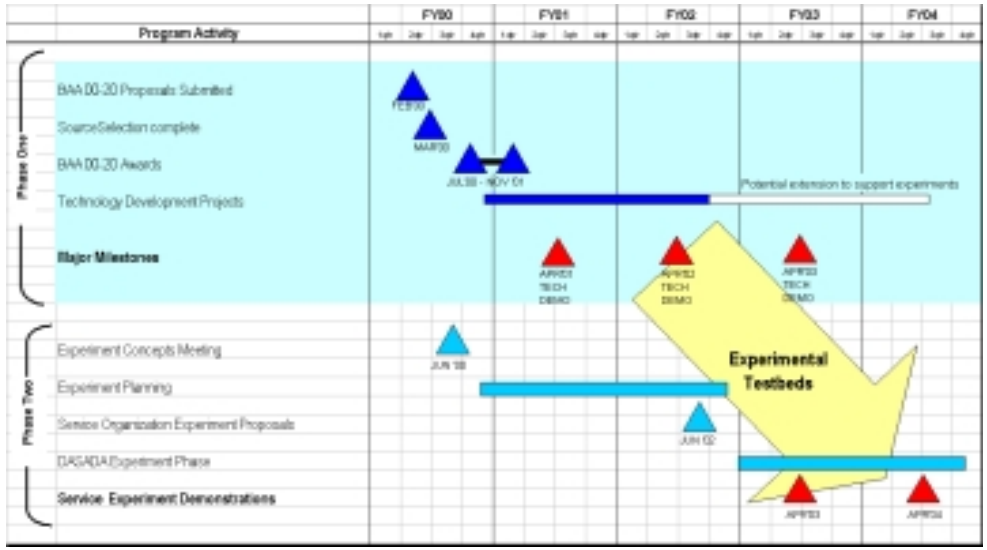


DASADA Phase 2: New Opportunities:Transition Technology to Experiments

GUARANTEE CRITICAL PROPERTIES

REDUCE TIME TO INTEGRATE

INCREASE ASSURANCE EFFICIENCY



Phase 2 Plans

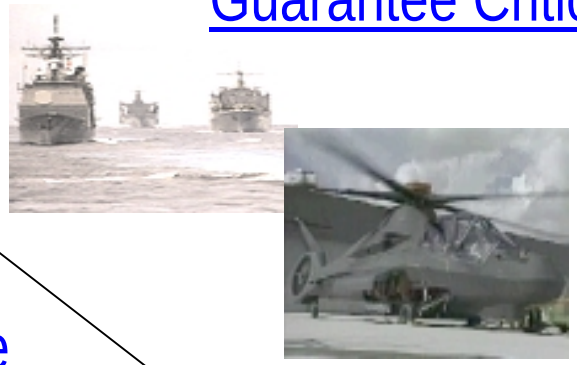
- (Partially) funded planning efforts in FY01-02 (estimate $25K/year)
- Experiments conducted in FY03-04 (estimate 2-3 @ $5,000K each)
- Requires application by DoD organization

Looking for programs with:

- Real problems
- Ability to evaluate the impact of the technology(ies)
- Interest and commitment of the Service organization and contractor(s)

Example Problems for Technology Transition to Experiments

Guarantee Critical Properties



Architectural assessment guarantees critical properties

Reduce Time To Integrate

NEW OPPORTUNITIES

Increase Assurance Efficiency

Models of architecture and behavior reduce integration time/cost.

Update models and axioms based on operational experience

Action Items

- Watch our progress – at ISO WWW site.
- Think about becoming active in planning an experiment – info at ISO WWW site
- Contact us (jsalasin@darpa.mil)