25th anniversary
annual INCOSE
international symposium
Seattle, WA
July 13 - 16, 2015

Architectural Modelling Patterns for Systems of Systems

Claire Ingram, Richard Payne, John Fitzgerald
Newcastle University, UK

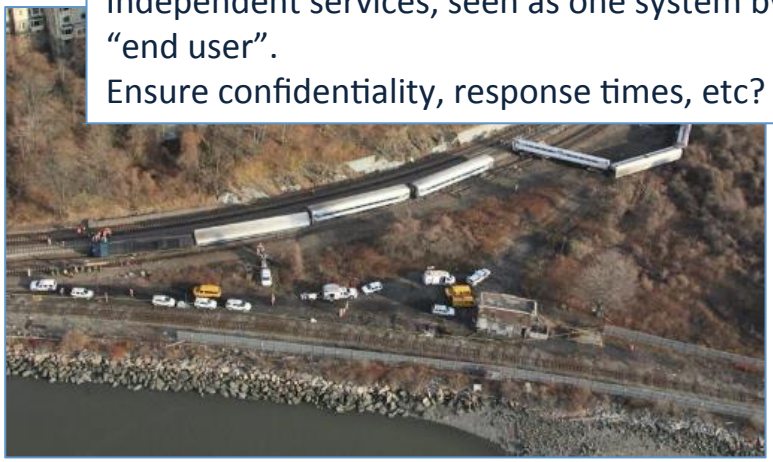
Outline

1. **Context: SOSs and the COMPASS project**
2. Architectural challenges for SoSs
   - What is an architecture?
   - What is a pattern?
3. Modelling patterns for SoSs
   - Architectural patterns
4. Future work

Systems of Systems (SoSs)

Audio/Video (Bang & Olufsen)
Independent networks, devices, content services. Ensure a consistent "SoS experience"

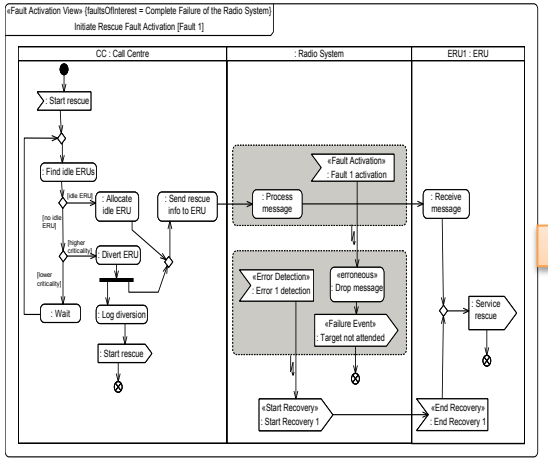
Emergency Response (Insiel)
Independent services, seen as one system by "end user".
Ensure confidentiality, response times, etc?



- SoSs are comprised of elements that are themselves independent systems
- Often exhibit:
  - Operational & managerial independence
  - Distribution
  - Emergence
  - Evolution
- Challenging aspects include:
  - Operational & Managerial Independence of Constituent Systems
  - Complexity of confirming/refuting SoS-level properties
  - Semantic heterogeneity

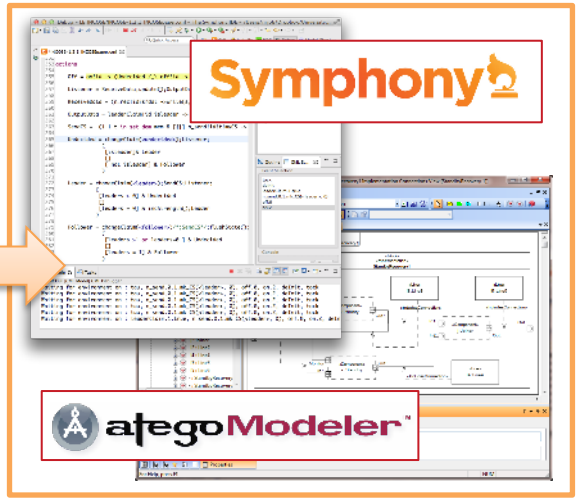
COMPASS

www.thecompassclub.org




```
process CallCentreProc = begin
actions
 MERGE1(r) =
  (dcl e: set of ERUId @ e := findIdleERUs();
   (do e = {} -> DECISION2(r)   |
        e <> {} ->  (dcl e1: ERUId @
                          e1 :=  allocateIdleERU(e, r);
                                 MERGE2(e1, r))
  end)) …

process InitiateRescue =
CallCentreProc [| SEND_CHANNELS |]
RadioSystemProc  [| RCV_CHANNELS |] ERUsProc
```



Architectural Modelling

- SoS Modelling Frameworks
- … instantiated to domains
- **SoS Modelling patterns** & profiles, e.g. Fault-Error-Failure
- Guidelines on negotiation, requirements, integration, test, etc.

Underpinning Formalisms

- Behavioural semantics of SoS
- Tight link to modelling frameworks
- Cope with multiple paradigms.
- Compositional Design
- Dynamic response to adaptation & evolution
- Covering cyber elements, physical, human, economic, social, …

Tool-supported V&V:

- Exploration of Design Space
- Efficient verification by model-checking and proof
- Test generation
- Simulation
- Tools Robustness
- Conformance during evolution, and emergence

25th anniversary annual INCOSE international symposium
Seattle, WA
July 13 - 16, 2015

Outline

1. Context: SOSs and the COMPASS project
2. **Architectural challenges for SoSs**
   - What is an architecture?
   - What is a pattern?
3. Modelling patterns for SoSs
   - Architectural patterns
4. Future work

What is an architecture?

An architectural design may address:

- **System structure:** major components of the system, their organisation and structure.
- **System behaviour:** "dynamic response of the system to events, *providing a basis for reasoning about the system*."
- **System layout:** physical layout & packaging of the system.

Stevens et al. 1998

SoS Architectural Challenges

- Lack of full disclosure between CSs
- Accurately predicting emergent behaviours
- Long lifecycles, legacy or COTS components
- Constituent systems (CSs) evolve with/without the SoS
- Lack of central decision-making authority
- Multi-disciplinary, cross-domain
- High requirement for availability, a volatile operating environment

SoS Architectural Considerations

These prompt questions such as:

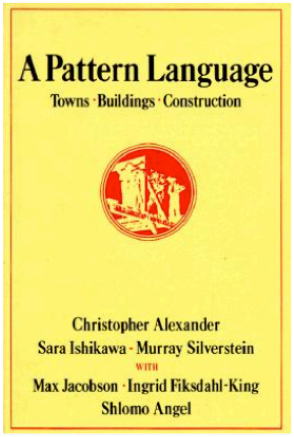
- How far do we need to control propagated changes?
- What is the required level of assurance of emergent behaviour?
- Is there a central decision-making authority?
- To what extent do we want separate concerns?
- How important is resilience or adaptability?
- Do we need a clear, traceable chain of command?

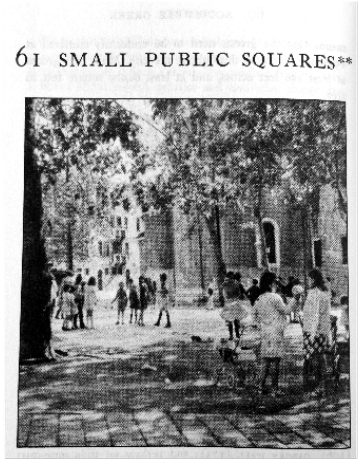
We need:

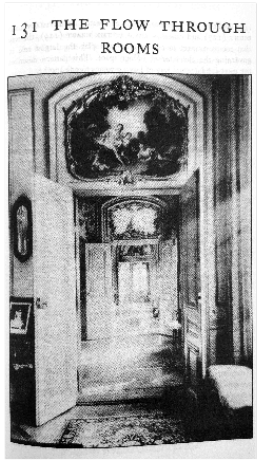
- a basis for comparing alternative SoS architectures
- a means of sharing and passing on experience

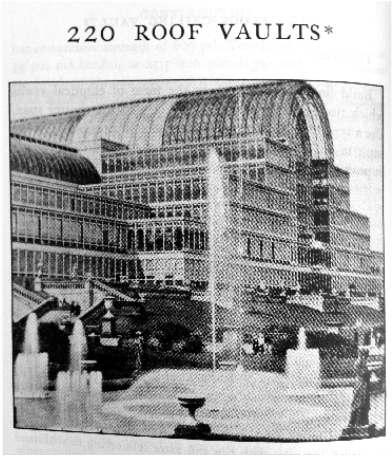
What is a ‘pattern’?

“A pattern *describes a problem which occurs over and over* again in our environment, and then describes the *core of the solution* to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”









Alexander et al., 1977

Outline

1. Context: SOSs and the COMPASS project
2. Architectural challenges for SoSs
   - What is an architecture?
   - What is a pattern?
3. **Modelling patterns for SoSs**
   - Architectural patterns
4. Future work

Patterns for SoS Models

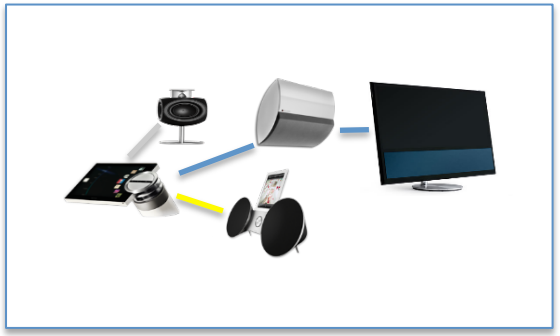
We use *modelling pattern* to mean a pattern that can be applied to modelling aspects of a system, such as architecture or interfaces

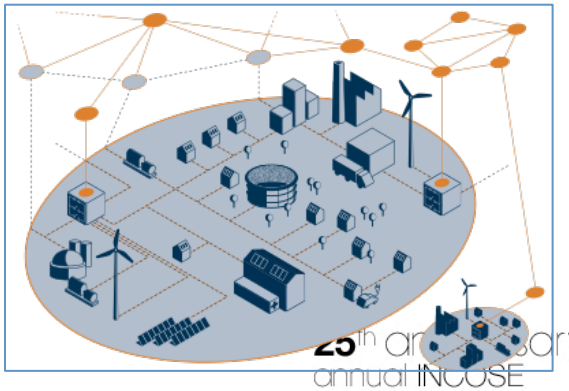
Developing a catalogue of patterns can:

- Facilitate sharing lessons between SoS domains
  - Which SoS challenges does a pattern cope well with or cope badly with?
- Help us learn more about SoS contexts and constraints
  - How and why does a particular pattern arise?
  - How does an architecture or control structure affect SoS performance?

Patterns for SoS Models

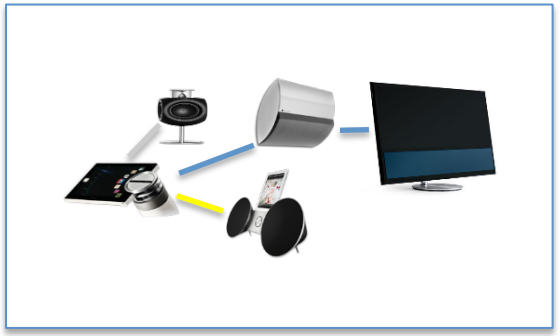
- Patterns observed in or inspired by COMPASS SoSs:
  - Centralised
  - Service-oriented
  - Publish-subscribe
  - Pipe & Filter
  - Supply Chain
  - Reconfigurable Control
  - Infrastructure Grid
  - Blackboard

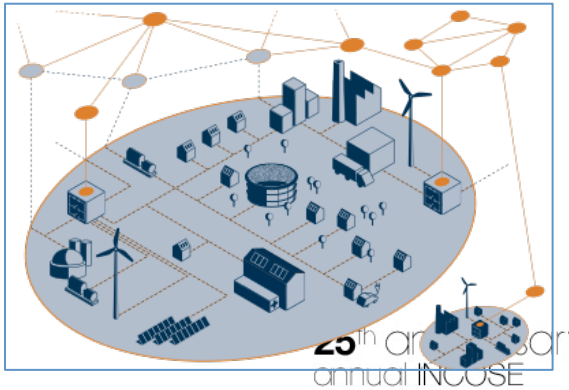




Patterns for SoS Models

- Patterns observed in or inspired by COMPASS SoSs:
  - **Centralised**
  - Service-oriented
  - Publish-subscribe
  - **Pipe & Filter**
  - **Supply Chain**
  - **Reconfigurable Control**
  - **Infrastructure Grid**
  - Blackboard





Centralised

- Central point of control
- “Hub” connected to other CSs, responsible for delivering SoS behaviour
- Hub typically developed specifically for SoS
- Some CSs may be legacy/COTS, or purpose-built
- May or may not force all CSs to communicate through the hub(s)
- Subtypes:
  - Fully centralised
  - Distributed-centralised
  - Hierarchical-centralised

Centralised

SoS considerations

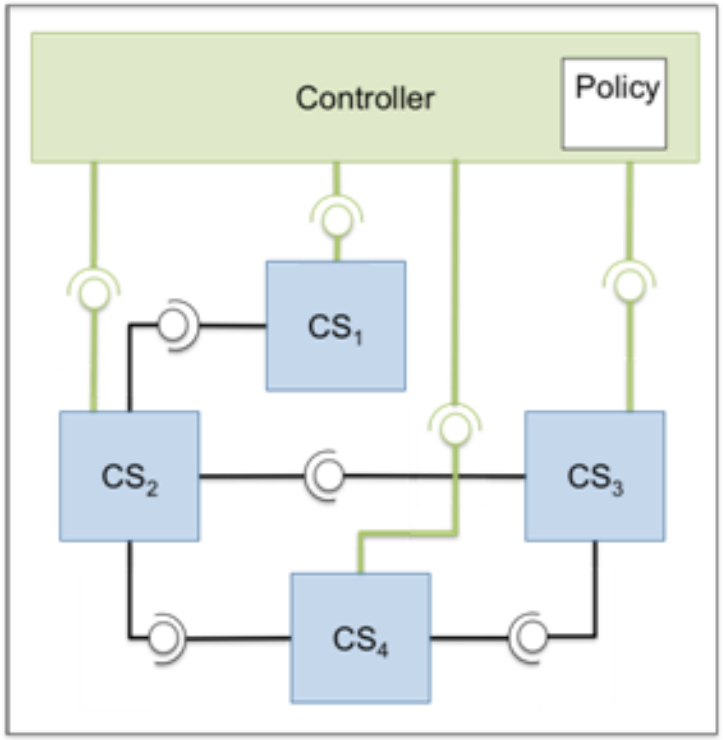
- Centralised control/management
- Can track and/or log where decisions are made
- Re-use existing systems
- If CSs communicate only through the hub, SoS can become loosely coupled
- Permits verification in early design stages

Reconfigurable Control

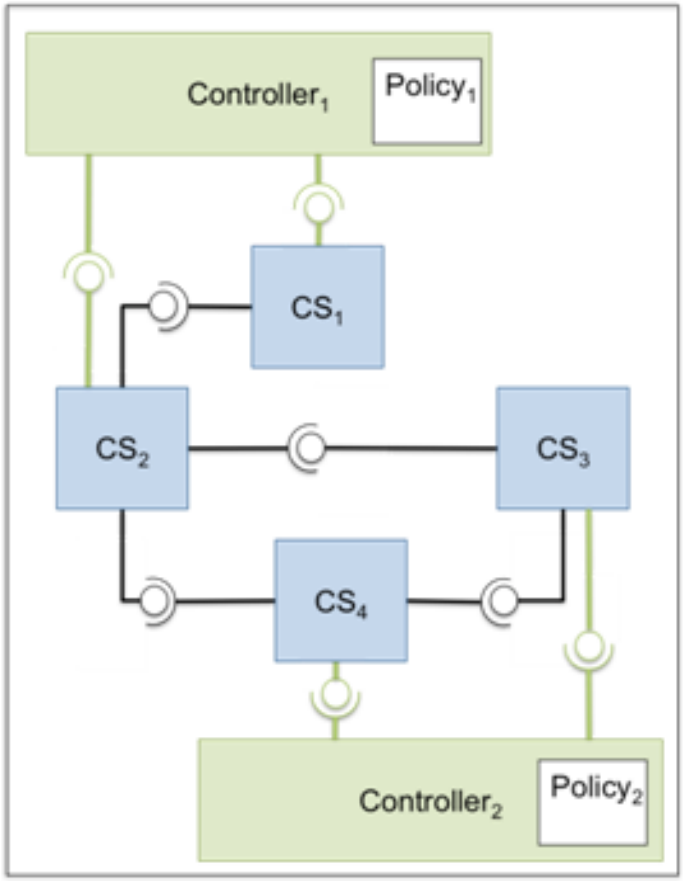
- Dynamic reconfiguration requires some provisions:
  - CS functionality and (optionally) QoS must be specified
  - Alternatives are available for these functions
  - SoS can monitor current performance
- *Metadata* used to describe the functions CS offer
- A *policy* details *when* and *how* to reconfigure SoS
  - Lists necessary functions and minimum performance for each
  - Lists conditions under which action taken
  - Can provide prioritisation
- Explicit *reconfiguration control* CS can monitor CS functionality & performance to decide on actions

Reconfigurable Control

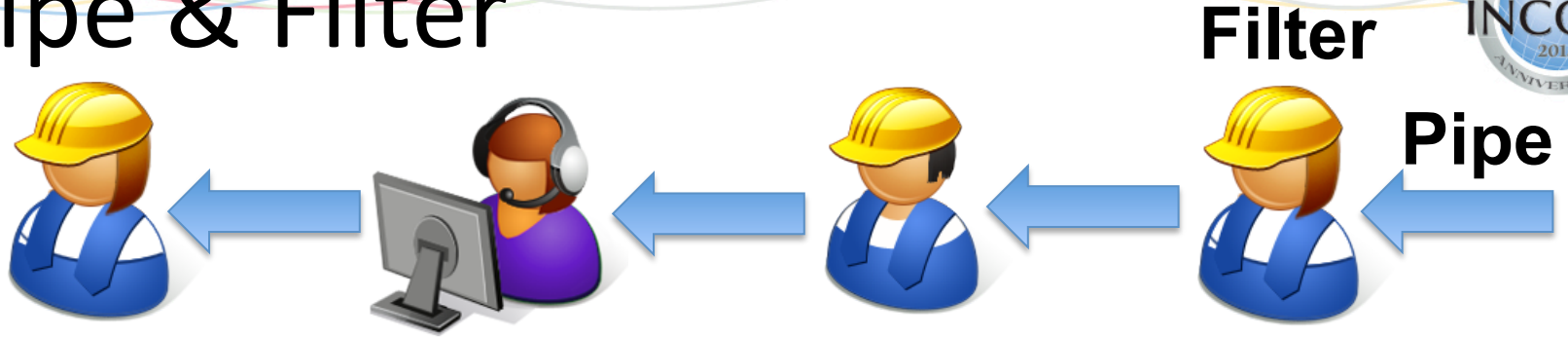
Centralised



Decentralised



Pipe & Filter



- Data or materials processed from input form to output form
- Filters represent the processing steps
- Pipes represent connections between Filters
- Filters are independent, do not share state or know each other's identities

Garlan & Shaw 1996, Buschmann et al. 1996

SoS considerations

- Unsynchronised evolution is possible
- Dynamic reconfiguration is possible
- May or may not have central control

Supply Chain

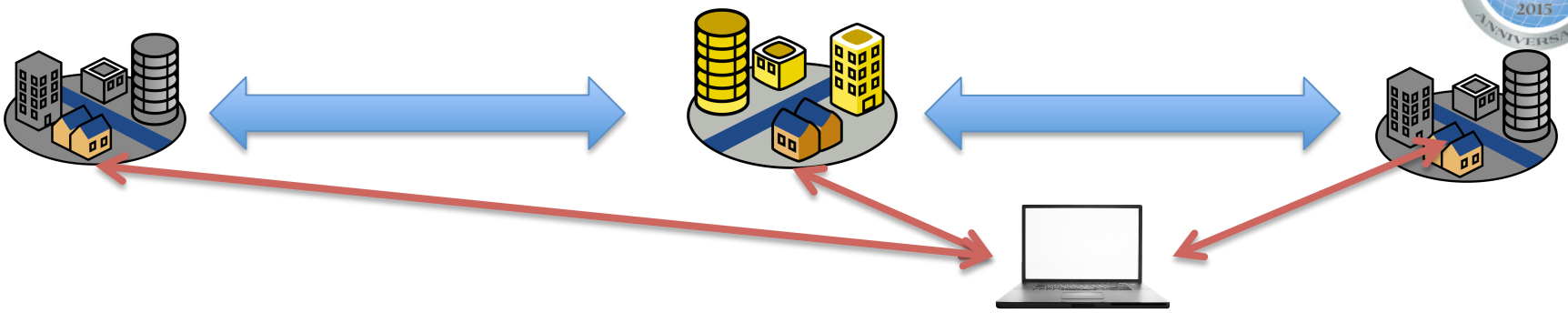
A specialised pipe-and-filter

- Suppliers/integrators are the “filters”
- Logistics acts as a “pipe”

Differences with pipe-and-filter:

- Logistics shares internal state and participate actively
- CSs may be aware of the final goal
- CSs may be aware of internal status of their peers
- CSs are also capable of generating input to be returned upstream

Infrastructure Grid



- Delivers critical civil infrastructure, e.g., power, water, roads, communications, etc.
- Divided into fixed geographical regions, each operated by an autonomous controller
- CSs exchange flows with direct neighbours, and data with any other CS
- Optional central authority; regulations impose standardisation
- May optionally be a hub for communications

Differences from pipe-and-filter:

- CSs know identity of neighbours – tightly coupled
- The flow may be bi-directional
- CSs may share details of internal state

Infrastructure Grid

Differences from pipe-and-filter:

- CSs know identity of neighbours
- The flow may be bi-directional
- CSs may share details of internal state

Subtypes:

- *Fully decentralised*: no organisation with overall control
- *Partially decentralised*: one organisation controls an important proportion of infrastructure
- *Data-centralised*: no overall authority, but there is a central hub for data sharing

Reconfigurable Control

SoS considerations:

- Dynamic reconfiguration helps to provide resilience
- Performance optimisation facilitated
- Allows for central authority
- Should be partnered with a loosely-coupled architecture

Outline

1. Context: SOSs and the COMPASS project
2. Architectural challenges for SoSs
   - What is an architecture?
   - What is a pattern?
3. Modelling patterns for SoSs
   - Architectural patterns
4. **Future work**

Future work

SoS Architectural Considerations:

- How far do we need to control propagated changes?
- What is the required level of assurance of emergent behaviour?
- Is there a central decision-making authority?
- How important is resilience or adaptability?
- Do we need a clear, traceable chain of command?

We need:

- a basis for comparing alternative SoS architectures
- a means of sharing and passing on experience

Future Work

- More patterns – develop a catalogue
- SoS problems and means for assessing different SoS patterns against them
- Standardised approach for identifying, collecting and documenting patterns
- Better understanding of how and why SoS patterns arise/are applied
- Better understanding of weaknesses/risks of each pattern

Newcastle University

INCOSE 2015 TWENTY-FIFTH ANNIVERSARY

This work is part of the COMPASS project: research into model-based techniques for developing, maintaining and analysing SoSs

Claire.Ingram@ncl.ac.uk @_Claire_Ingram

John.Fitzgerald@ncl.ac.uk @NclFitz

Richard.Payne@ncl.ac.uk @riffio

COMPASS

thecompassclub.org

25th anniversary annual INCOSE international symposium
Seattle, WA
July 13 - 16, 2015

Defining systems of systems

A system composed of other constituents, each of which is an independent system in its own right

- Operationally & managerially independent constituents
- Geographically distributed
- Continuously evolving
- Exhibiting emergent behaviour

"emergent": global behaviour produced by the whole SoS, can't be produced by a single constituent alone

Three major technical challenges:

- **Independence and autonomy of constituent systems**
  - Constituent systems evolve at the behest of their owners
  - *Response: Collaborative SoS modelling by contractual (rely, guarantee) interface specification*
- **Complexity of confirming/refuting SoS-level properties**
  - Verification of emergence
  - *Response: verified refinement for engineering of emergent properties; simulation tools allow exploration for unanticipated behaviours*
- **Semantic heterogeneity (integrating models)**
  - Wide range of interacting features in models (e.g. location, time, concurrency, data, communication)
  - *Response: extensible semantic basis*

Service Oriented

- Applications composed using third-party services, offered by providers
- Services produce a contract (a standardised service description and service-level agreement)
- Services do not share internal state, making them stateless to the SoS
- Each service possibly more than one provider

Service Oriented

SoS considerations

- Analysis of SoS emergent behaviour
- Unsynchronised evolution is possible
- Allows a central SoS authority
- Cross-domain development
- Separation of concerns
- Support for redundancy

Suitable SoS types

- Directed
- Acknowledged

Publish-Subscribe

- Two types:
  - *Content-Based Publish-Subscribe* (EBPS): subscribers describe type of content they wish to receive
  - *Data-Centric Publish-Subscribe* (DCPS): messages are categorised using topics provided by publishers
- We focus on DCPS here

Publish-Subscribe

- No central hub
- Concepts:
  - Topic – a data-object in a given domain
  - Publisher – responsible for data distribution, uses a Data Writer to publish data on a Topic
  - Subscriber – receives data on Topics, using a DataReader
  - Publishers & Subscribers have defined interfaces for interacting, typed for a given Topic
  - Publisher, Subscriber, Topic, interfaces each have QoSPolicy
- Any CS can be Publisher, Subscriber or both
- CSs register/deregister on a Topic, to leave or join the SoS

Publish-Subscribe

SoS considerations

- Loose coupling between publisher & subscriber
- Subscribers don't need to understand publisher's domain (and vice versa)
- Redundant designs possible
- No central manager
- Monitoring performance may be difficult

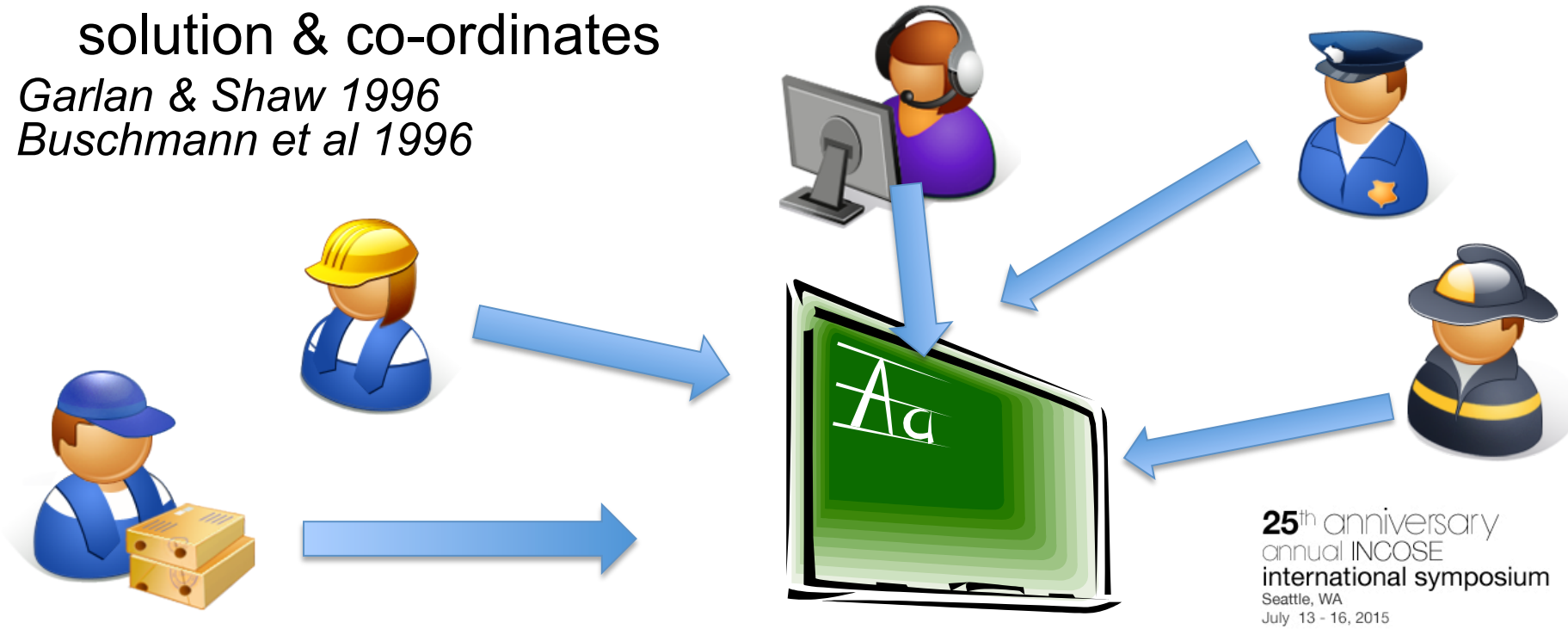
Suitable SoS types

- Collaborative

Blackboard

- Blackboard CS provides interface for reading/writing data
- Knowledge Source CSs write to/remove from the Blackboard
- Knowledge Sources work independently & in parallel
- Control CS evaluates solution & co-ordinates

Garlan & Shaw 1996
Buschmann et al 1996



Blackboard

SoS considerations

- Development of expert or knowledge based systems
- Separation of concerns
- Efficient problem-solving
- Possible to support some degree of central authority
- Loose coupling
- Redundancy is possible

Suitable SoS types

- Directed
- Acknowledged
- Collaborative