

The BROADEN Distributed Tool, Service and Data Architecture

Martyn Fletcher, Tom Jackson, Mark Jessop, Stefan Klinger,
Bojian Liang, Jim Austin.

University of York.

The Domain

- Aero- engine diagnostic and maintenance domain – and others!
- Data collected on aircraft arrival.
- Analysis done:
 - Automatically in flight - onboard.
 - Automatically on ground - more extensive.
 - By Remote Domain Experts for new / emerging conditions.

The Architecture

- Geographically distributed e.g. at airports:
 - Data – potentially in vast amounts.
 - Services used by tools or other services – services taken to data.
- Geographically distributed users and tools e.g. at data processing centres, engine manufacturers, etc.
 - User analysis, visualisation - Domain Experts
 - Tools – legacy and new

Tools

- Tools designed to use remote services.
- Tools designed to interact with other tools – need standard for interactions.
- Legacy tools designed to operate standalone and not interoperate.

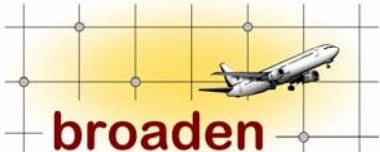
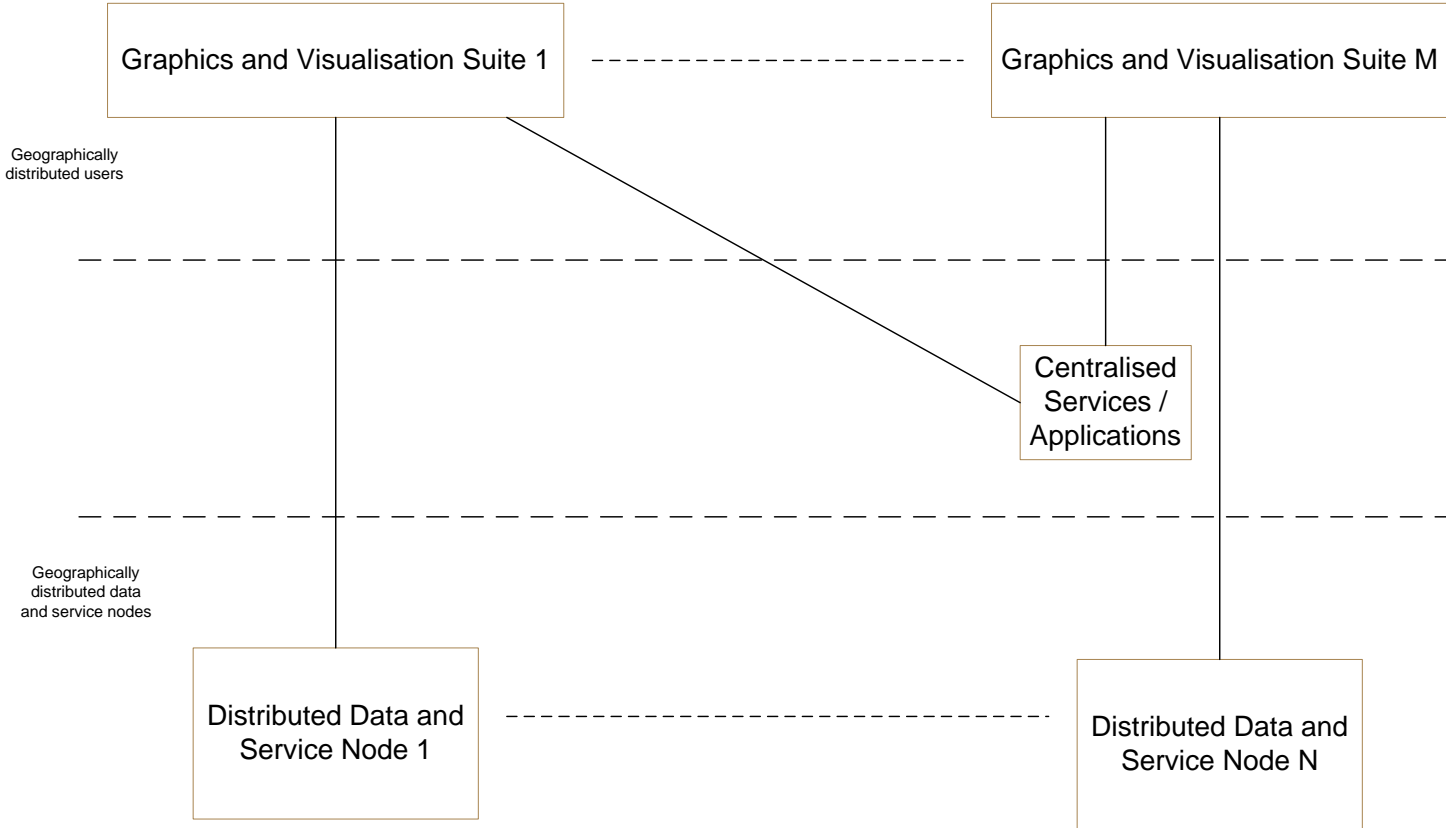


Services

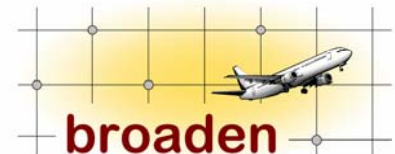
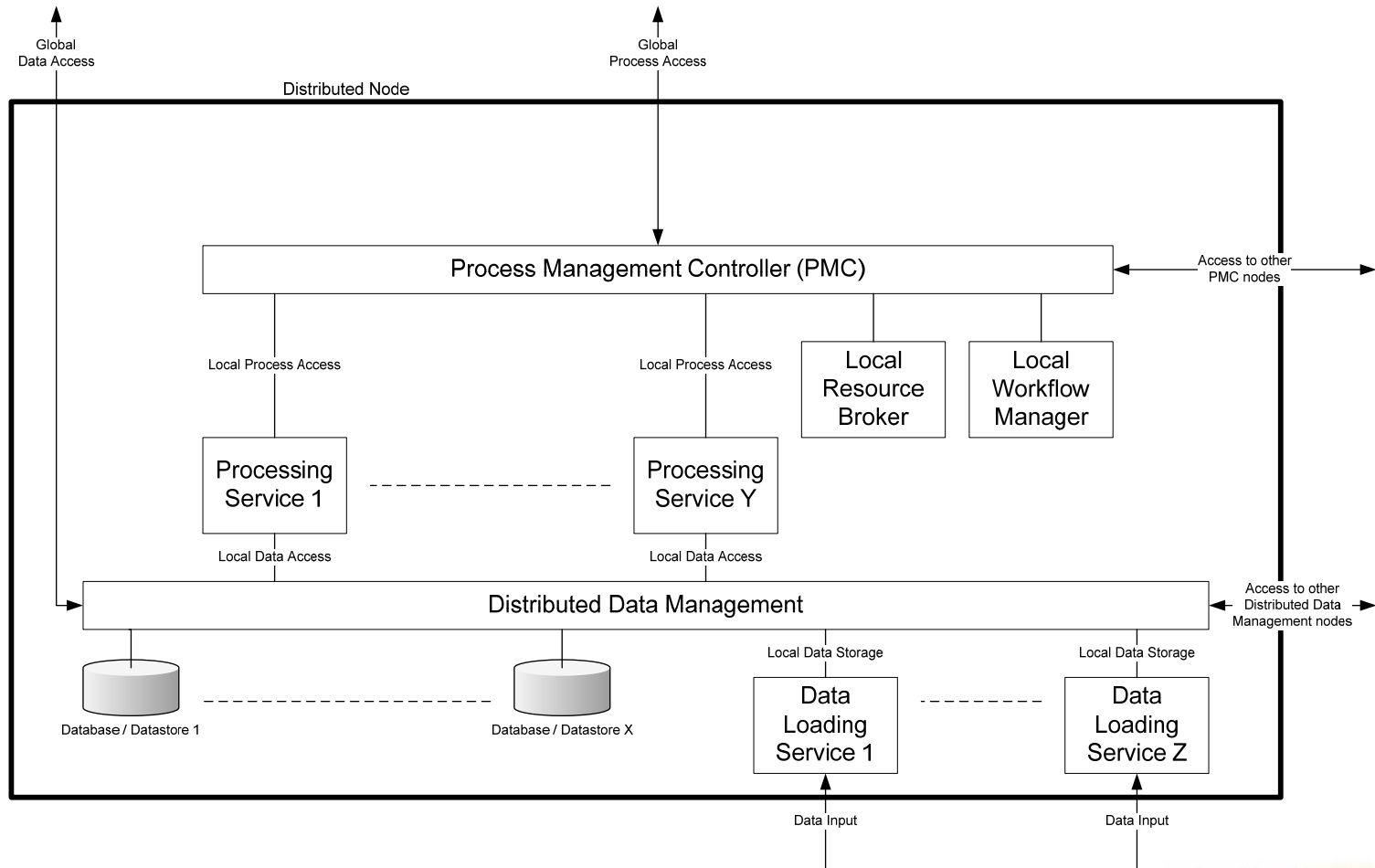
- Services may be distributed – particularly located near the data sources.
- Services may be autonomous e.g. data input services triggered by data arrival.
- Services may interact with other services.
- Some centralised services.



Existing Architecture



Overview of Existing Distributed Node



Existing Architecture

- Caters for tool – service interactions.
- Does not cater for tool to tool interactions other than direct connection.
- PMC currently implements the following design patterns:
 - “Recipient list” – sends out request to a list of recipients.
 - “Aggregator” – aggregates results.



Require

- Tool – tool interactions.
- Loose coupling between tools.
- Allow additional tools and services to be added into the framework:
 - Launch, data exchange etc.
 - Without modification to existing tools or services.

Patterns required

- Some design patterns required for middleware:
 - “Wire tap” – to log data and tool launches.
 - “Application Launcher” – launch tools.
 - “Publish – subscribe” – data available to many tools.
 - “Message translator” – allow systems with different data format to communicate.
 - Etc.



Approach

- Introduce middleware.
- All tools / centralised services register automatically or via manual administration interface with middleware.
- All tools get registry information from middleware and populate menus accordingly to services interacting tools available.
- Middleware captures data for logging in data store(s).
- Middleware monitor services and tools to ensure correct operation.
- Middleware exchange information between “tools & tools”, “tools & services” and “services & services”.
- Middleware allows tools to be launched from workbench or from other tools – in each case this is logged.



Required Interactions

- Tool to Tool:
 - Tools will need to launch other tools with data supplied
 - Data exchange:
 - XML.
 - Likely to need non XML – for legacy tools or if XML is not efficient.
- Tools to Service data exchange:
 - XML e.g. SOAP.
 - May need non XML.
- Services – service:
 - XML e.g. SOAP.
 - May need non XML.

Options

- Middleware options:
 - Enhanced PMC.
 - Enterprise Service Bus (ESB) – also could use Message Framework for non XML data?

PMC as middleware - Advantages / Disadvantages

- Advantages:
 - Can modify / customise as we want.
 - High Performance.
- Disadvantages:
 - Significant modifications needed.

ESB / Messaging Framework - Advantages / Disadvantages

- Various design patterns can be implemented.
- Advantages:
 - Versatile, Expandable, Scalable.
 - Workflow manager may be added.
 - Standardised for future expansion – if use standardised version.
 - Incrementally integrate ESB.
- Disadvantages:
 - Wide ranging technology - learning curve.
 - Potential low performance.
 - Difficulty with non XML messages – could use message broker.
 - More facilities than needed.
 - Technology still developing

ESBs available

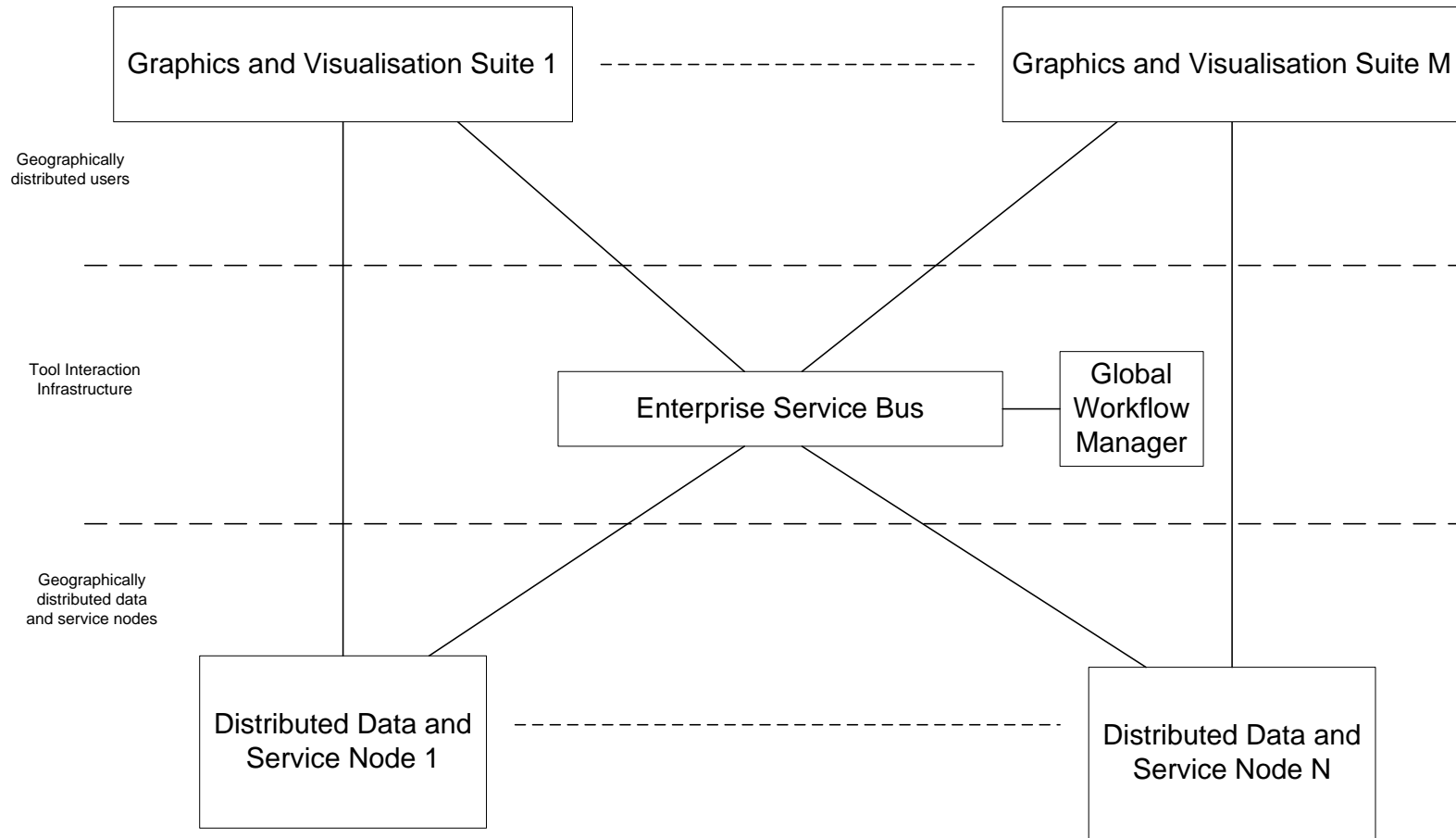
- Proprietary ESBs:
 - Websphere ESB (IBM) - not JBI compliant.
 - TIBCO BusinessWorks.
 - BEA AquaLogic Service Bus - not BPEL / JBI compliant.
 - Cape Clear 6.1
 - Cordys 4.2
 - FioranoESB Suite 3.7
 - FusionWare Integration Server 3.0
 - Iona Artix 3.0 Advanced
 - PolarLake Integration Suite 4.0
 - Sonic SOA Suite 6.1.
 - Vitria BusinessWare.
 - Microsoft BizTalk together with the .NET framework.
 - Kenamea
 - KnowNow
 - SeeBeyond
 - Snapbridge software
 - SpiritSoft
 - Wakesoft
 - webMethods
- Open source / community ESBs:
 - Open-ESB from Sun.
 - ServiceMix from Apache.
 - Petals from ObjectWeb.



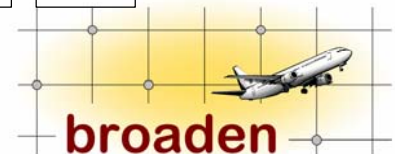
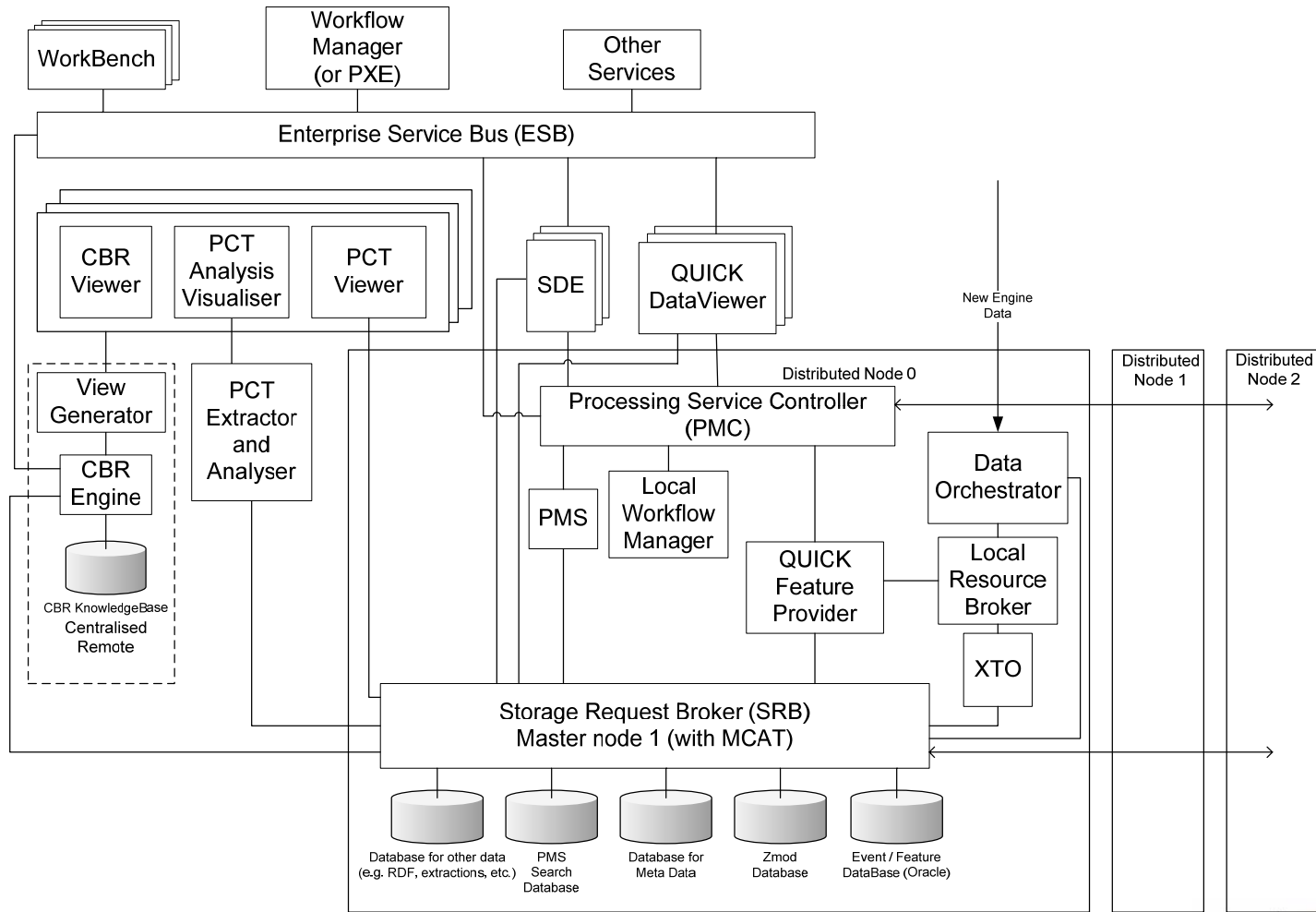
Current Investigations

- Use of ESB.
- Use of a messaging framework for applications that do not rely on XML data exchange.
- Currently investigating the use of:
 - ServiceMix ESB
 - Mule messaging framework
- Some enterprise integrations require both an ESB and a Message Framework.

Overview of Architecture



The Real Architecture



Concerns / Issues

- Efficiency of ESB – using XML.
- Feasibility of using ServiceMix / Mule in the domain.
- Are ESB and PMC needed as separate elements?
- Should we modify PMC to encompass whole middleware task?
- Should we integrate PMC with ESB, etc. as a high performance extension?

Future Work

- Assess merger or otherwise of PMC and ESB.
- The testing and demonstration of the generic architecture in the BROADEN application domain.
- The testing and demonstration of the generic architecture in other application domains.
- The introduction of fault tolerance as appropriate (inter and intra node).
- The exploration of fault tolerance techniques within the ESB.
- Performance testing would be helpful to discover issues early in the process.

Acknowledgements

- The BROADEN project is part-funded via the UK Department of Trade and Industry under its Technology Innovation Programme.
- The work described was undertaken by teams at the Universities of York, Leeds and Sheffield with industrial partners Rolls-Royce, EDS, Oxford BioSignals and Cybula Ltd.

