Assurance of Distributed Systems

More questions than answers

Robert O’Dowd

With thanks to: Mark Davies, Derek Dominish, Kate Foster, Thong Nguyen, Jeff Tweedale

Backdrop: Increasing accountability in Defence

- Publicity if projects exceed cost or schedule
- Increasing regulation and policy emphasis on safety
  - Risk management
  - Personal responsibility
- WHS Legislation
- Defence Aviation Safety Plan (DIG OPS 02-2)
  - Wide ranging definition of aircraft
  - Covers
    - All risks to people or property that are a consequence of the flight
    - Aeronautical products with intended end-use in aircraft – including software
  - Obligations and liability on ALL Defence personnel
    - Military, civilian, contracting authorities
Increasing complexity of systems being acquired

- Difficulty locking down needs and requirements
  - And mapping requirements to a design
- Reliance on communications, system integration, networking
  - Interconnections a driver of complexity
- Software a key enabler of integration
  - Inherently abstract
  - Limited ability to design before implementing
    - A point where implementation is the design
  - Easy to change, hard to get right
- Vendors sell features, not minimalism
- More critical applications
- Rapid turnover and obsolescence after delivery
  - Design and implement for change

Aerospace Technical Regulation

- Regulators (FAA, CASA, DGTA)
  - Define rules, regulations
  - Provide guidance
  - Assess competence of organisations
  - Delegate authority to designated representatives
- Competent organisations (Projects, SPOs, DARs, OEMs)
  - Interpret rules, regulations, and guidance
  - Gather evidence of compliance
  - Seek sign-off by authorised delegates
- Liability traditionally rests with organisation, not people
- Contrast with WHS
  - Can override any other legislation, including ....
  - Criminal investigations of mishaps
  - Individual liability
System Assurance

- Provision of evidence that a system is dependable
- Dependability is ability to justifiably trust a product or service
  - Availability (readiness for correct service)
  - Reliability (continuity of correct service)
  - Safety (absence of catastrophic consequences on user(s) or environment that are to be protected)
  - Integrity (absence of improper system alteration)
  - Maintainability (ability to undergo maintenance or repair)
  - Confidentiality (absence of unauthorised disclosure of information)
- Relevant system properties
  - Usability
  - Functionality
  - Performance
  - Manageability
  - Cost

High criticality systems vs trends in distributed systems

<table>
<thead>
<tr>
<th>Glass cockpit avionics</th>
<th>Middleware approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Fly by wire” systems</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>Minimise complexity</td>
<td>Enable complex systems</td>
</tr>
<tr>
<td>Predictable system behaviour</td>
<td>Unpredictable emergent properties</td>
</tr>
<tr>
<td>Reliability by design</td>
<td>Reliability a deployment property</td>
</tr>
<tr>
<td>Hard timings</td>
<td>Timing unpredictability (statistical)</td>
</tr>
<tr>
<td>Traceability</td>
<td>Governance</td>
</tr>
<tr>
<td>Tight configuration control</td>
<td>Minimise configuration constraints, adaptability, implementation diversity</td>
</tr>
<tr>
<td>Reviewer independence from designer and developer</td>
<td>Architects, designers, and developers have implied responsibility for assurance</td>
</tr>
</tbody>
</table>
Different approaches to assurance

- Physical devices like aircraft
  - Mathematical modelling of design, environment, requirements
  - Validate modelling through test
  - Product-based certification based on properties of (mathematical models of) the product
- Discrete (digital) systems: both software and hardware
  - Control, monitor, document the development process (e.g. DO178B/C, DO254, MIL-STD-498)
    - “Cannot show how well we built it so we’ll show how hard we tried”
  - Infeasible to do complete testing
    - 114000 years to justify claim of $10^{-9}$ probability
    - Cannot extrapolate from incomplete testing
  - Formal methods seek move to product-based assurance
    - Not yet scalable to large software-based systems
- Perspective of distributed systems architects
  - Emphasis is QoS assurance
  - Non-functional requirements
    - Response time, throughput, availability, security, ...

One approach to assurance: the Airborne Mission Entertainment System

- System developer not required to provide assurance evidence
  - OEM responsible for achieving certification of base aircraft
  - Relies on separation of mission system from flight critical systems
  - Lack of evidence from test contractually deemed evidence of absence of faults
- Liability for impacts of system use rests with acquirer and operator
- Will this remain possible for future military acquisitions?
  - Increasing integration and automation
    - Including directing or controlling flight
  - C2 systems support decisions with lethal effect
  - Mission systems that control weapons
    - Weapons dangerous by definition
  - Developers lobbying for less demanding acceptance process
  - Trends toward evolvable architectures and designs
    - Implementation size and complexity
    - Facilitates building more complex systems
Architecture frameworks, middleware, programming frameworks, ...

- Primary goal is developer productivity
  - “Abstract away” implementation details
  - Support development of more complex systems
- Specific trade-offs are made ... sometimes covertly
  - Qualities of service (QoS)
  - Assurance often treated as a deployment activity
- The shift is in who is exposed to complexity
  - Application developer
  - Middleware developer
  - System architect
  - Repository manager
  - Acquisition organisation
  - Support
  - End user
- Who has the end-to-end view?

Distributed system folks recognise QoS assurance as an end to end concern ...but ...

- To date, most advances in context of individual architectural components
  - Distributed system platforms, Operating systems, Transport systems, Networking, ...
- Mostly addressed from infrastructure or provider viewpoint
  - Application or user concerns secondary
  - Focus on prioritising network traffic
- Incomplete support in architectural frameworks
  - Subset of facilities needed to control, monitor, maintain QoS
  - Often specific to a system level or network architecture
- Limited QoS enforcement beyond security and bandwidth shaping
  - Connections based on matching application need & service offer
    - Client or service responsible for subsequent monitoring
    - No predetermined guarantee
  - Policies where framework support is optional
  - “Hint” policies: neither monitoring nor enforcement required
    - Realtime policies over IP infrastructure
Link-16 and SOA – very different things

Link-16
- Military Tactical Data Exchange network
- Exchange tactical picture in near realtime
  - Also text, imagery, voice
- Standards (US and NATO specifications)
  - Terminal interface (hardware/software)
  - Procedural interface (message formats and protocols)
  - SOPs

SOA
- Software architecture design pattern
- Compose software systems from discrete pieces (services)
  - Facilitate cooperation of computer systems over a network
- Standards-based
  - Protocols that describe how services pass and parse messages
  - Metadata specifications to characterise services and the data that drives them

Aspects of assurance: planning and design philosophy

Link-16 network
Maximise central planning & design
- Standardised approach
- In advance of use
- Optimise use of fixed resource

Minimise online changes
- Network manager responds to change of operational situation

A SOA based system
Business agility
- Architectural design pattern
- Assemble system from services
- Share/reuse across network
- Self-describing interfaces
- Adapt to changing resource

Maximise online flexibility
- Extensibility
- Dynamic discovery
- Incremental implementation
Aspects of assurance: interoperability

Two families of specifications
- US and NATO
- Each an interlocking set of standards

Explicitly managed interoperability
- US DoD Joint Interoperability Test Command
- ADF-TA has role to ensure correct TDL functionality at platform level
  - Assist TIED policy development
  - Development advice
  - Test and compliance services
  - Operational services (Network design, support of JICO)
  - Joint JDL training

Multiple international standards & specifications addressing different needs
- OMG, Open Group, IEEE, ....
- Some element of mix and match

System developers rely on implicit interoperability
- “Abstracted away”

Aspects of assurance: Characteristic elements

Planning

Design

Initialisation

Operation

Recording/analysis

Standards to
- Describe application structure, behaviour, architecture (UML, MDA)
- Publish component interfaces

Architectural composability

Visual design, execution, and maintenance of software and other processes
Aspects of assurance: End usage of deployed system

<table>
<thead>
<tr>
<th>Process centric</th>
<th>Workflow centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mostly) pre-planned context</td>
<td>Context of use often unknown at design time</td>
</tr>
<tr>
<td>Assurance (or achieving sign-off) a key consideration in preparatory work before deployment</td>
<td>Assurance often viewed as deployment activity, so not explicitly considered earlier</td>
</tr>
</tbody>
</table>

Discussion

- Changing legal, regulatory, policy space
  - State of practice catching up
- Commercial trends from tightly managed to more loosely coupled distributed systems
  - Benefits for developers
  - More rapid delivery of new functionality
  - Different challenges and expectations related to assurance
    - And even interpretation of what assurance is about
- Focus here is assurance of a complete system
  - QoS assurance is only part of the problem
The basic concern is handling of complexity

General means of easing assurance

Minimise complexity

Enable end-to-end analysis

Ensure stability
  – Design and implementation
  – System behaviour over time

Ensure consistency
  – Data and information integrity
  – Authentication, authorisation, auditing

Commercial trends in distributed system development

Ease jobs of building and managing more complex systems

Encourage architectural approach and governance

Ease changes of design and implementation
  – Dynamic behaviours
  – Management through monitoring

Evolving elements of best practice
  – Achieving scalability
  – Governance
  – Maturing standards for authentication and authorisation
  – Some progress to standards for auditing

And boundaries

Acceptance of SOA-based aircraft mission systems
  – Locked down and defined configuration
    ▪ Start up and fault tolerance schemes
    ▪ Constrain options for dynamic behaviour
  – Limit scope to on-board platform
  – Limit (or prevent) interaction with flight systems

System architects have much greater potential in mind
  – Dynamic updates, load balancing, fault tolerance, self-healing, etc
  – Sharing off-board system resources
  – Exploiting enterprise approaches in tactical setting
Questions

Who is responsible if things go pearshaped?
- Military operator?
- Project?
- Regulator?
- System Developer?
- Government?
- All of the above and more?

Where do challenges sit?
- Policy?
- Technical approach?
  - Locking down system configuration versus exploiting dynamic emergent behaviours
  - Focus on minimising time to deliver features?
- Supporting management decisions?
  - Roll-up of details in management brief can conceal complexity
  - Complexity makes it hard to identify impacts
- Paradigm shifts related to the above?
  - Accepted approaches may no longer apply