Software Interoperability.... Same game, same rules?

Dave Penkler
CTO OpenCall, HP
Presentation Outline

• Introduction
• Interoperability Challenges
• The software bulge
• Software Interoperability
• Conclusion
Introduction

Interoperability, a working definition:
• The ability of two or more users, devices, networks, information systems, components and applications to communicate, exchange information and use it.

Interoperability in a converged ICT environment:
• User Experience of Content, Information and Communication services:
  − Satisfactory Quality, Performance and Cost
  − Safe, Secure & Dependable
  − Improving over time (Features, Cost, QoS etc)
• While allowing technology innovation, vendor choice & competition:
  − Media formats: acquisition, protection, processing and rendering
  − Networking: Access, Transmission, Switching & Control
  − Information Services, Processing & Storage
  − Sensors & Transducers
• And Maintaining Investment Protection in
  − Equipment
  − Software
  − Support functions
Interoperability Challenges

• Explosion of Standards and Standard Development Organizations
  – 170+ consortia listed at http://www.consortiuminfo.org/links/interoperability/

• High cost of specifying, testing and maintaining multilateral and multi-standard interoperability required for end-to-end interoperability
  – no clear owners

• Consortia and their ecosystems at best address interoperability for specific technology / value chain related vertical or horizontal slices
  – the days of monolithic interoperable out of the box end-to-end standards are gone

• Industry R&D investment in technology innovation:
  – Generating intellectual property
  – Looking for disruptive technology
  – Creating ecosystems / value chains around assets

• Portability of services, content and user identity/addresses across
  – Multiple devices
  – Different Networks
  – Service Providers
The Software Bulge

• The bulk of ICT infrastructure development activities are now software related.
  – System on a Chip
  – Software defined radio
  – Digital Signal Processing
  – Middleware
  – Applications

• New technology / functionality introduced through software upgrades to existing systems
  – Too expensive to replace whole system or build from scratch

• Software component interoperability dependencies:
  – Vertical dependency on their deployment platform
  – Horizontal dependencies on multiple client/server/peer systems and services in the infrastructure

• The well established standardization techniques must be extended address software interoperability
Software Interoperability Environment

- **Software Interoperability**
  - Portability of software components -> HW/OS Technology Choice/Innovation
  - Substitutability of components and containers: -> SW Choice/Innovation
  - Open standards with detailed conformance test specifications enable Portability, Substitutability and investment protection

- Implemented as **Components** hosted on specific containers.
  - Interoperability required with containers and application service components

- Implemented as **Components** hosted on specific containers.
  - Abstraction layer for remote applications: e.g.:
    - Authentication, file transfer, application protocols

- Implemented “containers” that host and provide platform services to software Components
  - E.g: CORBA, J2EE, .NET

Lacking open Conformance Test specifications

Existing Standards:
- POSIX 1003
- ISO DIS 23360
The Software Component model

- Open or proprietary component implementation
- A Vehicle for introducing IPR and differentiation

Component

Service Provider Interface
Standardised Service interface

Service User Interfaces
Standardised Interfaces

Communications Standardisation Focus

Vertical Interoperability
API

Portability

Substitutability

Portability

Horizontal Interoperability

Standardised Protocol interface

Communications

Software Standardisation Focus

Vertical Interoperability
API
Software Interoperability Issues

• Fragmentation due to multiple container technologies and incompatible implementations

• Proprietary control of important container technology

• Software test specification is still very labour intensive
  − Often neglected or deferred in standards development in order not to delay publication

• Possible role of Open source
  − Reference Implementations of containers and components
  − Additionally require conformance test specifications.
  − Open Source test suites
  − Difficult to control partial and extended implementations
  − Project maintenance

• Software layering is never perfect: changes in lower layers do impact and can obsolete upper layer implementations.
Current State & Future Work

• Some progress in software standard specification & testing
  − UML 2.0 (OMG)
  − TTCN-3 (ETSI/ITU) – black box testing
  − Model Driven Architecture
  − Container SDK support for testing

• Much work still needs to be done

• Example of a successful approach
  − MHP Multimedia Home Platform (DVB Project)
  − Standard, test suites and implementations create a positive feedback loop
  − ETSI as custodian
Conclusion

- Ensuring end to end horizontal and vertical software interoperability with open standards presents new challenges:
  - Complexity
  - Evolving
  - Will require continued constructive dialogue between
    - Standard development organisations
    - Technology suppliers
    - Equipment Manufacturers
    - Users
    - Regulators
- Good conformance test specifications a prerequisite for software interoperability
- Standards and conformance test specifications developed together
- Software conformance testing technology & methodologies not mature
- Open Source can help
- ETSI has a significant role to play