Component Software & TP

Robert Atkinson
Microsoft Corporation
HPTS September 1995

Component Software is the Nirvana We All Seek

- Off the shelf software that you can buy instead of build
- A commercial marketplace of reusable components
  - Software production and distribution *within* a large organization has the same dynamics and forces
- Call it "objects" if you like, but a rose...

Component Software is a Reality Today

- The fundamental issues are well understood
  - Lots of people make their living at it right now
  - Not yet in the TP domain, but in other domains
- TP is *just* another domain
  - Some things different
  - Many things the same

Desktop Document Component Domain

- > 600 commercial applications, >350 Vendors
  - including *all* major commercial desktop applications
- > 35,000,000 installed base
- ≥ 200,000 units / month of MS Office
- "In business since 1991"

GUI Widget Component Domain

- Independently produced shrink wrap application components, mix'n'match
- OLE Controls
  - 250 commercially produced OLE Controls shipping today, largely by companies solely in that business
  - Dozens of OLE Control aware tools
- VBX Controls (the previous generation of technology)
  - > 500 *companies* in the VBX business

Secondary Phenomena Arising

- Specialized component software distribution channels
  - e.g.: "OLE Broker" online retail channel for OLE Controls and OLE Control related tools
- People whose livelihood depends not on writing software, not on writing reused software, but on the marketing of others writing reused software

Other Existing Commercial Component Software Domains

- WOSA XRT financial market data
- OLE for Design and Modeling (3D CAD)
- OLE for Point of Sale
- Healthcare, insurance, financial services...

TP is Just Another Domain of Component Software

- It shares in common the fundamental principles that engender a component software marketplace
- It has of course unique special issues and requirements, as does any domain
- If you don't address the former, the latter doesn't matter
- Remainder of this talk explores lessons we've learned about the former

Some Mechanics of Commercial Software

- Need to be able to ship new versions of my product: "Asynchronous Deployment"
  - With enhanced functionality
  - Without breaking my existing installed clients
- Example
  - Compound document components distributed through traditional channels (Egghead, ...)
- In enterprises, Asynchronous Deployment is the norm, not the exception

Implications of Versioning

- Implies crisp separation of what the clients can rely on from me from what they cannot
  - How else do I know what invariants to maintain?
  - Interface (contracts) vs. implementation
- Implies architectural separation of specification of enhanced functionality from old functionality
  - Along with a negotiation infrastructure so that mixes of old / new clients with old / new components work at highest functional level

More Implications of Versioning

- Implies freedom to innovate in functionality
  - No central committee coordination required to invent new interoperability standards
  - Anyone can do it in a "first class" way
- Support for "three-body problems" required
  - More than just the instantiating client needs to be able to do a version check

Commercial Software is a Binary Industry

- Source code contains trade secrets
- Source code leaks implementation details
  - Makes versioning impossible
- The mechanics of source code distribution and integration are complicated and fragile
  - Integration with “make” files, header files, etc.

Commercial Software is a Binary Industry

- Source code is coupled to particular languages and development tools
  - Implies consumer and provider using same tools
  - In tension with innovation and competition in the development tools industry

Development Tools & Binary Components

- Component interoperability layer beneath the tools layer
  - A rich highly functional level of interoperability
- Tools then compete to "make it easy"
  - Focus on particular customers
  - Don't sacrifice interoperability in the process
- Contrast with LCD interop approach

Virtualization of Legacy Systems

- "I can wrap my old thing and make it appear to be first class in this new world"
  - Component suppliers
  - Component consumers
- Crucial for commercial acceptance
- Implications for architectural separation from tools
  - No assumption of common implementation base

Conclusion

- Component software is not what the industry is striving for
  - It is here already, today
- Retention of key "Right to Innovate" principles from the non-component commercial industry was crucial
- Same principles apply to new domains of component software

Uniformity of Architecture Under Scale

- Uniform underlying programming model for large and small objects is achievable and beneficial
  - Size, Time, Spatial locality, Quantity, ...
- No brick wall compartmentalization of components
- No brick wall learning and stylistic curves
  - E.g.: online vs. off-line work