Internet Software – Product or Service?

- Thoughts on Internet Software Engineering -

Dr. Raymond A. Paul
Raymond.Paul@osd.mil

1

Outline

- Software as a service – an emerging model
- Impact on the practice of software engineering
- Impact on software business
- Impact on Software Applications
- Conclusion

2

Software as a service – an emerging model

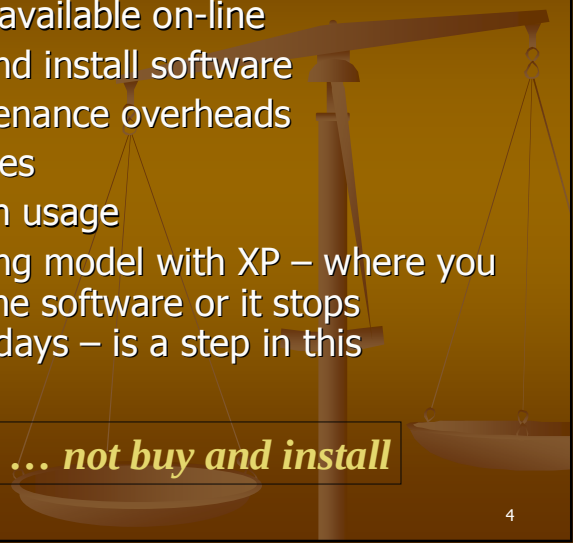
- Service model of software
- Leveraging existing services

3

Service model of software

- Web services are available on-line
- No need to buy and install software
- No need of maintenance overheads
- Automatic upgrades
- Payment based on usage
- Microsoft's licensing model with XP – where you have to register the software or it stops functioning in 30 days – is a step in this direction

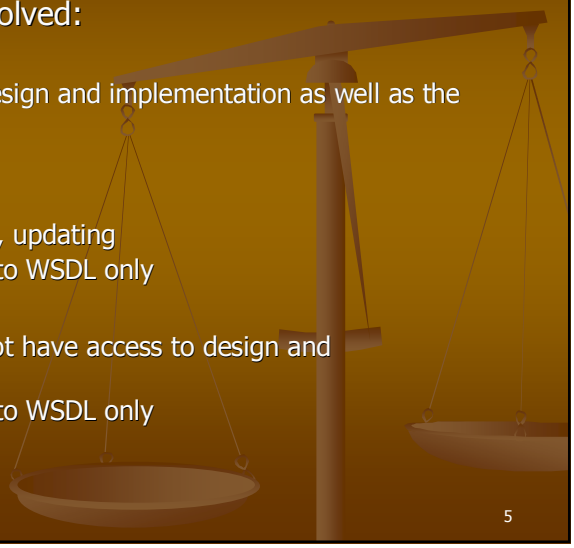
Use and pay … not buy and install



4

A New Computation Model (continued)

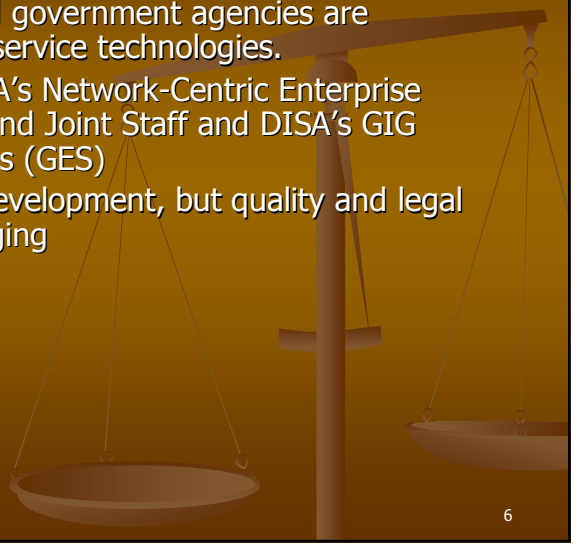
- Three parties are involved:
  - Service providers
    - Have access to design and implementation as well as the interface WSDL
    - May host services
  - UDDI
    - Provide searching, updating
    - May have access to WSDL only
  - Clients
    - Customer, may not have access to design and implementation
    - May have access to WSDL only



5

Current Status

- Many companies and government agencies are embracing the Web service technologies.
  - For example, DISA's Network-Centric Enterprise Services (NCES) and Joint Staff and DISA's GIG Enterprise Services (GES)
- Lots of technology development, but quality and legal issues are just emerging
  - Security
  - Reliability
  - Performance
  - Safety



6

Leveraging existing services

- Most web based services are built upon other web based services
- Example 1: Citibank's integrated portfolio service (www.myciti.com) built on top of Yodlee's account aggregation service (www.yodlee.com)
- Example 2: Multi-airline comparison-shopping air ticket service Orbitz (www.orbitz.com) built on top of on-line services of various airlines
- No need to build components that can be leveraged in a "frictionless" manner – pure "Keynesian specialization" at work

7

Impact on the practice of software engineering

- Knowledge of existing services is as important as ability to build a new one
- Loosely coupled service interaction → detailed API knowledge not needed
- The critical role of architecture
- Organization building software often must also use it
- QA and reliability requirements of Web services

8

Knowledge of existing services is as important as ability to build one

- The ability to code well is needed to build your own software
- The need for speed in rolling out new features requires being up-to-date with other related services and potential partners
- Standard 'competitive landscape analysis' needs to be augmented with 'collaborative landscape analysis'
- Communication between engineering and product marketing needs to be much higher bandwidth than passing around of requirements documents and feature lists

9

Loosely coupled service interactions → detailed API knowledge not needed

- Being abreast of the infrastructure provider's application programming interface (API), and being ahead of the competition's knowledge of the same has been a holy grail of engineering teams
- 'Join development agreement' and 'advance release API' have been the norm
- With interaction across services limited to a http requests, why is there a need for any of this?

10

The critical role of architecture

- Need flexibility and agility to rearrange components, change other services to partner with, replace a self-developed component with that of a partner ➔ good architecture
- Need ability to scale up ➔ good architecture
- Need ability to continuously improve availability, security, etc. ➔ good architecture
- Clearly software architecture plays a crucial role ➔ CTO's must be chief architects in addition to being technology evangelists

11

Organization building software often must also use it

- The organization building the service is also the one running it
- The operations team is located right next to the development team
- High bandwidth, low latency communication between ops and engineering, can often be acrimonious
- Very short bug fix and release cycle – sometimes excessively so

12

Reliability and QA Requirements of Web services

- 'Shrink wrapped & shipped' software must go through a formal and rigorous reliability and quality assurance (QA) process
- When the only running instance of the software is internal, the reliability and QA process is informal at best
- Significant back-and-forth make engineering development and QA interspersed
- Relationship between development and QA processes (and management) needs to be redefined

13

Impact on software business

- Focus on core competency
- Life-long learning is important
- Web services reliability – a key operational issue
- Mutual inter-dependence – benefits and vulnerabilities
- Managing web services vulnerabilities
- Conclusion

14

Focus on core competency

- An innovative company wants to create a new building for which the bricks don't exist today
- So, it ends up creating the bricks itself; e.g. the Obidos application server of amazon.com
- When others see the new building, they want to build one too, which creates a market for bricks
- Enter the brick builders, who develop the best type of bricks, e.g. Broadvision, Vignette, ART, Weblogic – all with generic application servers
- Amazon.com is left dependent on its 'crumbling bricks' at great operational and financial expense

15

Life long learning is important

- The tools and technologies are changing at an ever increasing pace
- Software engineers can get out-moded quickly
- Need to have short 'technical vitality' refresher courses for engineering
- Need to inculcate the necessity of continuous learning at undergraduate level

16

Organizational inter-dependence

- Cross-organizational cooperation will be needed not just for software development, but for the operation of the service
- Example: many Web-based services are dependent on Exodus for hosting, whose failure will bring all of them down; it's like the phone service or electricity utility
- Need to build better cooperation and collaboration – the Japanese 'keiretsu' model?

17

Benefits of cooperation

- Service interoperability requires good inter-organization cooperation
- Competitive mud-slinging needs to be dropped in favor of cooperative betterment
- The 'waffle cone' story
  - Hot summer day on Santa Cruz boardwalk
  - Ice-cream vendor ran out of cups; waffles weren't selling
  - The vendors cooperated to create the 'waffle cone'
- Need for a well-defined service level agreement (SLA), and a 'kinder and gentler' approach to its implementation

18

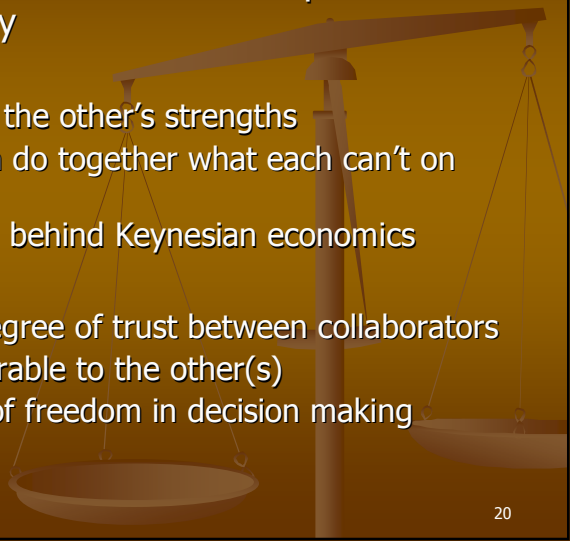
Impact on Software Applications

- Mutual Interdependence
- Reliability
- Vulnerabilities

19

Mutual Interdependence – Virtues and Pitfalls

- "Interdependence is better than independence" – Stephen P. Covey
- Certainly, because
  - Each can leverage the other's strengths
  - Two (or more) can do together what each can't on his own
  - The basic principle behind Keynesian economics
- However, it also
  - Requires a high degree of trust between collaborators
  - Leaves each vulnerable to the other(s)
  - Reduces degrees of freedom in decision making



20

Mutual Interdependence in Web Services

- Web Services based applications
  - By their very nature are mutually interdependent, since
    - Applications are built by leveraging the services provide by other applications;
  - However
    - Smooth operation of an application, and its ability to fulfill the needs of its clients satisfactorily, depends in turn on the smooth operations of other applications whose services it is using.
  - Hence
    - There is a need to have policies, mechanisms, and agreements in place to manage the risks associated with the added vulnerability

21

Web Services Reliability

- Various aspects of reliability
  - Service availability – what fraction of time is the service available
  - Service quality (QoS) – quality of the service provided
    - Timeliness
    - Precision
    - Accuracy
  - Graceful degradation & recovery – how well does the service degrade and recovers
  - Non-repudiation & dispute handling – how well does the service provider handle agreements in case of a dispute

22

Evaluating Web services at Runtime in Real Time

- Traditional 'Shrink wrapped & shipped' software has its own QA process, and integration with other software is either limited or impossible.
- Web services are different. It interacts with other software frequently and extensively, and it is necessary to evaluate its quality at runtime in real time.
- First, what are attributes of quality for web services?
  - Reliability – the service will not crash
  - Performance – the service will return results rapidly
  - Security – the service will not leak sensitive data to 3rd parties and it will not return false, malicious information back to the client
  - Safety – the service will not harm its environment

23

Evaluate "Reliability" of Services at Runtime

- Can we test a *foreign* web services in real time and at runtime?
  - Functional testing: Can we generate the test cases/scripts for a foreign service?
  - Coverage analysis: What kinds of coverage can we have?
  - Test evaluation and monitoring: how can we collect and evaluate test results including security and scalability test results?
- Can we have reliability models for web services?

24

Managing Web Services Vulnerabilities

- A well-defined service level agreement (SLA) between web services provider and client
- Specific clauses defining reliability, availability, QoS, escalation, etc.
- Have a 'test drive' stage in the relationship before entering long-term agreements
- Have a 'trust but verify' clause as part of the agreement, to ensure honesty and transparency on all sides
- Potentially have a financial stake in the service provider to ensure its 'good behavior' – the Japanese 'keiretsu' model
- Have backup plans to handle emergencies and disasters – e.g. backup data centers helped financial institutions to resume operations within a week of 9/11

25

Conclusion

- "Software as a service" is the model emerging for Internet software
- This has impact on
  - Role of engineers vis-à-vis product marketing
  - Software development process
  - Intra- and inter-organizational interaction
- Action needed on
  - Updating software/systems engineering programs to incorporate these changes
  - Developing new software development, reliability, security, QA, and operations processes and methods
  - Developing new models for intra- and inter-organizational interaction

26

Conclusion (continued)

- Quality issues of Web services are just getting attention recently.
- Need a new and innovative approach for quality assurance, many traditional ways of quality assurance need to be changed
  - From IV&V to CV&V
  - From independence to inter-dependence
  - Not only technology issues need to be changed, organizational and cultural issues must be addressed.

27

Welcome to the brave new world of Internet software engineering!!

28