The Conundrum of Distributed Systems

ICDCS 2003, Providence, RI

Dr. Alfred Z. Spector
Vice President, Services & Software Research
IBM Corporation

aspector@us.ibm.com
May 21, 2003
Abstract

We have been engineering distributed computing architectures according to roughly the same formulas for quite a number of years; yet each time, the end result has had greater complexity. Modern distributed computing architectures have an enormous number of protocols, layers, configuration parameters, APIs, etc. While we seem to be proceeding in the right direction in functionality, the growth in complexity is perplexing and has the potential to slow progress in research and in commercial use. I'll discuss this topic and propose areas of research that are relevant to resolving the issue.
IBM Research Division

ICDCS 2003, Providence, RI

aspector@us.ibm.com
5/22/03
Outline

- Distributed Systems
- A Historical Perspective
- Web Services
- Programming and Systems
- The Complexity Challenge
  - Aka the Conundrum of Systems
  - What is complexity?
  - Why are our systems too complex?
  - What could Computer Science do to measure & solve?
- Future Challenges
Distributed Systems

Three definitions:

- A computer science discipline aimed at reducing the cost of developing & managing applications that utilize multiple, networked systems
- Particular technology (infrastructure, specifications, or software) aimed at reduced development & management cost
- A complete system that implements a particular distributed application
3 Dimensions of Distributed Systems
3 Dimensions of Distributed Systems

The Distributed Systems Cube

Function Shipping

Data Sharing

Programming Lang’s & Tools

Infrastructure Runtime

Lifecycle & Manage
## Architecture

<table>
<thead>
<tr>
<th>Function Shipping</th>
<th>Data Sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages:</strong></td>
<td><strong>Ease of Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>– OO Nature</td>
<td>– Predefined capabilities</td>
</tr>
<tr>
<td>– Flexibility</td>
<td></td>
</tr>
<tr>
<td><strong>Examples:</strong></td>
<td><strong>Examples:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>– DCE, Corba, RMI, DCOM</td>
<td>– Distr. File Systems</td>
</tr>
<tr>
<td>– Eventual Web Appeal</td>
<td>– Distr. Database</td>
</tr>
<tr>
<td></td>
<td>– Initial Web Appeal</td>
</tr>
</tbody>
</table>

- Note: There really is a continuum
  - Databases have stored procedure
  - Object-oriented systems have dual flavors
- The Web today utilizes both
## Goals

<table>
<thead>
<tr>
<th>Create</th>
<th>Lifecycle &amp; Manageability</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Primary focus of academic computer science</td>
<td></td>
</tr>
<tr>
<td>- Plethora of techniques now required</td>
<td></td>
</tr>
<tr>
<td>- Blending of runtime &amp; tools</td>
<td></td>
</tr>
<tr>
<td>- The greater cost</td>
<td></td>
</tr>
<tr>
<td>- Plethora of tools now required</td>
<td></td>
</tr>
<tr>
<td>- Blending of runtime &amp; tools</td>
<td></td>
</tr>
</tbody>
</table>

- Note: This research community has tended to focus on dev’t
- Lifecycle & Manageability are at least equal problem
- Albeit, proper development is also key to manageability
- *Autonomic Computing* must be a greater academic focus
## Technology

<table>
<thead>
<tr>
<th>Programming Langs &amp; Tools</th>
<th>Runtime Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditionally, a community unto itself</td>
<td>Traditionally, a community unto itself</td>
</tr>
<tr>
<td>Often overly focused on programming issues, while ignoring essential distributed semantics</td>
<td>Not as focused on programming, but rather on algorithms, architecture &amp; semantics</td>
</tr>
</tbody>
</table>

- Note these are hard problems:
  - Programming gets far harder if one tries to include issues adaptability, reliability and availability, concurrency, …
  - Runtime infrastructures can be remarkably difficult to use
A Historical Perspective

- Distributed computing fundamentals known for at least 25 years
- Early great papers & collections – some favorites:
  - Using Encryption for Authentication in Large Networks of Computers, R. Needham and M. Schroeder, CACM, 21(12), 12/78

- The Distributed Systems summer course 1982-1989

aspector@us.ibm.com

IBM®
1980’s Western Institute of Computer Science
Distributed Systems Course Topics

- A. Spector / D. Gifford / R. Rashid, (J. Gray replaced Spector in late 80’s)
- Threading
- Function shipping: Message Passing & Remote Procedure Call
- Authentication, authorization, privacy
- Integrity/availability
  - Transactions
  - Distributed Replication
- Name Service, Time Service, …
- Data sharing techniques, and caching
- Related programming language issues
Results of Our Research?

- Many attempts to create a successful world-wide distributed computing standard
  - Mach Projects
  - DCE
  - CORBA/IIOP
  - DCOM
  - RMI
  - And more…, but they did not have desired, universal impact

- HTTP & HTML with DNS had impact!
  - *Not* based on breadth of distributed computing research
  - And yet, wildly successful!

aspector@us.ibm.com

5/22/03
Why Limited Impact of “Correct” Systems?

- Providing a *complete* programming, operating, & management environment for distributed computing is remarkably difficult.
- There was then minimal economic incentive to extend technology and & work out details.
- The technology was not fully understood.
- The technical community tended to fragment along traditional lines.
Why the Great Impact of HTML/HTTP?

- Simple goals → an elegant implementation
  - No security implied, paradoxically, lessened deployment resistance
  - Simple models supported initial use with few, if any, tools
  - Retooling of programs not required
- Significant economic value realizable without massive expense
- Worked with extant technology
- Limited need for a concerted effort by worldwide technical community
Limitations of HTTP/HTML

- Data presentation formats focused on presentation, not meaning
- Implied programming model ad hoc
  - Difficult to use state-of-the-art programming models
  - In fact, much programming a throwback to the past
- Weak security
- Behind the curve in supporting multi-domain services models
Enter Web Services

- Standards addressing many aspects of the cube
  - Greater focus on heterogeneity, than previously achieved
    - Support of multiple communication protocols
    - Support for arbitrary object hierarchies
  - Seamless, extensible data model
    - From storage abstraction
    - To communication standard
  - Breadth of focus
  - Supported by economic incentive

- Protocol and Programming Model, alike
Web Services (partial picture)

- BPEL4WS
- Reliable Messaging
- Security
- Transactions
- Coordination
- WSDL, UDDI, Inspection
- SOAP (Logical Messaging)
- XML, XSL, … (Encoding)
- Other protocols
- Other services
- Transports
- Quality of Service
- Policy
- Description
- Messaging
- Transport

aspector@us.ibm.com
5/22/03
The Good News

- There is greater care and creativity in Web Services than in previous approaches:
  - E.g. Service Level Agreement
    - Distributed systems research traditionally discussed transparency of services
    - In reality, transparency of services not exactly desired: Service Level Guarantees are desired
  - E.g., The world is inevitably heterogeneous at one or more levels
  - E.g., XML is well-grounded, audacious, & creative
  - E.g., Coming security standards are more complete
Some Observations (2)

- Web Services stds have significant complexity and success requires endurance
  - Performance not trivially achieved
  - Breadth of standards are not easily implemented
  - Complex integration required with other worlds; e.g., J2EE, Windows
  - Architecture and collection of requisite standards still growing in size
  - Most importantly, great tools are needed (see next slides...)

aspector@us.ibm.com
5/22/03
public class UpperCase {

    /*
     * This method converts the input parameter string to uppercase and returns it to the caller
     */

    public String convertToUpperCase(String inputString) {
        if (inputString != null) {
            return inputString.toUpperCase();
        } else {
            return null;
        }
    }
}
Deploying Web Service From WebSphere Studio
Web Service Creation Step

Namespace

SOAP deployment descriptor

Web Service Descriptions

Data Mapping
Web Service has been deployed. Studio integrates Client-server for immediate testing.

Service Proxy generated automatically – simplifies usage.

Invoke the Proxy for Service

Test Client

Results Window
The Tools Dimension

Many more types of tools will be required to make Web Services easy enough:

- Service modeling and definition
- Deployment
- XML
  - Data modeling and definition
  - Transformation
- Debugging
- Performance evaluation
- Quality of service specification
- Tools specialized for particular customer segments
  - Industry-specific schemas will often have their own tools

⇒ Quality tools will be required
Example: Eclipse tools

www.eclipse.org

- Because of the diversity and creativity required, an open-source development community has great potential
- Eclipse Project has built a universal platform for integrating development tools
- Its open, extensible architecture is based on plug-ins
- Brought to market by commercial offerings
- With source licensed for royalty free world-wide distribution
Eclipse Overview

Eclipse Platform
- Workbench
  - JFace
- SWT
- Workspace

Platform Runtime

Eclipse Project
- Java Development Tools (JDT)
- Plug-in Development Environment (PDE)

Team
Help
Debug

Another Tool
Your Tool
Their Tool

aspector@us.ibm.com
5/22/03
The Bad News is Complexity:

The Conundrum of Distributed Systems
Distributed Software Systems Today

- Score high on most metrics:
  - Amount of code
  - # of dependencies
  - # of programmatic interfaces
  - # of layers
  - Administrative interface size & configuration options
  - Non-uniformity
  - Non-orthogonality
  - Defects
  - Documentation
  - # of programmers involved
Anecdotes on Systems

- **Implementation Complexity:**
  - Windows XP Code is tens of millions of lines of code supposedly with circa $10^5 - 10^6$ bugs
  - Cisco Routers have support for more than 100 protocols

- **Usage (Administrative) Complexity:** Sendmail
  - Access.db, domaintable.db, local-host-names, mailertable.db, submit.cf, …
  - More than a page of Features, defines, etc for sendmail.cf
  - Longest O’Reilly book, at ~1200 pages

- **Usage (Use) Complexity:** New BMW 7 Series
  - “But why did those Bavarian motor masters have to ruin a wonderful driving machine by filling it with more gadgets and gizmos than you'd find in the cockpit of a space shuttle? The only thing intuitive is how to open the doors,” *Milwaukee Journal Sentinel*, 4/19/2002.
Example: Credit Card Processing

An application server typically supports:
- 5 Applications
- 10 EJBs
- Hundreds of servlets
- ~100 configuration parameters

A web server typically serves:
- Thousands of web artifacts
- ~20 configuration parameters

Failure protocols for each component are different: time-out, number of retries, where and what they log, how they fail.

* my credit card account

aspector@us.ibm.com

5/22/03
RosettaNet Purchase Orders

- There are 551 XML fields in the PurchaseOrderRequest
- There are 700 XML fields in the PurchaseOrderConfirmation

Excerpted First lines of purchase order confirmation:

```
fromRole.PartnerRoleDescription
  |--- ContactInformation
  |     |-- contactName.FreeFormText
  |     |-- EmailAddress
  |     |-- facsimileNumber.CommunicationsNumber
  |     |-- telephoneNumber.CommunicationsNumber
  |--- GlobalPartnerRoleClassificationCode
  |--- PartnerDescription
  |     |-- BusinessDescription
  |        |-- globalBusinessIdentifier
  |        |-- GlobalSupplyChainCode
  |--- GlobalPartnerClassificationCode
GlobalDocumentFunctionCode
PurchaseOrder
  |--- AccountDescription
  |     |-- accountName.FreeFormText
  |     |-- AccountNumber
  |--- billTo.PartnerDescription
  |     |-- BusinessDescription
  |        |-- globalBusinessIdentifier
  |        |-- GlobalSupplyChainCode
  |--- GlobalPartnerClassificationCode
  |--- ContactInformation
  |     |-- contactName.FreeFormText
  |     |-- EmailAddress
  |     |-- facsimileNumber.CommunicationsNumber
  |     |-- telephoneNumber.CommunicationsNumber
  |     |-- physicalLocation
  |        |-- GlobalLocationIdentifier
  |        |-- ProprietaryDomainIdentifier
  |        |-- ProprietaryIdentifierAuthority
```

Note: RosettaNet is a consortium of major companies working to create and implement industry-wide, open e-business process standards, that will form a common e-business language, globally aligning processes between supply chain partners. (From RosettaNet Home Page.)
The Cost of Complexity is Large

As hardware costs decrease, labor costs of I/T dominate

In storage, for example, labor costs already approach 3 times cost of hardware

Indirect costs of PCs may Soon approach 60% of total cost

IDC, June, 2001

(note: I/T spending data clearly wrong)

Peer support 47.0%

Casual learning 32.0%

Formal & self learning 4.0%

File & Data Mgmt 9.0%

Dev. of Personal Appl. 6.0%

Downtime 2.0%
Personal Experiences

- **XP:**
  - I have no idea if the ACLs on my home system are right
    - I would guess this is true of banks as well
  - Many XP applications don’t work with security turned on

- **Adventures with Linux-based email:**
  - 6 rules engines to configure (And, I missed one!)

- **Home automation**
  - Nice idea, but won’t the water leak detector shut-off water during a fire?
  - I’ve assigned circa 100 X-10 network addresses

- There are more than 150 (!) icons on my screen as I write this talk, with considerably varying behaviors

- The number of requests for PC personal administrative assistance are mind-boggling
What if The Home PC Were Truly Important

- If the Home PC were “mission-critical”, customers would want:
  - No data loss
  - High availability
  - Freedom from security problems
  - etc.

- What percentage of the population could possibly configure such a system?

- What percentage can afford the time to manage it?
Definition
3 Categories of Complexity

- *Classic Complexity*
  - Time
  - Space

- *Implementation Complexity*
  - Logical
  - Structural
  - Comprehensibility

- *Usage Complexity*
Usage Complexity Must Be Focus

- Consider a humble table (that is, legs and a horizontal surface)
  - *Classic Complexity* is not relevant as defined but there may be parallels
  - *Implementation Complexity* very high
    - Physicists do not fully understand tables, I suspect
  - *Usage Complexity* very low

- While Classic & Implementation Complexity may impact Usage Complexity, they are less important end goals. (In effect, they are tools of Computer Science.)
Dual Impact of Reliability

- Reliability requirements often add to complexity, if cost is a surrogate

- When systems are not reliable, anomalous failures drive us crazy

2001 Giga Information Group
based on cost estimates from service providers

![Bar chart showing relative cost vs. availability](image-url)
Wherefore Complexity
Causes of Complexity: The Problem Itself

- We aim to automate:
  - Complex problems
  - Ill-defined problems whose space changes over time & where we don’t understand fully the primary endpoint
  - Problems requiring human & extra-system adaptation

- Contrast to (structural) bridges:\(^1\):
  - Requirements straightforward and derivable from AASHTO specifications


aspector@us.ibm.com
5/22/03
Causes of Complexity: Design Methodology

- 2 related tenets of software methodology
  - Generalization, Encapsulation, Re-use
    - Complex things get more complex if they support re-use
    - E.g., a commercial product is 10x cost of a one-of solution
  - Integration of Components
    - (Don’t build new, assemble and connect)
    - This exposes many more interfaces
    - Boundaries are imperfect and “don’t connect”
    - Internal interfaces rarely recede into the woodwork
    - Component success may be based on context of use
    - E.g., “Business Process Integration” as a market category
  - These may contribute much to complexity, but it’s not clear what choice there is
Causes of Complexity: Taking Time

- Most students and most programmers want to start coding without adequate thought
  - Perhaps, due to a lack of confidence
  - Perhaps, due to the need to prove progress
  - Perhaps, due to the way they are trained or financially motivated

- Time to market considerations of industry
  - It’s better to be first than …

- The “Web Year” deemed a bad thing
Causes of Complexity: We

- Most computer scientists love complexity
  - We are proud of mastering it.
  - Criteria for my CMU 15-212 final assignments was maximal tenable complexity (I note to my defense, 20 years ago.)
  - And, we (!) design, and program systems

- We push what we invent without regard to the goal or primary endpoint

- *We Seem to Misinterpret Complexity as Sophistication*

- Where did ITS, the Incompatible Time Sharing System Originate?
  
  We are the fox guarding the henhouse
  We have met the enemy and he is us
Causes of Complexity:
Market Forces & Standards

- Market forces tend to promote “creeping featurism”:
  - Technical reviews & industry analysts do this, partially motivated by a need to have objective comparison metrics
  - Early adopters may revel in complexity

- With the desire to abstract and create reusable abstractions, standards groups tend towards standards that are unions
Silicon & Magnetic Technology Scaling

- A boon:
  - Makes everything possible

- A Curse:
  - Carelessness acceptable: Bloated code
  - A new layer every few years
  - Reduced motivation to discard the old
  - Perhaps, poor performance lead to complexity

- Perhaps, we would benefit from an end to easy scaling
No Clear Idea of Fitness to Task

- I think we are conflicted as a profession
- Perhaps the 150 icons on my screen were a good thing?
  - Perhaps, a “real hammer” is as complex
  - Our friends / colleagues are complex, so should be our computers
  - Good things take time to master
- On the other hand
  - Why should it take days to make computers to have applications perform takes we think we intuitively understand?
Old Code Does Not Ride Off Into The Sunset

- **Automobiles:**
  - It took close to 50-years to complete the transition to fuel injectors.
  - I’m aware of no cars that had both carburetors & fuel injectors simultaneously (although there were hybrid devices.)

- **But with software, we’d keep both**
  - There would be minimal extra expense
    - Except for the user interface & extra control logic
  - **Justification:**
    - We’d argue there were benefits to both
    - And then, there would be the widget that was dependent on the old interface
  - Admittedly, distributed systems/networks represent a particular dilemma
What Could We Do?
There are steps to take

- Meaning
- Measuring
- Methodology
- System Architecture
- Science and Technology
- Acknowledgment, Legal & Cultural Change

aspector@us.ibm.com

5/22/03
Meaning

- Computer scientists instantly know about time and space bounds
- It is just as important to arrive at clear definitions of all forms of complexity
- There has been work in this area, but we are likely to arrive at something like:
  - Classic Complexity
  - Implementation Complexity
  - Usage Complexity
- I note this topic is a very small part of the CS curriculum today
The Unmeasured Life Is Not Worth Leading

- If we can reach some definitions, we should try to create metrics
  - Minimization or Maximization adds focus and fun
  - Where metrics have existed in the field
    - Latency/Throughput
    - Word accuracy
    - Recall & precision
    - Translation quality
    More progress has been made

- The are risks to measuring things (you get what you measure)
  - I think metrics could be the strongest weapon against complexity
Methodology

- User-centered methods
- Ethnography
- Product Lines
- Increased use of metrics
- Component-based
- Sunset Clauses
Focus on The Right Function

- What do our user communities really want?
  - Can we more directly provide exactly that and dispense with distracting and wasteful items
  - Can we focus on the breadth of the problem and provide a solution to it, perhaps with incrementally more function
  - Perhaps, either directed or automatic adaptation to usage community required
System Architecture

- We need to have higher standards
- Example:
  - In systems today, we have disks, partitions, volumes, logical volumes, file systems, and directories structures
    - Do all of these still need to be visible interfaces?
    - Why not have a configuration option to set MBTF to Low, Medium, or High?
  - With ACLs, there could be far more useful profiles established
- What about increased use of classic AI techniques?
Science and Technology

- Autonomic computing concept: Making systems robust in the presence of stimuli occurring in different dimensions

![3D diagram of different dimensions: Attack, Failure, Load Variability, Small, Sparse, Highly malicious, Malicious, Catastrophic, Aggressive]
Autonomic Computing

- Subsystem design improved to eliminate manual control
- Core techniques:
  - Control theory
  - Increased use of rules systems; perhaps, with inference & common sense
  - Negotiation
- Standardization of event reporting to provide opportunities for data mining, statistical machine learning, and more feedback control

Architecture
Acknowledgment, Legal, & Cultural Change

- We should take public responsibility
- We should increase university and research focus
  - Education curriculum
  - Research agenda
  - Opportunity to broaden university collaborations
- We need to debate role of legal system
  - As we ever-more depend on computers, how do customers/society evaluate risk?
- Systems builders need to return more to artistry
Conclusions
Challenges: The Conundrum of Distributed Systems

- Complexity grows despite all we have done in computer science, from Simon’s *Sciences of the Artificial* to modern programming languages & software engineering techniques.
- There is valuable, rewarding, and concrete work for Computer Science in combating complexity:
  - Meaning
  - Measuring
  - Methodology
  - System Architecture
  - Science and Technology
  - Acknowledgment, Legal & Cultural Change
- This area of work could prove as valuable as direct functional innovation: It requires focus.
Challenges: Broadly Construed

- Distributed computing has been broadly understood for 25+ years
  - But, the quest requires more technology than had been initially thought
  - Web Services aimed at the breadth of the problem
  - Economic incentive finally exists
  - Complexity of core architecture is interesting
  - Seamless integration of tools and runtime essential
  - Proof of initial theses (e.g., from 1970s) finally to occur

- There are many key challenges remain
  - Quality of Service
  - Autonomic Computing
  - Components
  - Security
  - Scale
  - And Complexity

- Opportunities abound to change the world
  
  Distributed Systems become the World’s Operating System

aspector@us.ibm.com

5/22/03