Software Archeology & the Handbook of Software Architecture

Grady Booch
IBM Fellow
Software Archeology

- What it is/what it not/why we dig
- The process of archeology
- Architectural reconstruction
- Handbook of Software Architecture
- Preservation of classic software
How Much Software Exists In The World?

- SLOC is a measure of labor *(not a measure of value)*
  - Old code never dies *(you have to kill it)*
  - Some code is DOA
- Some assumptions
  - 1 SLOC = 1 semicolon
  - Number of software professionals worldwide
  - % of software professionals who cut code
  - SLOC/developer/year
  - $100/SLOC to develop
Number Of Software Professional Worldwide

Number of IT professionals worldwide

\[ y = -128.47x^3 + 12800x^2 - 59294x + 146623 \]

- Number of IT professionals worldwide (assumptions)
- Poly. (Number of IT professionals worldwide (assumptions))
% Of Software Professionals Who Cut Code

% of IT professionals worldwide who cut code

\[ y = 0.9075x + 0.7575 \]
New or modified source lines of code per year per developer

\[ y = -0.0328x^3 + 4.8392x^2 - 67.596x + 1062.8 \]
New Or Modified SLOC/Year And Cumulative

New or modified source lines of code per year per developer & cumulative

- New or modified source lines of code per year
- Cumulative source lines of code
Software Archeology

- The recovery of essential details about an existing system sufficient to reason about, fix, adapt, modify, harvest, and use that system itself or its parts.
Why We Dig

- To reason about
  - CAATS
- To fix
  - Y2K
- To adapt
  - Homeland Security
- To modify
  - JPL Mars Rovers
- To harvest
  - Patriot Missile System
- To use
  - AWACS Mid-term modernization
The Process of Archeology

- Study of the source code
- Reverse engineering
- Probing and other instrumentation
- Review of existing documents
- Interviews with tribal leaders
The process of archeology

- Most design information lives in tribal memory
- Typically there exists very high level architectural views and very low level design views, but little in between
- Reconstructing deployment and implementation views is easy
- Reconstructing the use case view is possible
- Reconstructing the logical and process views is hard
- Harvesting patterns is harder still
# The Process of Archeology

<table>
<thead>
<tr>
<th>Selection</th>
<th>First contact</th>
<th>Assessment</th>
<th>Review</th>
<th>Publish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a system</td>
<td>Create initial materials</td>
<td>Conduct an assessment</td>
<td>Revise materials</td>
<td>Publish model</td>
</tr>
<tr>
<td>Conduct an initial survey of the system</td>
<td></td>
<td>Conduct an assessment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Make first contact</td>
<td></td>
<td>Create description</td>
<td>Review materials</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Create manuscript</td>
<td></td>
</tr>
</tbody>
</table>

- **Selection**: Select a system, conduct an initial survey of the system.
- **First contact**: Create initial materials, make first contact.
- **Assessment**: Conduct an assessment.
- **Review**: Revise materials, review materials, create description, create manuscript.
- **Publish**: Publish model, search literature and web, establish benefits (visibility and assessment), meet virtually or face to face, evaluate permissions, offer registered access to site, identify other references, create a sketch.
Architectural Reconstruction

- **Architecture**
  - The fundamental organization of a system, embodied in its components, their relationships to each other, and the principles governing its design and evolution (IEEE Std 1471-2000, 2000, p. 3).

- **Stakeholder**
  - An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system (IEEE Std 1741-2000, 2000, p. 3).

- **Concern**
  - Those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability (IEEE Std 1471-2000, 2000, p. 4).

- **View**
Representing Software Architecture

Logical View
- End-user
  - Functionality

Implementation View
- Programmers
  - Configuration management

Use Case View
- Use Case

Process View
- System integrators
  - Performance
  - Scalability
  - Throughput

Deployment View
- System engineering
  - System topology
  - Communication
  - Provisioning

Conceptual
- Physical

Source: Kruchten, “The 4+1 Model View”
Architectural Views

- **Use case view**
  - The view of a system's architecture that encompasses the use cases that described the behavior of the system as seen by its end users and other external stakeholders.

- **Logical view**
  - The physical place where a system is developed, used, or deployed.

- **Process view**
  - The view of a system's architecture that encompasses the threads and processes that form the system's concurrency and synchronization mechanisms; a process view addresses the performance, scalability, and throughput of the system.

- **Implementation view**
  - The view of a system's architecture that encompasses the components used to assemble and release the physical system; an implementation view addresses the configuration management of the system's releases, made up of somewhat independent components that can be assembled in various well-structured ways to produce a running system.

- **Deployment view**
  - The view of a system's architecture that encompasses the nodes that form the system's hardware topology on which the system executes; a deployment view addresses the distribution, delivery, and installation of the parts that make up the physical system.
Barriers To Software Archeology

- Concerns over leakage of IP
- Accessibility of the development team
- Language
Architecture Metamodel
Architecture Metamodel
Air Traffic Control

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
ATM

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Battlefield Management

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Cargo Routing

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Computational Chemistry

![Diagram of Computational Chemistry System]

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Enterprise

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Games

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Games

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Google

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
MetLife

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Mobile Phone

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Mozilla

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Pathfinder

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Router

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Speech Recognition

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Washing Machine

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Web Server

Source: http://www.booch.com/architecture/architecture.jsp?part=Gallery
Eclipse

- www.eclipse.org
- Eclipse was started about 2 yrs go - when IBM made a $40M contribution to the main code base – but is now an independent entity
- The principal architects are John Wiegand, Dave Thomson, John Duimovich all part of the OTI team which jump-started Eclipse.
Eclipse Artifacts

- Eclipse Platform Technical Overview
- How to use the Eclipse API
- Eclipse Overview
- More detailed information exists for each of the subprojects.
Eclipse Architecture
Eclipse Use Case View

- Check In/Out Resource
- Close Perspective
- Close Window
- Display Help
- Invoke New Tool
- Open Perspective
- Open Window
- Refresh Workspace
- Shutdown Workbench
- Startup Workbench
# Eclipse Implementation View

<table>
<thead>
<tr>
<th>Core Libraries</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>ant.jar</td>
<td>servlets.jar</td>
</tr>
<tr>
<td>antrunner.jar</td>
<td>servlets-common.jar</td>
</tr>
<tr>
<td>antsupport.jar</td>
<td>servlets-default.jar</td>
</tr>
<tr>
<td>antsupportlib.jar</td>
<td>servlets-invoker.jar</td>
</tr>
<tr>
<td>appserver.jar</td>
<td>servlets-manager.jar</td>
</tr>
<tr>
<td>boot.jar</td>
<td>servlets-webdav.jar</td>
</tr>
<tr>
<td>bootstrap.jar</td>
<td>slimlauncher.jar</td>
</tr>
<tr>
<td>catalina.jar</td>
<td>snippetsupport.jar</td>
</tr>
<tr>
<td>commons-logging.jar</td>
<td>startup.jar</td>
</tr>
<tr>
<td>compare.jar</td>
<td>swt.jar</td>
</tr>
<tr>
<td>cvs.jar</td>
<td>team.jar</td>
</tr>
<tr>
<td>dcore.jar</td>
<td>teamcvsssh.jar</td>
</tr>
<tr>
<td>dtd.jar</td>
<td>teamecv sui.jar</td>
</tr>
<tr>
<td>editors.jar</td>
<td>tomcat-coyote.jar</td>
</tr>
<tr>
<td>externaltools.jar</td>
<td>tomcat-httpd.jar</td>
</tr>
<tr>
<td>forms.jar</td>
<td>tomcat-utility.jar</td>
</tr>
<tr>
<td>help.jar</td>
<td>tomcatwrapper.jar</td>
</tr>
<tr>
<td>helpworkbench.jar</td>
<td>ui.jar</td>
</tr>
<tr>
<td>helpworkbenchwin32.jar</td>
<td>updatecore.jar</td>
</tr>
<tr>
<td>jakarta-regexp-1.2.jar</td>
<td>update-servlets.jar</td>
</tr>
<tr>
<td>jasper-compiler.jar</td>
<td>updateui.jar</td>
</tr>
<tr>
<td>jasper-runtime.jar</td>
<td>update32.jar</td>
</tr>
<tr>
<td>jdt.jar</td>
<td>versioncheck.jar</td>
</tr>
<tr>
<td>jdtcore.jar</td>
<td>views.jar</td>
</tr>
<tr>
<td>jface.jar</td>
<td>webapp.jar</td>
</tr>
<tr>
<td>jfacetext.jar</td>
<td>workbench.jar</td>
</tr>
<tr>
<td>jsp.jar</td>
<td>workbenchwin32.jar</td>
</tr>
<tr>
<td>junit.jar</td>
<td>resource.jar</td>
</tr>
<tr>
<td>junitsupport.jar</td>
<td>resources-ant.jar</td>
</tr>
<tr>
<td>launching.jar</td>
<td>runtime.jar</td>
</tr>
<tr>
<td>launchingsupport.jar</td>
<td>search.jar</td>
</tr>
<tr>
<td>lucene-1.2.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>naming-common.jar</td>
<td>resources.jar</td>
</tr>
<tr>
<td>naming-factory.jar</td>
<td>resources-ant.jar</td>
</tr>
<tr>
<td>naming-resources.jar</td>
<td>runtime.jar</td>
</tr>
<tr>
<td>optional.jar</td>
<td>search.jar</td>
</tr>
<tr>
<td>parser.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pde.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pde-ant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdebuild.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdebuild-ant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdecore.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdeui.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdeuiant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jdt.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jdtCompilerAdapter.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jdtcore.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jface.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jfacetext.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jsp.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>junit.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>junitsupport.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>launching.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>launchingsupport.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>lucene-1.2.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>naming-common.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>naming-factory.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>naming-resources.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>optional.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>parser.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pde.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pde-ant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdebuild.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdebuild-ant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdecore.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdeui.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>pdeuiant.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jdt.jar</td>
<td>servlet.jar</td>
</tr>
<tr>
<td>jdtCompilerAdapter.jar</td>
<td>servlet.jar</td>
</tr>
</tbody>
</table>
Eclipse Logical View

- Plugin Development Environment (PDE)
- Workbench
- Team
- Debug
- Ant
- Help
- Java Development Tools (JDT)
PDE
PDE
Workbench

- Properties, Tasks, Outline, Navigator,...
- Resource, Debug,...

Platform

Perspective

Views

Editors

ExtensionPoints
Drag and Drop

```
public class DragSourceListener
{
    public DragSourceListener()
    {    }
    public void dragStart()
    {    }
    public void dragSetData()
    {    }
    public void dragFinished()
    {    }
}

public class DropTargetListener
{
    public DropTargetListener()
    {    }
    public void dragEnter()
    {    }
    public void dragLeave()
    {    }
    public void dragOperationChanged()
    {    }
    public void dragOver()
    {    }
    public void drop()
    {    }
    public void dropAccept()
    {    }
}

public class DragSourceAdapter
{
    public DragSourceAdapter()
    {    }
    public void dragStart()
    {    }
    public void dragFinished()
    {    }
    public void dragSetData()
    {    }
}

public class DropTargetAdapter
{
    public DropTargetAdapter()
    {    }
    public void dragEnter()
    {    }
    public void dragLeave()
    {    }
    public void dragOperationChanged()
    {    }
    public void dragOver()
    {    }
    public void drop()
    {    }
    public void dropAccept()
    {    }
}
```
Widgets
Debug
Ant
Core boot
Lessons Learned

- A lot of design information lives in tribal memory
- There were very high level architectural views and very low level design views, but little in between
- Reconstructing the deployment and implementation views are easy
- Reconstructing the use case view is possible
- Reconstructing the logical and process views are hard
- Harvesting patterns is harder still
Handbook of Software Architecture

- No architectural reference exists for software-intensive systems
- Goals of the handbook
  - Codify the architecture of a large collection of interesting software-intensive systems
  - Study these architectural patterns in the context of the engineering forces that shaped them
  - Satisfy my curiosity
IBM Research

- **Artificial Intelligence**
  - Asimo
  - Avida
  - Blondie24
  - CYC
  - Swarm
  - Trilogy

- **Commercial and Non-Profit**
  - Amazon
  - eBay
  - Home Depot
  - LDS
  - Library of Congress
  - Sabre
  - Starwood
  - Ticketmaster

- **Communications**
  - 5ESS
  - 911
  - Nokia

- **Content Authoring**
  - Avid
  - Massive
  - Microsoft Word
  - Photoshop
  - Renderman
  - Wall Street Journal
  - Yamaha

- **Development**
  - Eclipse
  - emacs
  - JIT

- **Devices**
  - Bernini Artista
  - CocaCola
  - Foveon camera
  - General Electric
  - Hamilton Automation
  - Otis
  - Suunto watch
  - Triton

- **Entertainment and Sports**
  - Disney Hall of the Presidents
  - Hong Kong Jockey Club
  - NBC control room
  - Olympics
  - Spiderman
  - Veri-Lite

- **Financial**
  - Fedline bond system
  - Great Plains
  - NYSE
  - Visa
  - Wells Fargo

- **Games**
  - Deep Blue
  - Doom III
  - StarCraft
  - The Sims
IBM Research

- Industrial
  - Dow Chemical
  - NERTO
  - Toyota

- Legal
  - Identix
  - Lexis/Nexis
  - Supreme Court

- Medical
  - Cogency
  - Medtronics
  - Philips
  - Siemens

- Utilities
  - AOL Messenger
  - Babblefish
  - Google
  - Groove
  - sendmail

- Military
  - AEGIS
  - AWACS
  - Bendix
  - Chyenne Mountain
  - F16
  - Force XXI
  - GPS
  - Pathfinder
  - Predator
  - Space Command

- Operating Systems
  - Linux
  - Palm OS
  - Wind River OS
  - Windows XP

- Platforms
  - .NET
  - J2EE

- Scientific
  - ABI Prism
  - Earth Weather Simula
  - Jason
  - Mars Exploration Rover
  - Mathematica
  - Mona Loa observatory
  - National Data Buoy Center
  - National Ignition Facility
  - NavTech
  - Protein Data Bank
  - SETI@home

- Transportation
  - BMW
  - British Rail
  - CAATS
  - Evans and Sutherland
  - Fedex
  - Ford
  - NuTech
  - OOCL
Adobe Photoshop
Second Life
Logical View

- **Asset System** - transport for animations, sounds, notecards, scripts, etc
- **Avatar Appearance** - appearance is constructed from a mesh, parameters to deform the mesh, and textures to describe clothing
- **Avatar Profiles·Culling** - octree and occlusion based culling to speed rendering
- **Error Logging System** - command and control of debugging output
- **Groups·Image System** - prioritizes and decodes JPEG2000 images into OpenGL textures
- **Texture cache** - used for reading and writing texture data to the local disk cache
Logical View

- **Image System** - prioritizes and decodes JPEG2000 images into OpenGL textures
  - **Texture cache** - used for reading and writing texture data to the local disk cache
  - **Image Pipeline** - fetches textures from the servers and decodes them

- **Inventory** - server-side storage of assets for each user

- **L$ System·Message System** - reliable and unreliable transport over UDP

- **Movie System** - QuickTime-based video on object surfaces

- **Muting Objects and Agents** - How the viewer manages the muted object list

- **Rendering System** - from viewer object to drawable to face to vertices to graphics
Logical View

- **Sound System** - cross platform audio based on FMOD
- **Tools** - anything that takes effect when you click your mouse on the world view.
- **UI Widgets** - cross platform buttons, scroll bars, etc.
- **UI Floaters** - dialogs and windows built from XML files
- **VFS** - cached data is held in two "virtual file systems" in large files on the client
- **Viewer Object System** - objects in scene for rendering and editing
- **Web Browser** - the integration of the viewer with the web.
Process View

- **Main thread** -- The input/output main program function (including rendering).
- **VFS thread** -- Thread responsible for reading/writing to the local virtual file system.
- **LFS thread** -- Thread responsible for some reading/writing to the local native file system.
- **Image thread** -- Thread responsible for requesting and decoding image data.
- **Error Thread** -- Thread responsible for catching exceptions, calling the (currently unused?) error handler, and retiring.
- **Worker Threads** -- Threads designed to do CPU intensive background tasks. These threads may be paused during rendering so as not to reduce performance (design in-progress).
Process View

1. Initialize - newview/viewer.cpp:: main()
2. Loop - newview/viewer.cpp:: main_loop()
   - Gathers keyboard and mouse input
   - Pumps the TCP i/o
   - idle()
   - Render the frame
   - Let filesystem and worker threads process
3. Shutdown
Preservation of Classic Software

- No comprehensive and intentional activity has yet been undertaken to preserve software artifacts
- There are a number of reasons to act now
  - Many of the authors of seminal systems are still alive
  - Many others may have the source code or design documents for these systems collecting dust in their offices or garages
  - Time is our enemy: over time, these artifacts will become lost forever
Goals

- Preserving such artifacts for future generations is more than a valuable historical curiosity
  - The understanding and codification of architectural patterns
  - The evolution of software architecture and how they were products of their time
  - A statement of prior art relevant to the issues of proving and disproving software patents

- Preserving such artifacts also provides raw materials for future generations of software archeologists, historians, and software developers who can learn from the past regarding
  - What worked and what didn't
  - What was brilliant and what was an utter failure
Metahistory

- I’ve been inspired by two groups:
  - Bruce Andersen (Handbook of Software Architecture)
  - Ward Cunningham (and the Hillside Group)
- Spring 2002
  - Survey of 500 of my closest friends
- Fall 2002
  - Museum board meeting
- Fall 2003
  - Workshop on preservation of software
- Current
  - Establishment of Software Collections Committee
I'm seeking your help in preserving classic software products before they are forever lost to future generations.

As Bjarne Stroustrup, the inventor of C++, has observed: “our civilization runs on computers.” There are a number of organizations - most notably the Computer Museum History Center - dedicated to preserving classic hardware, but it occurs to me (and others) that we should also preserve classic software as well.

There are a number of reasons to act now. Although many of the authors of such systems are still alive and many others may have the source code or design documents for these systems collecting dust in their offices or garages, time is our enemy and over time, these artifacts will become irrevocable. By the way, notice that I said “source code” and “design documents.” Executables are interesting only insofar as we have the machines upon which to run them (and preserving running hardware is an entirely different issue), but the code and designs that these executables manifest tell us much about the state of the software practice, the minds of their inventors, and the technical, social, and economic forces that shaped these products in their time. Preserving such artifacts for future generations is certainly a valuable historical curiosity, but they also offer a statement of prior art (relevant to the issues of proving/disproving software patents), the evolution of software architecture (and how they were products of their time), and the creation of architectural patterns (patterns are common solutions to common problems, and such patterns have emerged for particular domains and development cultures). Preserving such artifacts also provides the raw materials for future generations of software archeologists, historians, and software developers (the latter who can learn from the past regarding what worked and what didn’t as well as what was brilliant and what was an utter failure).

What constitutes a classic software product? I can only offer the subjective criteria that such a product is one that made a seminal technical, economic, or social difference. Lisp 1.5, the Mark I Time Sharing System, Smalltalk-72, Dijkstra’s THE operating system, Unix/Linux, and EMACS are all examples of technically significant products. DOS, Windows 3.1, VisiCalc, and the original Mac OS all had significant economic impact. Space War, Pong, Zork, sendmail, IM, and the original Netscape browser all wove themselves into the fabric of society.

I’m not affiliated with the Computer Museum, although, from my discussions with them, my intent is to initiate this effort but to deliver up all such artifacts to them for their preservation and care. I expect to be creative - some of you have or may know of people who have dusty card decks, listings, or disks that we’d want to scan, capture and otherwise preserve, while certain companies may have source code locked up in their vaults as precious intellectual property yet who might be open to the suggestion of putting that source code into some secure escrow so that 50 or even 100 years from now, at which time there would be zero value in that IP, we could open a time capsule of software.

You are receiving this email because you are an important contributor to the theory and/or practice of software. I’ve emailed about 500 such folks, ranging from Turing Award winners to CTOs/CEOs, and researchers. I apologize if I’ve spammed you, but I know of no better way to begin this effort than to connect with those people who have helped invent the present and will continue to invent the future.

To draw this message to a close, I’m only asking two things of you. At the very least, could you respond to the first? If you are so able, some help on the second would be of enormous value.

First, tell me what’s in your top ten list of classic software products. What software artifacts would you want to preserve for future generations, products that made a difference in shaping the software industry?

Second, please let me know if you have access to any such artifacts or if you know of colleagues who do. I’m aware of a few related efforts to keep old software running, and if you know of other such activities, please help me get in touch with their organizers. I’ll happily absorb card decks, listings, documents, napkins full of sketches, and the like. If your company owns the rights to some of these artifacts, let’s enter into a serious dialog about how we might put those artifacts into secure escrow to save them before your company discards them by intention, neglect, or merger/economic demise. If you are or know of the original architects of these classic software products, let’s also talk about capturing an oral history of their creation.

Thank you for your time in listening to my ranting. Personally, I’m doing this because the software industry has been good to me, and I simply want to give back to those who will drive our industry in the future.

Grady Booch
Chief Scientist
Rational Software Corporation
ebg@rational.com
Spring 2002

- Personal contacts
- Web contacts
- Classic software
- Actions
  - Conduct a workshop regarding the preservation, presentation, and archeology of classic software
  - Investigate forming a public escrow for software currently under IP protection
  - Make contact with Microsoft, Apple, and IBM archivists
  - Publish a call in IEEE Software (Steve McConnell)
  - Publish a call in the Annals of the History of Computing (Tim Bergen)
  - Publish a call to the owners of all Web contacts
  - Send an acknowledgement message to all personal contacts
Things Collected

- Altair Basic
- ThingLab (Alan Borning)
- Multics
- MacPaint
- Fortran compilers
- Whirlwind
- ...

...
The first four bytes on the stack are (or rather, should be) two return addresses. We're not interested in them, so the first thing to do is set HL to point to SP+4.

```
0192  21U400  GetFlowPtr  L#1 H,00004  HL=SP+4 (ie get word
0195  39      DAD SP     just past return addr)
```

Get the keyword ID, the byte that precedes the flow struct. Then we increment HL so it points to (what should be) the flow struct, and return if the keyword ID is not 'FOR'.

```
0196  7E      MOV A,H
0197  23      INX H
0198  FE81    CPI 81    'FOR'?
019A  C0      RNZ Return if not 'FOR'
```

Special treatment for FOR flow structs. Here we check that we've got the right one, ie the one required by the NEXT statement which called us.

When we're called by NEXT, it sets DE to point to the variable in the NEXT statement. So here we get the first word of the FOR flow struct which is the address of the FOR variable, and compare it to the one we've been given in DE. If they match, then we've found the flow struct wanted and we can safely return. If not then we jump 13 bytes up the stack - 13 bytes is the size of the FOR flow struct - and loop back to try again.

```
019E  F7      RET PushNextWord  PUSH (HL)
019C  E3      XTHL  POP HL (ie HL=(HL))
019D  E7      RET CompareHLDF HL==DE?
019E  010D00  LXI E,000D
01A1  E1      POP H    Restore HL
01A2  C8      RZ     Return if var ptrs match.
01A3  09      DAD E   HL+=000D
01A4  C39601  JMP GetFlowPtr+4 Loop
```
MacPaint

- Source code fell into my lap from Tim O’Reilly and Andy Herzfeld
- Legal/IP issues abound
- Next step is to conduct an interview
Fall 2002

- Grady Booch
- John Toole
- Dave Babcock (volunteer)
- Sharon Brunzell (librarian)
- Mike Williams (head curator)
- Ike Nassi (board member)
- Len Shustek (board chair)
- Sowmya Krishnaswamy (cyermuseum)
- Mike Walton (cyberexhibits)
- Lee Courtney (volunteer)
- Bob Supnik (via phone)
Present

- Al kosskow
  - Computer History Museum Software Curator
The Big Questions

- What systems do we preserve?
- What artifacts should we collect?
- How do we make these artifacts available?
- How can we create a sustainable program?