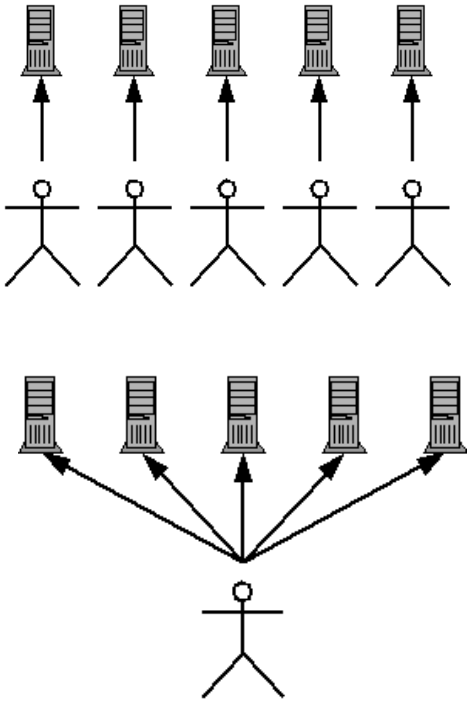
Capstone Presentation

Team Fugu

and the

OpenBSD Tools Project



Team Fugu: Cast of Characters

The Team

Ben Atkin

Thad Boyd

Erik Wilson

Nauman Qureshi

The Discipline

Computer Science

The Technical Advisor

Dr. Eck Doerry

The Sponsor

USGS Astrogeology Team

Ernest Bowman-Cisneros and Margaret Johnson

The Story Thus Far:

- What Is OpenBSD?
  - UNIX-based operating system
  - Open-source
  - Secure
- What's With the Fugu?

Thad Boyd

Fugu: A poisonous blowfish.

The blowfish is the OpenBSD mascot.

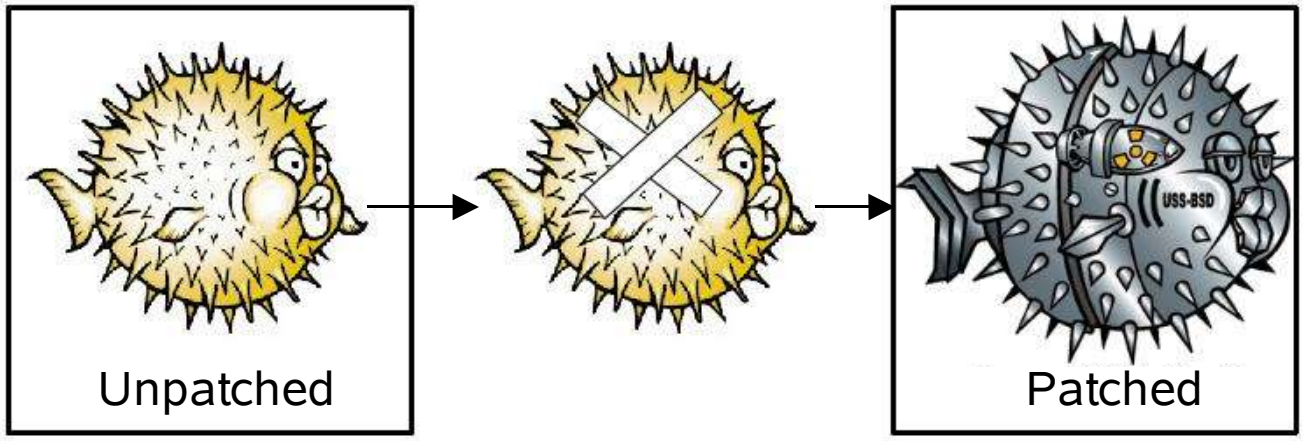
Fugu mascot designed by Jon Gardner.

The Client

- US Geological Survey (USGS)
  - Astrogeology Team
  - Map Landscape of Planets
    - Custom software for image processing
    - Using high-end UNIX workstations
  - Information Technology Division
    - Multiple Servers (Mail, FTP, Web)
    - Multiple Architectures (x86, Sparc)

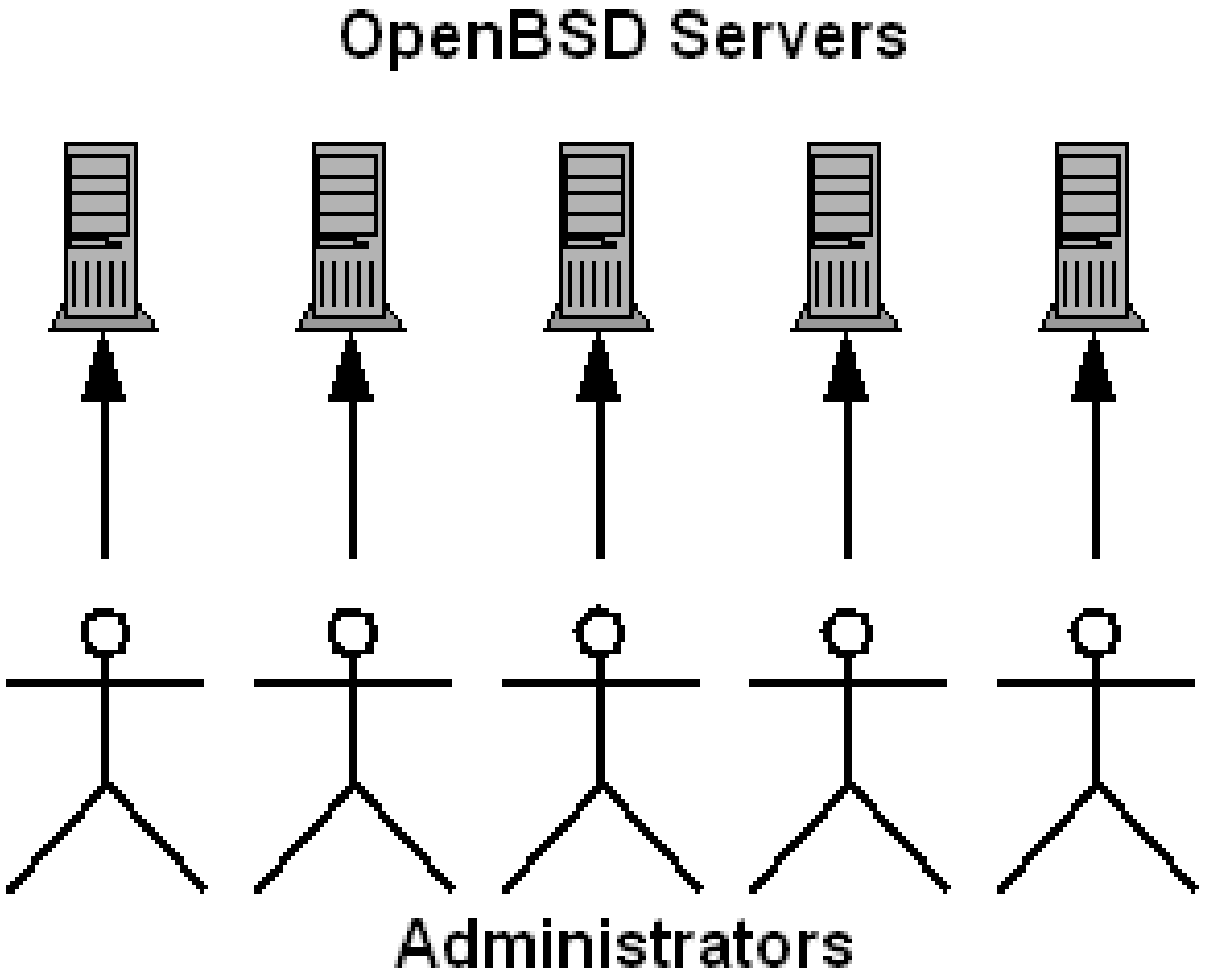
Problem

- Time-consuming to install OpenBSD on many systems
- Patches for OpenBSD require manual installation on each system



- **20 machines x (1 hour install + 1 hour patches) = *40 hours total***

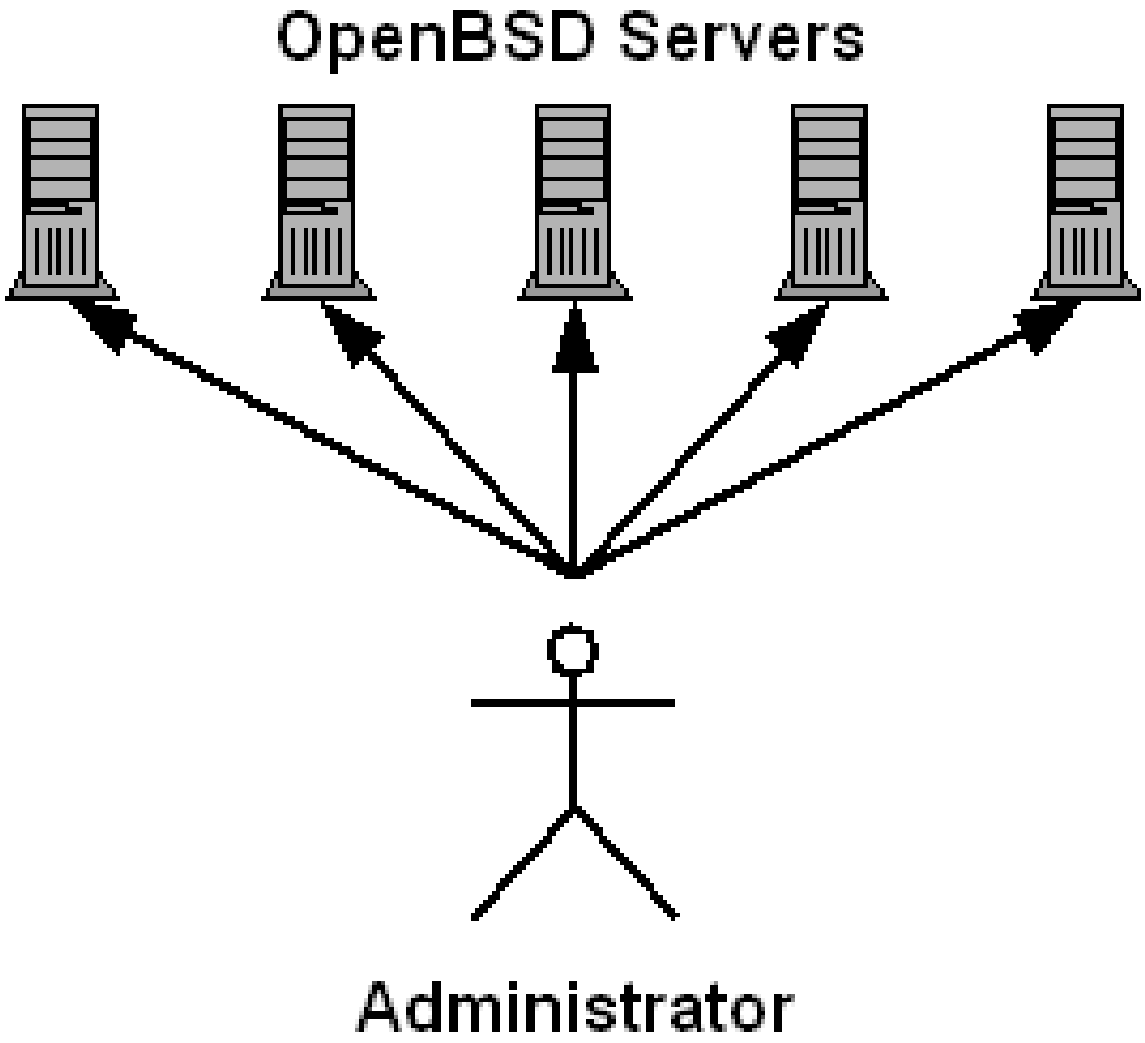
Diagram: Manual Maintenance



Needs

- **Two Projects**
- OpenBSD Auto-Installer
  - Need a non-interactive system
  - *Similar Products*:
    Solaris Jumpstart, Redhat Kickstart
- OpenBSD Auto-Patcher
  - Auto download and install of patches
  - Ability to “roll back” or uninstall patches
  - *Similar Product*: Tepatche
- *Both must run on Intel i386,*
  Sun Sparc64 platforms

Diagram: Automatic Maintenance



Installer Requirements

- Must be future-version compatible
- Must handle install *or* upgrade
- Install configuration file must be read from:
  - CDROM / Floppy
  - FTP / HTTP
  - Local hard drive
- Must handle partitioning of disks
- Must seek out and back up important files (eg SSH keys)

Patcher Requirements

- Must handle source or pre-compiled patches
- Must track what patches have been installed, and what patches have failed to be installed

Automated Installer

- Traditional Installer
  1. Boot installer
  2. **Answer questions**
  3. Reboot into installed system
- Automated Installer
  1. **Create configuration file**
  2. Boot installer
  3. Install is done automatically
  4. Reboot into installed system

Installer Configuration File

- Can be loaded from disk or network
- Contains information for
  - Network
  - Partitioning disks
  - Filesets
  - Pre-install script
  - Post-install script
- Designed to be user-friendly
  - Case insensitive (“disks” or “Disks”)
  - Divided into sections

Installer Configuration File

- Do not need to know specifics when making configuration file
- One configuration file used for computers with differences in hardware
  - Different device names
  - Different disk geometry
- Disk geometry
  - For security, there are separate filesystems for web, e-mail, documents
  - Filesystems should be organized to get best use of space

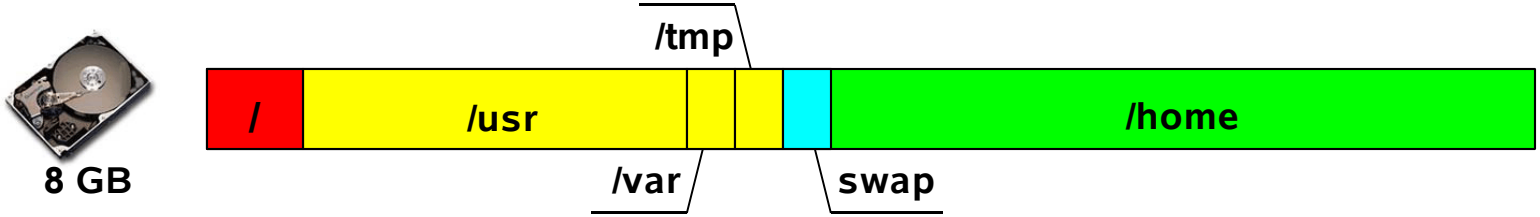
Installer: Disk Partitioning

- Partition a “class” of systems
- May contain one or two disks
- Configuration File:

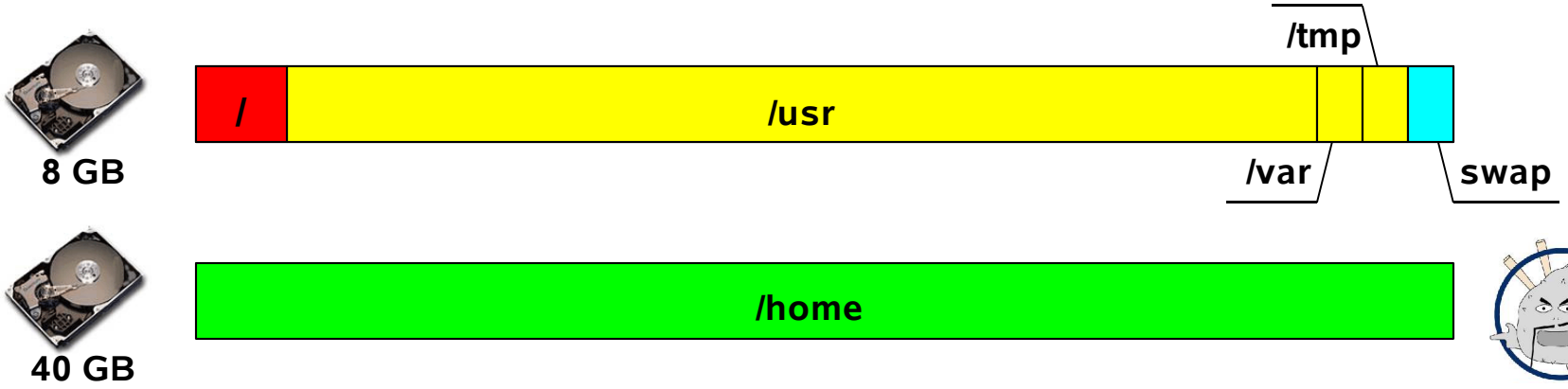
```
[Disks]
Disks=Main Homedisk
Main.Device=primary
Homedisk.Device=secondary primary
Main.Slices=root usr var tmp swap
Homedisk.Slices=home
extra=home usr
usr.min=2gb
```

Installer: Disk Partitioning

- System 1
  - One 8GB Hard Disk



- System 2
  - One 8GB Hard Disk
  - One 40GB Hard Disk

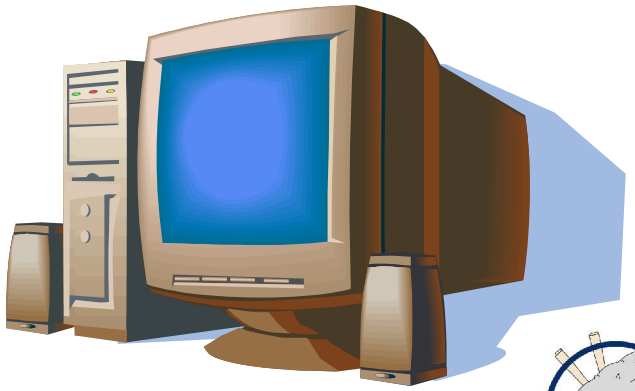


Architecture: Installer

- Based on existing Automated Installer scripts
- Additional subroutine files
  - disks.sub (disk partitioning)
  - util.sub (reading from configuration files)
- Only uses programs contained in Interactive Installer media (that can fit on a floppy)
- Coded in sh and sed
- Perl used for:
  - Configuration file validator
  - Online monitoring utility

Automated Installer: Features

- Allow options to be entered manually, upon request
- Works on i386, SPARC64
- Internet Monitoring
  - Simple web page for logs
- Configuration File Validator
- Build custom disk images



Screenshot

```
DHCPACK from 192.168.56.254
New Network Number: 192.168.56.0
New Broadcast Address: 192.168.56.255
bound to 192.168.56.131 -- renewal in 900 seconds.
No more interfaces to initialize.
DNS domain name? (e.g. 'bar.com') [localdomain] (timeout=1)
  - Press !Enter! for Manual Input -
DNS nameserver? (IP address or 'none') [192.168.56.2] (timeout=1)
  - Press !Enter! for Manual Input -
Use the nameserver now? [y] (timeout=1)
  - Press !Enter! for Manual Input -
Default route? (IP address, 'dhcp' or 'none') [dhcp] (timeout=1)
  - Press !Enter! for Manual Input -
Edit hosts with ed? [n] (timeout=1)
  - Press !Enter! for Manual Input -
Do you want to do any manual network configuration? [n] (timeout=1)
  - Press !Enter! for Manual Input -
Manually configure the disks? [n] (timeout=1)
  - Press !Enter! for Manual Input -
preparing wd0...
Putting all of wd0 into an active OpenBSD MBR partition (type 'A6')...done.
# using MBR partition 3: type A6 off 63 (0x3f) size 8385867 (0x7ff54b)
1 a 63 2448369 ffs /
2 b 2448432 449568 swap swap
_
```

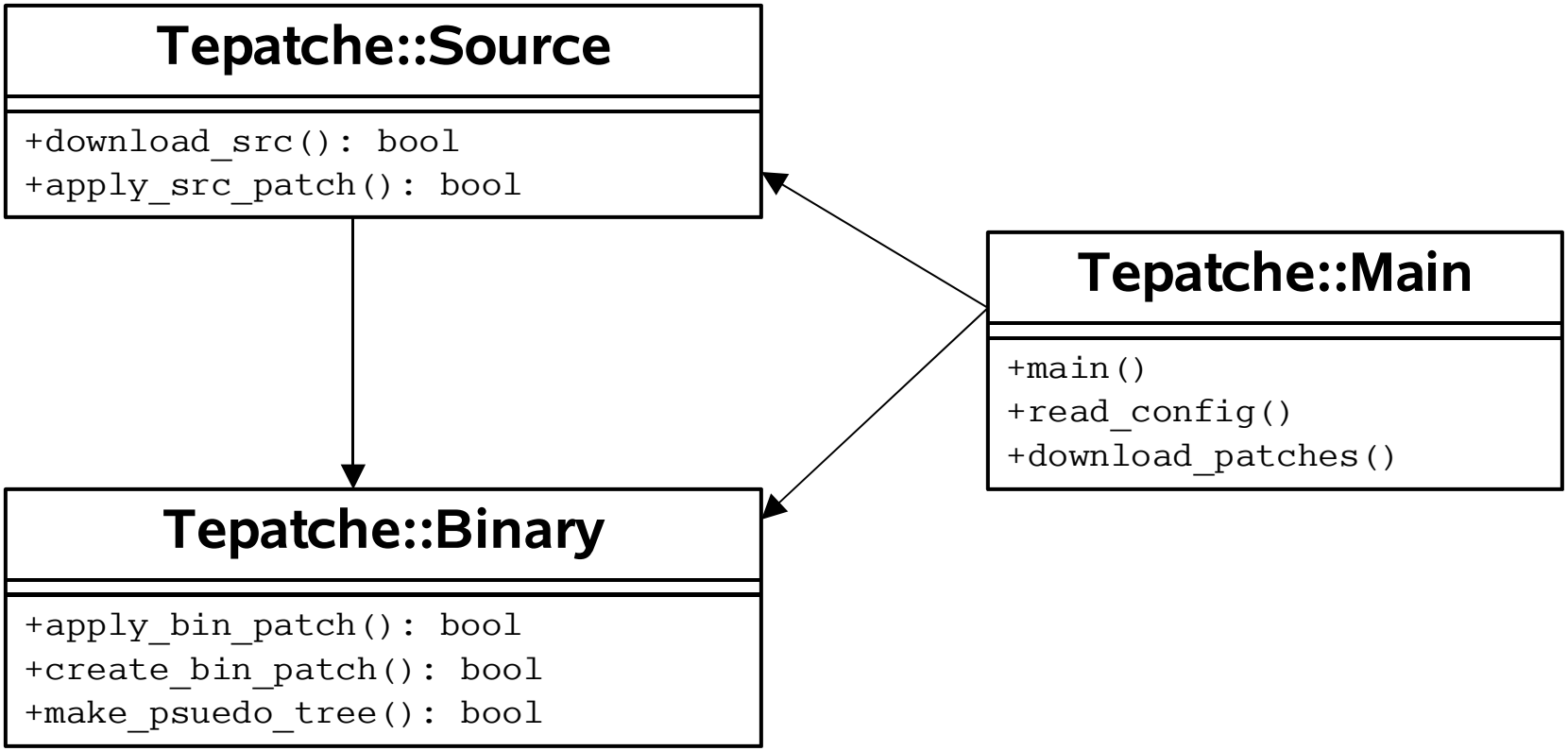
Tepatche

- Originally created at UNAM (Universidad Nacional Autonoma de México) by Gunnar Wolf
- Runs as a regularly scheduled task
- Checks for security patches on the Internet
- Source Patching
  - Downloads source
  - Compiles source into machine code
- Our task:
  - Contact Gunnar Wolf for implementation ideas
  - Add binary (machine code) patching ability
  - Make other needed improvements

Architecture: Tepatche

- Derived from existing Perl scripts
- Divided into modules
- Performs similarly to Tepatche
  - Will patch from source
  - Can roll back patches after installation
  - Runs on a schedule
- Uses OpenBSD package facility for Binary Patching

Architecture: Tepatche

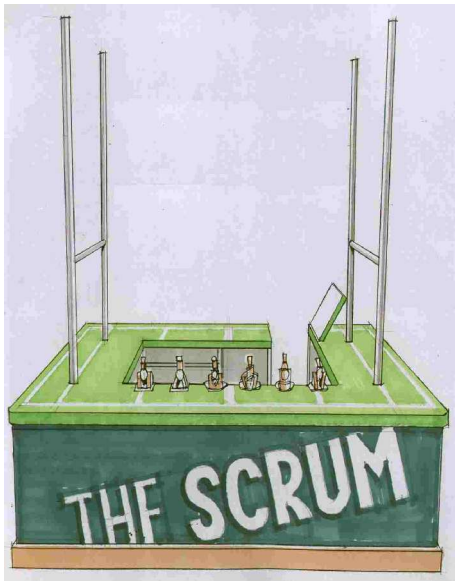


Functionality: Tepatche

- Functions the Patcher will perform:
  - Reads the configuration file.
  - Connects to the stated FTP server to download any new patches.
  - Applies security patches to the machine.

Design Paradigm

- Based on SCRUM
- Frequent meetings
- Scrum uses sprints
  - 30 day focus sessions
- Our experience
  - Sprints were shorter for our team, because of the short duration of our project
  - Sprints were slightly less effective
  - Frequent meetings were helpful



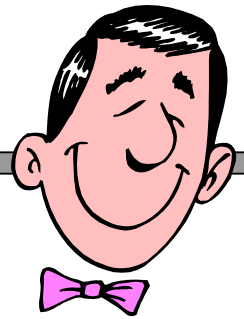
Project Timeline

- 2/18 Requirements Document Complete
- 3/05 Coding Begins
- 3/15 Design Document Complete
- 4/05 Product Mostly Working
- 4/10 Testing Begins
- ***4/23 Design Presentation***
- **4/25 Submit Product to Client for Testing**
- **5/3 Submit Final Product**

Project Difficulties

- Installer
  - TFTP (Trivial File Transfer Protocol)
  - Disk Partitioning
  - Limited tools
- Patcher
  - Learning PERL
  - Tepatche restructuring
  - Using package facility

Project Successes



- Major Functionality Complete
- Installer
  - Disk partitioning works
  - Automated installation works
- Tepatche
  - Bugs fixed
  - Binary capabilities exist

Project End Result

- Client is pleased with the functionality of the product.
  - Minor bugs need fixing.
  - Update to reflect changes in OpenBSD 3.5
- Documentation of functionality
  - Web FAQs.
  - UNIX style manual pages.

Project Exhibition & Demo

College of **E**ngineering & **T**echnology

Room 269

1:45 – 3:00

Questions