

Simple software  
Introduction

CSCI E 45a: The Cyber World – part A

Copyright © Scott Bradner & Ben Gaucherin 2015 1

---

---

---

---


---

---

---

---

Learning goals



- Understand the process and tools involved in making simple software
- Understand the environment within which software is run, and the key elements of running software
- Understand how one could abuse software to subvert it

Copyright © Scott Bradner & Ben Gaucherin 2015 2

---

---

---

---




---

---

---

---

Topics



- Running software - R  
Operating systems, and the basic structural elements of running software
- Making simple software - R  
Introduction to programming in a handful of slides
- The evolution of programming languages - R  
The landscape of programming languages

Copyright © Scott Bradner & Ben Gaucherin 2015 3

---

---

---

---

---

---

---

---

### Topics



- The craft of making software - R  
Going from Sunday afternoon coder to professional software engineer
- Unintended ways to use software - R  
What can you do if you use software in ways the people who made it did not expect



Copyright © Scott Bradner & Ben Gaucherin 2015

4

---

---

---

---

---

---

---

---

---

---

### Topics



- Artificial Intelligence - R  
A new twist on algorithms, data and programs exhibiting "apparent intelligence"

Copyright © Scott Bradner & Ben Gaucherin 2015

5

---

---

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

3 <http://blogs.screenconnect.com/image.axd?picture=linux-mac-windows.png>

3 "1983 CPA 5426 (1)" by Unknown - <http://www.muslimheritage.com/topics/default.cfm?ArticleID=631>, [1]. Licensed under Public Domain via Commons - [https://commons.wikimedia.org/wiki/File:1983\\_CPA\\_5426\\_\(1\).png#/media/File:1983\\_CPA\\_5426\\_\(1\).png](https://commons.wikimedia.org/wiki/File:1983_CPA_5426_(1).png#/media/File:1983_CPA_5426_(1).png)

4 "Swanson Shoe Repair 18" by Joe Mabel. Licensed under CC BY SA 3.0 via Commons - [https://commons.wikimedia.org/wiki/File:Swanson\\_Shoe\\_Repair\\_18.jpg#/media/File:Swanson\\_Shoe\\_Repair\\_18.jpg](https://commons.wikimedia.org/wiki/File:Swanson_Shoe_Repair_18.jpg#/media/File:Swanson_Shoe_Repair_18.jpg)

4 CIA logo

5 The official seal for the Algorithmic Warfare CrossFunctional Team - <https://imgix.bustle.com/inverse/73/9e/19/2d/a025/42ba/a81e/735e7ff30d6f/the-official-seal-for-the-algorithmic-warfare-cross-functional-team-aka-project-maven.png?w=710&h=752&fit=max&auto=format%2Ccompress&q=50&dp-r=2>

6

---

---

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

5 The official seal for the Algorithmic Warfare CrossFunctional

Team -

<https://imgix.bustle.com/inverse/73/9e/19/2d/a025/42ba/a81e/735e7ff30d6f/the-official-seal-for-the-algorithmic-warfare-cross-functional-team-aka-project-maven.png?w=710&h=752&fit=max&auto=format%2Ccompress&q=50&dpr=2>

Copyright © Scott Bradner & Ben Gaucherin 2015

7

---

---

---

---

---

---

---

---

Simple software  
Running software

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---


---

---

---

---

Operating systems



- Environment for running software
- User interface  
Command line, menus, Graphical User Interface (GUI)
- Hardware drivers  
Memory, storage, video, etc.
- Services
- Utilities

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

Historical highlights



- IBM's OS/360
- DEC's TOPS 10
- Xerox Alto OS
- DEC's VMS
- UNIX(es)  
System III, BSD, System V, Solaris
- UCLA's LOCUS
- Digital Research's CP/M
- Microsoft's DOS

3 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Historical highlights, contd.



- IBM's OS/2
- Original Apple Mac OS
- Microsoft's Windows
- More UNIX(es)
  - NeXT's NeXTSTEP
  - Linux
- Apple's Mac OS X and iOS
- Google's Android

4

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

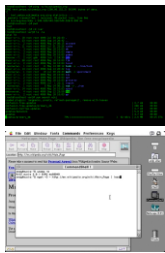
---

---

---

---

### Operating system user interface



- Command line
  - Still the most flexible
- Menu
- Touch/pen
- Graphical User Interfaces (GUIs)
- Multi-touch
- Gesture
- Brain controlled

5

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Operating system services

- **Service** - Software that runs in the background and performs utility functions
  - e.g. print spooler/manager
- Typically comes with tools to manage the service
- Often includes a programmatic interface to the service, an Application Programming Interface (API)

6

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Operating system services, contd.

- Example services:
  - Task management, scheduler
  - Memory management
  - Device management
  - File system
  - Print
  - Graphics/Video, Audio
  - Security
  - Networking
  - Etc.

7

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Operating system task management



- Manages processes
  - Chunks of code to be run
  - Loading/Unloading,
  - Starting/Stopping
- Single task vs. Multi-task
- Multi-task: Cooperative vs. Pre-emptive
  - Tragedy of the commons

8

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Process

- Top level chunk of self-contained, run-able software in “memory”
- Creating processes
  - Running an executable file
  - Operating systems primitives (e.g. Fork, Exec in UNIX)
- Operating systems provide facilities for processes to “talk” to each other

9

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Process, contd.

- Process hierarchy:  
Process ID (PID) and their Parent PID (PPID) - what started this process  
Starting with the "init task" of an operating system  
e.g. PID 1 on OS X is `launchd`
- Zombie/orphan processes

10

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

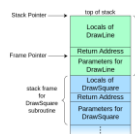
---

---

---

---

## Call stack and heap



- Call stack  
As functions are called, information about the function call is **pushed** on a stack  
When a function has completed, it returns control to the caller of the function and it gets **popped** off the stack
- Heap  
The memory space within which non-stack variables are allocated

11

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Threads



- Concurrency within a process
- Threads each have their own call stack
- Threads share the process heap

12

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The impact of resource glut



- Software developers used to have to worry about:
  - Limited resources: memory, storage, etc.
  - Time costly operations: disk writes, etc.
  - Code optimization
- But with resource glut, we don't need to be as careful...

13

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The impact of resource glut, contd.

- ...or do we?
  - Portable/mobile devices, other resource constrained devices (storage, processing, battery power, etc.)
  - Remember that technology continues to change
    - e.g., WAP is no longer needed



14

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

- Slide# credit
- 2 <http://blogs.scienceconnect.com/image.axd?picture=linux-mac-windows.png>
- 3 OpenVMS, MS DOS logos
- 3 AT&T System V <https://covers.openlibrary.org/b/id/6604025-M.jpg>
- 4 OS/2, Windows, NeXT, Android, Mac OS,
- 4 "Tux" by lewing@isc.tamu.edu and The GIMP. Licensed under Attribution via Wikimedia Commons - <https://commons.wikimedia.org/wiki/File:Tux.png#media/File:Tux.png>
- 5 "Apple Unk with Netscape" by Taken by me on a Quadra 650 running A/LUX 3.0.1. Via Wikipedia - [https://en.wikipedia.org/wiki/File:Apple\\_Unk\\_with\\_Netscape.jpg#media/File:Apple\\_Unk\\_with\\_Netscape.jpg](https://en.wikipedia.org/wiki/File:Apple_Unk_with_Netscape.jpg#media/File:Apple_Unk_with_Netscape.jpg)
- 5 "Linux command-line. Bash. GNOME Terminal screenshot" by ZxxZxxZ - Own work. Licensed under GPL via Commons - [https://commons.wikimedia.org/wiki/File:Linux\\_command-line\\_Bash\\_GNOME\\_Terminal\\_screenshot.png#media/File:Linux\\_command-line\\_Bash\\_GNOME\\_Terminal\\_screenshot.png](https://commons.wikimedia.org/wiki/File:Linux_command-line_Bash_GNOME_Terminal_screenshot.png#media/File:Linux_command-line_Bash_GNOME_Terminal_screenshot.png)

15

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---



### Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

8 Multitasking <https://flc.kr/p/7Mu767>

11 "Call stack layout" by R. S. Shaw (R. S. Shaw) - Own work.

Licensed under Public Domain via Commons - [https://commons.wikimedia.org/wiki/File:Call\\_stack\\_layout.svg#/media/File:Call\\_stack\\_layout.svg](https://commons.wikimedia.org/wiki/File:Call_stack_layout.svg#/media/File:Call_stack_layout.svg)

12 "Spool of white thread" by No machine readable author provided. Dmeranda assumed (based on copyright claims). - No machine readable source provided. Own work assumed (based on copyright claims).

Licensed under CC BY-SA 3.0 via Commons - [https://commons.wikimedia.org/wiki/File:Spool\\_of\\_white\\_thread.jpg#/media/File:Spool\\_of\\_white\\_thread.jpg](https://commons.wikimedia.org/wiki/File:Spool_of_white_thread.jpg#/media/File:Spool_of_white_thread.jpg)

---

---

---

---

---

---

---

---

Simple software  
Making simple software

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

"Problem solving is an art form not fully appreciated by some"

As proposed by the project sponsors

As specified in the project request

As designed by the senior analyst

As produced by the programmers

As installed at the user's site

What the user wanted

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

Simple software

- Self-contained application/utility
- Single executable file or script
- Single process

```
hisadmins-MacBook:~ jasper$ ./my_program
Hello World!
hisadmins-MacBook:~ jasper$
```

```
#include <stdio.h>
int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
```

3 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---



## Compilation



- Translating from one language into another
  - Specifically, takes **source code** and turns it into **object code** (or intermediate language)
  - Lexical, syntactic, and semantic analyses
- Rooted in linguistics work
  - Syntactic Structures
    - Noam Chomsky (1956)
  - Principles of compiler design
    - Aho/Ullman (1977)

7

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Libraries

- Packages of objects or functions to be used by developers
- Two primary types of libraries
  - Static libraries** – content of the library is included in the resulting executable
  - Dynamic libraries** – the executable is given a reference to an outside library that is loaded at runtime

8

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Linking

- **Linking** does two things:
  - “Glues” together individual chunks of object code
    - Some of the “chunks” can be static libraries or references to dynamic libraries
  - Sets up startup code for the operating system

9

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## The finished product

```
Jaspers-MacBook-Air:~ jasper$ ./my_program
Hello World!
Jaspers-MacBook-Air:~ jasper$
```



10

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

## Making software

The interpreted languages version

- Write **source code/script**
- An **interpreter** is used to interpret/run the script

Examples:

Shell script, Perl, PHP, Ruby, Python, JavaScript

“Slower” because of interpretation

Can be run in “interactive mode”

11

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

## An important scripting language - BASIC



- Beginner's All-purpose Symbolic Instruction Code **BASIC**

Making programming accessible to everyone

- 1964 Dartmouth College Kemeny/Kurtz

Enable students (not just in math or science curriculums) to use computers

- Microsoft's initial focus

12

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

### Runtimes, interpreters

- **Abstracting the OS**  
making the differences between OSes invisible to the running code
- **Portable code** – write once, run many
- **Runtime engine (or player)** runs an intermediate language  
e.g., JVM for Java, CLR for .NET, Flash player for Adobe Flash

13

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Core syntactical elements

- **Variables**  
scalar and structures  
`i=5, j={name: Bob, id:1234}`  
Variables can be typed: strings, numbers, date/time
- **Operators**  
General and type specific  
`str3 = strcat(str1, str2)`  
`now()`

14

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Core syntactical elements

- **Logic operators**  
&, |, ==, !=, <, >, etc.
- **Flow control**  
if...then, for loops, while loops, etc.
- **Error handling**  
try/throw/catch
- **Input/Output (I/O)**  
print, read, etc.

15

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Core syntactic elements

- **Functions/procedures/subroutines** –encapsulating functionality that can be called from other parts of the source code

Functions can return values  
Functions can take parameters

```
Int function  
add_two_numbers (int a,  
int b)
```

16

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide#	credit
2	Tree swing <a href="http://www.businessbills.com/images/treeswing/tree-swing-s-hogh.jpg">http://www.businessbills.com/images/treeswing/tree-swing-s-hogh.jpg</a>
3	Bill Amend FoxTrot <a href="http://mattturner.com/wordpress/wp-content/uploads/2011/04/pu-nit-ion-is.jpg">http://mattturner.com/wordpress/wp-content/uploads/2011/04/pu-nit-ion-is.jpg</a>
4	<a href="http://www.ollydbg.de/Pics/multilog.gif">http://www.ollydbg.de/Pics/multilog.gif</a>
7	"Green Dragon Book (front)" by Source (WP:NFC#4). Licensed under Fair use via Wikipedia - <a href="https://en.wikipedia.org/wiki/File:Green_Dragon_Book_(front).jpg#/media/File:Green_Dragon_Book_(front).jpg">https://en.wikipedia.org/wiki/File:Green_Dragon_Book_(front).jpg#/media/File:Green_Dragon_Book_(front).jpg</a>
12	"Altair Basic Sign" by Swtpc6800 en:User:Swtpc6800 Michael Holley - Swtpc6800 en:User:Swtpc6800 Michael Holley. Licensed under Public Domain via Commons - <a href="https://commons.wikimedia.org/wiki/File:Altair_Basic_Sign.jpg#/media/File:Altair_Basic_Sign.jpg">https://commons.wikimedia.org/wiki/File:Altair_Basic_Sign.jpg#/media/File:Altair_Basic_Sign.jpg</a>

17

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

Simple software  
The evolution of programming languages

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---


---

---

---

---

Functional programming languages



- Focused on describing the flow of processing
- Allows the implementation of Algorithms

al-Khwārizmī c. 780 to c. 850

Recipe for solving a given problem

Helps assess the complexity of a process - time to process the data in relation to the size of data being processed

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

Example—Simple C program

```
# include <stdio.h>

int main(void)
{
    int count;
    for (count=1; count<=500; count++)
        printf("I will not throw paper
        airplanes in class.\n");
    return 0;
}
```

3 Copyright © Scott Bradner & Ben Gaucherin 2014

---

---

---

---

---

---

---

---



## Object Oriented Design/Languages



Alan Kay

- Bringing programming languages closer to “real-world” things

Alan Kay’s early history of Smalltalk



Marvin Minsky

Marvin Minsky’s Frames  
Artificial Intelligence and knowledge representation

4

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Object Oriented Design/Languages, contd.

- Classes are models for objects
- Objects are **instances** of classes
- Two parts to an object:  
**Attributes** - data  
**Methods** – code for object behavior

5

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Example—Java

```
class Person
{
    String name;
    Date date_of_birth;
    void hello(){
        System.out.println("Hi! My name is " + name +
            "\n");
    }
    int age() {
        System.out.println("My age is " + to_string(now() -
            date_of_birth) "\n");
    }
}

Person John = new Person();
```

6

Copyright © Scott Bradner & Ben Gaucherin 2014

---

---

---

---

---

---

---

---

## Object Orientation—the good stuff

- **Encapsulation**  
You can only see the interface of an object  
Its “innards” are not visible  
e.g., in the `Person` class the calculation to get someone’s age is embedded in the object – anyone using this object does not know how the calculation is performed, or whether the age is stored or calculated

7

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Object Orientation—the good stuff, contd.

- **Inheritance**  
Allows a class to take on the shared characteristics of another class (parent class), and to modify/add to them  
e.g., if we had a base `Shape` class that captures basic behaviors of geometric figures; we could have the following classes inherit from this parent class: `Square`, `Circle`, `Triangle`

8

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Object Orientation—the good stuff, contd.

- **Polymorphism**  
The ability for code to act on objects of multiple types  
e.g., if our `Shape` class had a `color` attribute and a method `set_color(a_color)` to change the color of a shape - we could call `set_color` on `Squares`, `Circles`, and `Triangles`

9

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Object design patterns



- General, reusable solutions to common problems  
E.g.: façade, delegation, factory, singleton, etc.
- Model View Controller
  - Model** – the core domain/business logic
  - View** – the visualization of the Model
  - Controller** – routes requests to View/Model

10

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Visual/Event based programming

- Build a graphical user interface
- Associate actions to events  
e.g. `button1_click() { <do something> }`
- Event loop runs constantly to handle events as they come up  
Events queue up in an “event queue”

11

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## HyperCard/HyperTalk



- Started by Bill Atkinson in 1985
- HyperTalk added by Dan Winkler in 1986
- Released by Apple as free software on Mac OS in 1987
- Programming for the people  
Easy visual interface design  
English-like scripting

12

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Microsoft Visual Basic



- Initially created by Alan Cooper and his team at Tripod
- First released at COMDEX May 1991
- Visual programming in Windows (and DOS)
- First enterprise grade visual programming language

13

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Scripting the Web

- **Server side scripting:**  
Perl, PHP, Python, Ruby, JavaScript
- **Server side frameworks\*:**  
Ruby on Rails (RoR), Django, lots of PHP MVC frameworks
- **Client side scripting:**  
JavaScript
- **Client side frameworks\*:**  
jQuery, Dojo, etc.

\* **Framework:** collection of code, tools, designs to serve as foundation for building more complex structures

14

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

## Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit  
2 "1983 CPA 5426 (1)" by Unknown - <http://www.muslimheritage.com/to pics/default.cfm?ArticleID=631>, [1]. Licensed under Public Domain via Commons - [https://commons.wikimedia.org/wiki/File:1983\\_CPA\\_5426\\_\(1\).png#/media/File:1983\\_CPA\\_5426\\_\(1\).png](https://commons.wikimedia.org/wiki/File:1983_CPA_5426_(1).png#/media/File:1983_CPA_5426_(1).png)  
4 "Alan Kay (3097597186)" by Marcin Wichary from San Francisco, U.S.A. - Alan Kay. Licensed under CC BY 2.0 via Commons - [https://commons.wikimedia.org/wiki/File:Alan\\_Kay\\_\(3097597186\).jpg#/media/File:Alan\\_Kay\\_\(3097597186\).jpg](https://commons.wikimedia.org/wiki/File:Alan_Kay_(3097597186).jpg#/media/File:Alan_Kay_(3097597186).jpg)  
4 "Marvin Minsky at OLPCb" by Original uploader was Sethwoodworth at en.wikipedia, taken by BcJordan - Transferred from en.wikipedia; transferred to Commons by User:Mardetanha using CommonsHelper. Licensed under CC BY 3.0 via Commons - [https://commons.wikimedia.org/wiki/File:Marvin\\_Minsky\\_at\\_OLPCb.jpg#/media/File:Marvin\\_Minsky\\_at\\_OLPCb.jpg](https://commons.wikimedia.org/wiki/File:Marvin_Minsky_at_OLPCb.jpg#/media/File:Marvin_Minsky_at_OLPCb.jpg)  
10 Gang of four [http://aph.is.quoracdn.net/main-qimg-04ce4370594c6870fb7d26681676dd357cconvert\\_to\\_web-p=tru](http://aph.is.quoracdn.net/main-qimg-04ce4370594c6870fb7d26681676dd357cconvert_to_web-p=tru)e

15

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted  
Slide# credit  
10 "Design Patterns cover" by Source. Licensed under Fair use via  
Wikipedia  
[https://en.wikipedia.org/wiki/File:Design\\_Patterns\\_cover.jpg#/media/File:Design\\_Patterns\\_cover.jpg](https://en.wikipedia.org/wiki/File:Design_Patterns_cover.jpg#/media/File:Design_Patterns_cover.jpg)  
12 Hypercard icon  
13 VisualBasic logo

---

---

---

---

---

---

---

---

Simple software  
The craft of making software

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---


---

---

---

---

The craft of making software



- Also referred to as Software Engineering
- What goes into making good software besides coding
- Not as commonly understood or adhered to as one would think (or hope)

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

The craft – Analytical and system thinking

- A way of systematic analysis that asks, “How can I break this problem down into its constituent parts?”
- And conversely, the ability to understand how individual parts come together into a coherent whole

3 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

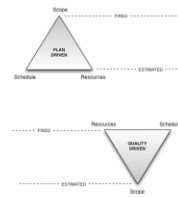
---

---

---

---

### The craft – Software Development Life Cycle



SDLC - Overall process for developing software

Old school – “waterfall”

Fully design, then fully develop, then fully test, then deploy

New school – Agile

Inspired from lean principles  
Develop deployable software in small (e.g. 2 weeks) iterations

---

---

---

---

---

---

---

---

### The craft –SDLC, contd.

- Why waterfall?  
Fully understand the scope and every detail before you start coding  
Thus, think you understand the level of effort, timeframe, and budget needed to deliver the solution  
Testing is a focused quality effort to finish up the project

---

---

---

---

---

---

---

---

### The craft –SDLC, contd.

- Problems with waterfall  
The business’ needs (or understanding of its needs) may have changed by the time the design or development is done  
A large percentage of the functionality you design may be theoretically, but not actually, useful  
Long wait time before you get anything useful

---

---

---

---

---

---

---

---

### The craft – SDLC, contd.

- Problems with waterfall
  - Long phases means the magnitude (and odds) of slippage are high
  - By making “testing” a separate phase it allows for quality and design issues to go undetected/un-addressed for long periods of time, and be very disruptive when found
  - And many more...

7

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft – SDLC, contd.

- Why Lean/Agile?
  - Reduce “waste”
  - Deliver value quickly and frequently
  - “Fail fast and often”
- Not a perfect model either, but a marked improvement over waterfall

8

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft - Architecture

- For simple and complex software
- Defining and implementing the optimal design for the software needed
  - Right mix of technologies
  - Logical structure
  - Physical structure
  - Connection points and dependencies with the rest of the environment

9

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---



### The craft – Source code control



- Keep track of the changes made to the source code  
Allow multiple developers to work together on the same code base
- And why they were made (so you can revert back, or audit the code)

10

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft - Versioning

- Important to compare to similar pieces of software and know which one is newer than the other
- Most versioning system use:  
<major>.<minor>.<more>  
e.g. 01.23.2014101701
- Minor releases are also referred to as **point releases**

11

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft – Coding standards

- Making the code legible/maintainable for others (and for you)  
Naming rules, indentation rules, source code file structure  
Source file headers, comments, etc.  
Embedded documentation
- Helped/enforced by code reviews and compliance tools

12

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft – Quality Assurance

- Knowing what it means to meet or exceed expectations  
Test Driven Development
- Many sub-categories of testing to ensure quality  
Unit testing  
Integration testing  
Performance and scalability testing  
Regression testing

13

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft – Build & Release



**Jenkins**



- **Build** - Process for producing the finished product
- **Release** - And moving it through different environments  
e.g., Dev, Test, Stage, Production
- Moving increasingly to automated, continuous build and release

14

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### The craft – DevOps

- Looking past the “silicon snake oil”  
Making the bridge between development and operations better, and more efficient  
Using Agile, automation and programmable infrastructures  
SDN, Cloud IaaS, etc.
- Many new tools emerging  
Puppet, Ansible, Chef, etc.

15

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---



Simple software  
Unintended ways to use software

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

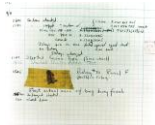
---

---

---

---

Compromising software



- Find bugs in the software  
Bugs with “useful” negative side-effects – **Zero Days**  
e.g., the Shellshock Bash bug
- Fuzzing**: testing by providing random, un-expected, invalid input  
*“The (security) risk in using a programming language is directly proportional to the expressiveness of the language”*  
Dan Geer: HKS IGA 236M talk 01/13

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---



---

---

---

---

Compromising software, contd.



- Patching executables  
Inserting nefarious code  
Bypassing protection mechanisms
- Using binders/joiners to “bundle” good and bad executables
- Patching dynamic libraries to change behavior  
**Rootkits** - modify operating systems libraries and executables

3 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Compromising software, contd.



Ken Thompson

- Compromising the tools  
e.g., compromised versions of compilers can surreptitiously introduce un-expected code



4

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

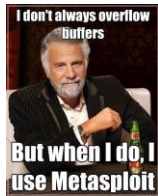
---

---

---

---

### Buffer overflow



- **Buffer\* overflow** –occurs when more data than a buffer can hold is written to the buffer  
The excess data overwrites other data in memory, leading to un-expected (or expected) results  
This has been a popular means of inserting/injecting nefarious code into a running process

\* **Buffer**: a small area of memory used for temporary storage of data. Usually used in reading/writing to/from keyboard, disk, network, etc.

5

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Disassembling/Reversing



- **Reversing**: contraction of Reverse Engineering  
**Crackme**: software built to practice reversing skills
- **Disassembling**: translate object code into assembly code
- Prohibited by many End User License Agreement (EULA)

6

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Preventing /detecting compromise

- Keep cryptographic hash of files  
Files - executables, libraries, scripts, etc.

```
Jaspers-MacBook-Air:~ jasper$
Jaspers-MacBook-Air:~ jasper$ md5 my_program
MD5 (my_program) = 3399cd45a6b966666df9ab3e385e480b
Jaspers-MacBook-Air:~ jasper$
```

- Code signing  
**Software publisher** – Creates digital signature of the executable  
**Software user** – Checks that the digital signatures is valid

7

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

### Beyond tech – ethics and software making



Timmi Gebu

- There's more to software making than the technical aspect of it
- What would you do if you were asked to build technology that...  
Serves a purpose that goes against your personal beliefs  
Could be used (today or in the future) as a tool for oppression and control
- Engineers are starting to revolt

8

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

- | Slide# | credit  |
|--------|---|
| 2      | "H96566k" by Courtesy of the Naval Surface Warfare Center, Dahlgren, VA., 1988. - U.S. Naval Historical Center Online Library Photograph NH 96566-KN. Licensed under Public Domain via Wikimedia Commons - <a href="https://commons.wikimedia.org/wiki/File:H96566k.jpg#/media/File:H96566k.jpg">https://commons.wikimedia.org/wiki/File:H96566k.jpg#/media/File:H96566k.jpg</a>  |
| 3      | "Cheval de Troie d'après le Virgile du Vatican" by after the Vergilius Vaticanus - Internet Archive. Licensed under Public Domain via Wikimedia Commons - <a href="https://commons.wikimedia.org/wiki/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg#/media/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg">https://commons.wikimedia.org/wiki/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg#/media/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg</a> |
| 3      | SONY logo   |
| 4      | "Ken n dennis" by Unknown - <a href="http://www.catb.org/~esr/jargon/html/U/Unix.html">http://www.catb.org/~esr/jargon/html/U/Unix.html</a> . Licensed under Public Domain via Wikimedia Commons - <a href="https://commons.wikimedia.org/wiki/File:Ken_n_dennis.jpg#/media/File:Ken_n_dennis.jpg">https://commons.wikimedia.org/wiki/File:Ken_n_dennis.jpg#/media/File:Ken_n_dennis.jpg</a>  |
| 4      | CIA and Apple Xcode logos   |
| 5      | "I don't always overflow buffers" <a href="http://www.quickmeme.com/meme/55qb">http://www.quickmeme.com/meme/55qb</a>   |

9

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

6 crackmes.de/logo

7 Screenshot by Ben Gaucherin

8 The official seal for the Algorithmic Warfare CrossFunctional Team -  
<https://imgix.bustle.com/inverse/73/9e/19/2d/a025/42ba/a81e/735e7f30d6f/the-official-seal-for-the-algorithmic-warfare-cross-functional-team-aka-project-maven.png?w=710&h=752&fit=max&auto=format%2Ccompress&q=50&dpr=2>

8 Timnit Gebru -

[https://upload.wikimedia.org/wikipedia/commons/thumb/6/6d/Timnit\\_Gebru\\_crop.jpg/440px-Timnit\\_Gebru\\_crop.jpg](https://upload.wikimedia.org/wikipedia/commons/thumb/6/6d/Timnit_Gebru_crop.jpg/440px-Timnit_Gebru_crop.jpg)

---

---

---

---

---

---

---

---

Simple software  
Conclusion

CSCI E 45a: The Cyber World – part A

1 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

Some key points

- Software making has evolved greatly:
  - To be more accessible to the masses
  - To promote reuse
  - To support a broader set of technologies, with a smaller set of tools
  - To involve more sophisticated practices to yield better value and quality

2 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---


---

---

---

---

Some key points, contd.



- Security is a problem
  - There are limited ways for software to be made secure (in the long run)
  - Many ways to abuse it
  - There is no “provable” way to eliminate all bugs
- No one knows if and how much you can trust the “code bits” you run

3 Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---



### Some key points, contd.



- Everyone should learn to code
- The 3 R's and the C: reading, writing, arithmetic, and coding

4

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---

### Image credits

All drawings and photos by Ben Gaucherin unless noted

Slide# credit

3 "Cheval de Troie d'après le Virgile du Vatican" by after the Vergilius Vaticanus - Internet Archive. Licensed under Public Domain via Wikimedia Commons - [https://commons.wikimedia.org/wiki/File:Cheval\\_de\\_Troie\\_d%27apr%C3%A8s\\_le\\_Virgile\\_du\\_Vatican.jpg#/media/File:Cheval\\_de\\_Troie\\_d%27apr%C3%A8s\\_le\\_Virgile\\_du\\_Vatican.jpg](https://commons.wikimedia.org/wiki/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg#/media/File:Cheval_de_Troie_d%27apr%C3%A8s_le_Virgile_du_Vatican.jpg)

4 CS50 Fair fair.cs50.net

5

Copyright © Scott Bradner & Ben Gaucherin 2015

---

---

---

---

---

---

---

---