C++ for Real-Time Safety-Critical Linux Systems

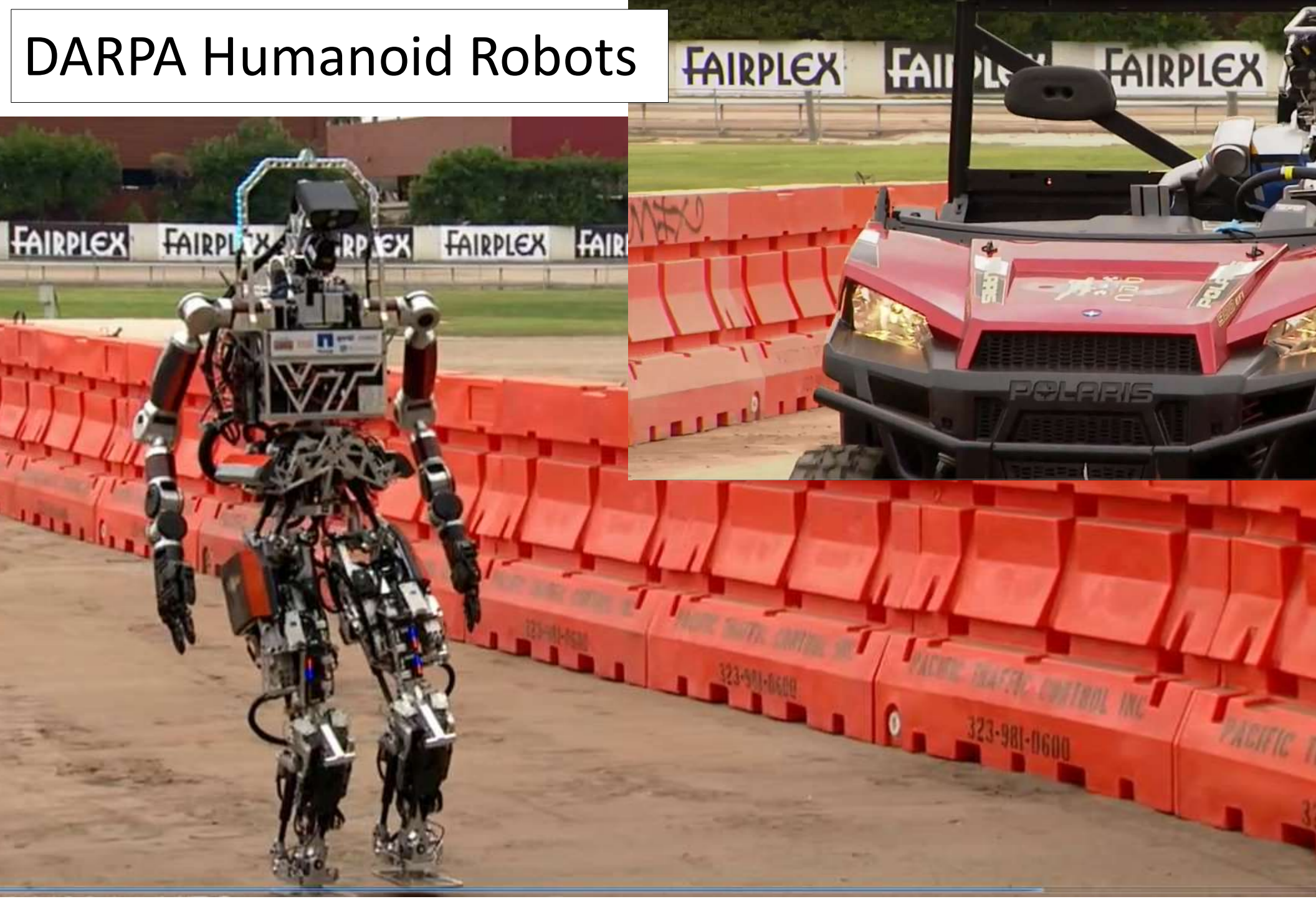
Robin Rowe & Gabrielle Pantera

Open Source Summit +
Embedded Linux Conference Europe 2020
Tuesday, October 27th, 2020

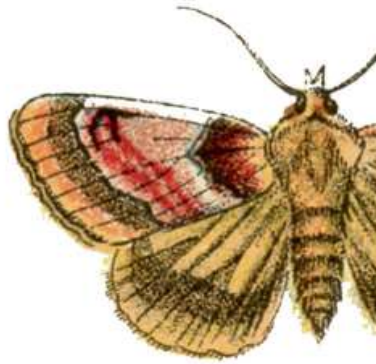
T
A


AUTOPILOT
65
65
mph
65
Daft Punk Radio
Aerodynamic
Discovery
Daft Punk
1:33
-2:23
270 mi
Avg
320
Wh
mi
P R N D

DARPA Humanoid Robots



Bugs and Uptime



- Fewer lines of code => fewer bugs
- Bug clusters, bugs tend to group
- Technical debt
- Timing errors
- At Google, 70% of failures happen when releasing a new version of code
- To reach nine 9s we must bank reliability
- Nine 9s is…

  1/10 the time of the blink of an eye

Availability %	Nines
90%	1
99%	2
99.9%	3
99.99%	4
99.999%	5
99.9999%	6
99.99999%	7
99.999999%	8
99.9999999%	9

Safety Standards

- ISO 9001 QA Process
- ISO IEC 23360 Linux LSB
- ISO 13485 Medical Software
- DO-178 Aviation Software
- ISO 26262 Automotive Software
- DOT ITS ATC Automotive Traffic Light Softwa
- MISRA C
- *Future: ISO 56007 Innovation Idea Manage*

Process Types

- Agile
- Waterfall
- Unstructured

What process do we have?

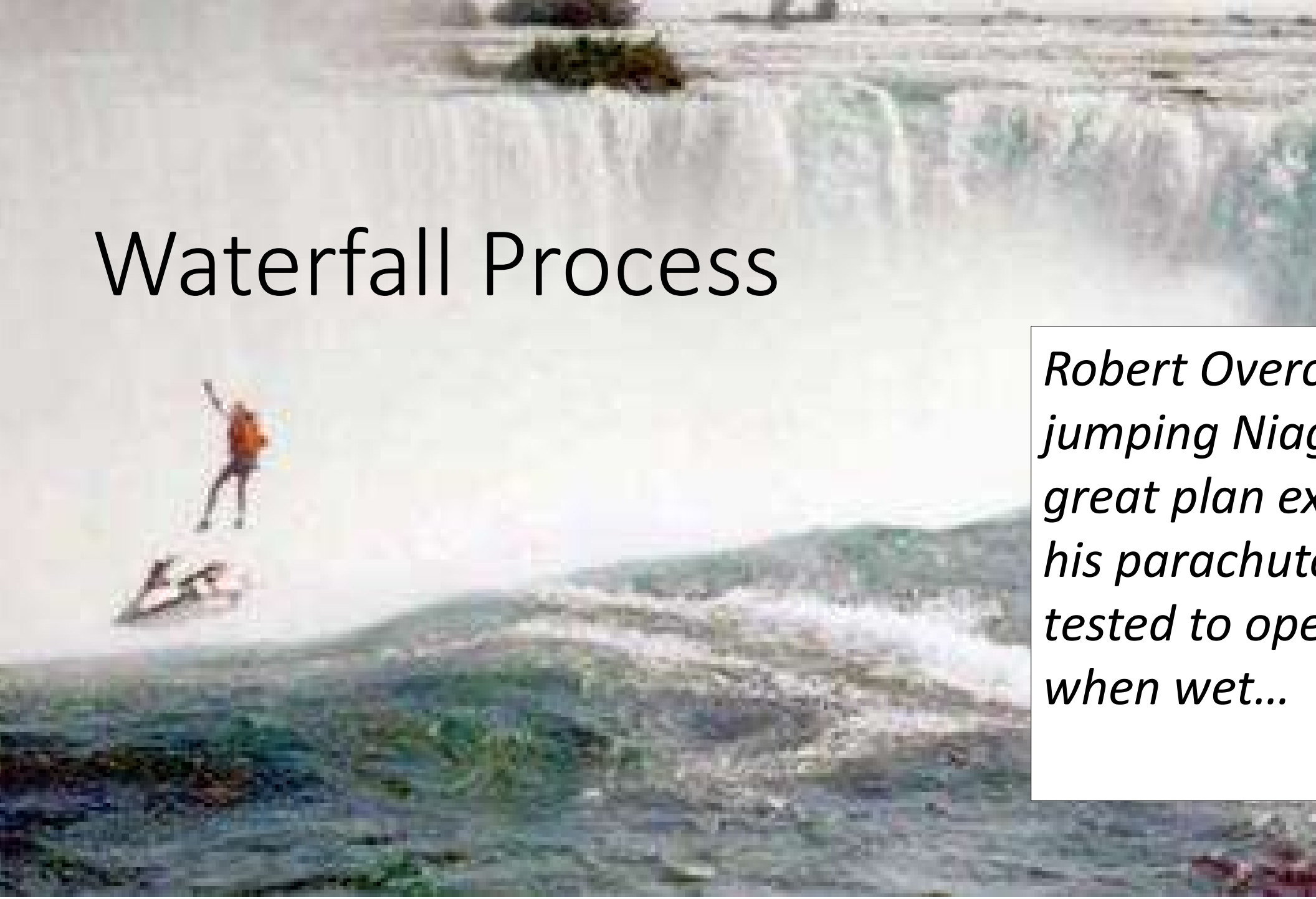
Unstructured Process Indicators

- No specific goals
- Top-down directives out of sync with conditions on the ground
- Deadlines and milestones seem incomprehensible to team
- No lessons learned, keep trying harder with the same plan
- Death marches, deadlines slide as the plan remains unchanged
- Personal baggage, team stressed out, mentally checked out, o
- Expectations of project failure voiced at meetings
- Managers consumed with putting out fires and reproaching te
- Team doesn’t know what the managers are doing
- Budget out of control, binge spending, illogical cost-cutting

Waterfall Process Indicators

- Top-down, business requirements provided by leader
- Requirements analysis and written specifications
- Preliminary Design Review, Critical Design Review
- Charge numbers, Bug tracking
- Microsoft Project, Gantt charts
- Daily team meetings discuss what happened yesterday
- Managers spend much of their time absent for planning meeti
- Rigid plans that demand sticking to the plan no matter what
- Big bang finished deliverable, deadlines tend to slip

Waterfall Process



Robert Overc
jumping Niag
great plan ex
his parachute
tested to ope
when wet…

Agile condensed, by agilesista.com

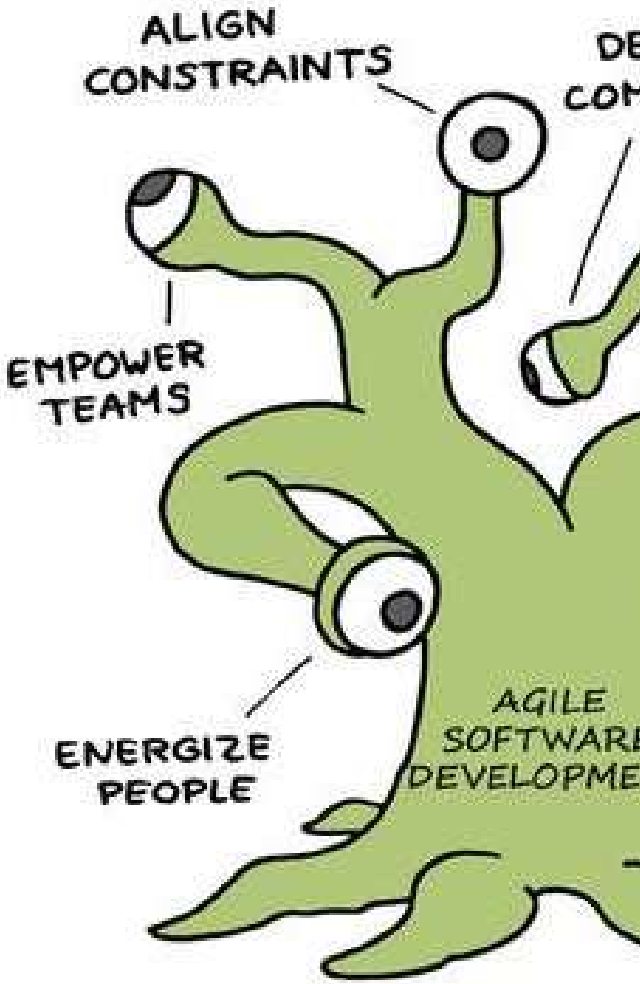


Image: Jurgen Appelo, Flick

Agile Process Indicators

- User stories
- Sprints and retrospectives
- Release early and often
- Pair programming
- Kanban boards
- Meetings are forward-looking or retrospectives
- Cloud-based project management
  - JIRA
  - Git

Why Do We Like C++?

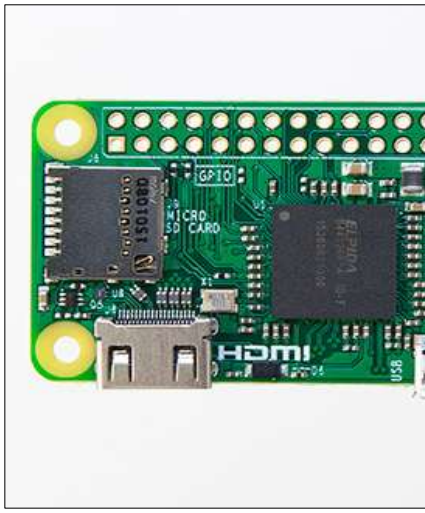
- Performance: 10x faster typical
- 20 million C++ programmers
- Object-oriented design safety
- Reliability: extensive set of tools for debug/test
- C++ leads motion picture visual effects, VR and autonom
- C++ everywhere: Windows, MacOS, Linux, iOS, Android, systems, IoT, cloud, aerospace, AI, databases

Object-Oriented Design

"The most important single aspect of software development is to be clear about what you are trying to build." -- Bjarne Stroustrup

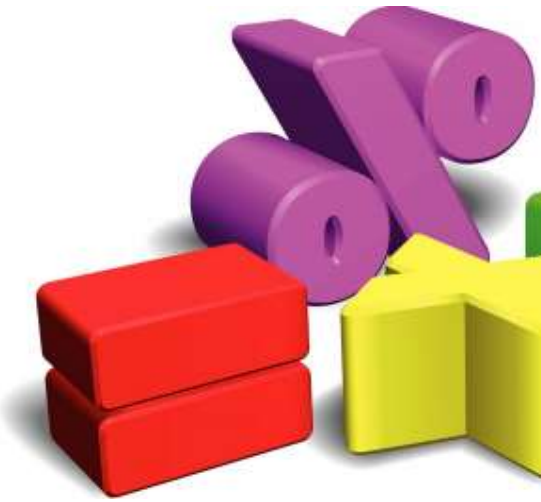
- Software that snaps together like Legos
- Nouns are classes, verbs are functions
- Encapsulation hides data from code that shouldn't change it
- C++ is as easy as PIE:
  - Polymorphism
  - Inheritance
  - Encapsulation
- Elegant design simplicity is what's left after removing complex

Embedded Systems Design



- Code design: think small, think fast
- Avoid the heap after main()
- Avoid termination, and therefore exceptions
- If rebooting is feasible, use a Highlander for auto-restart
- Avoid implicit initialization of static objects before main(),
- Bring-up: initialize explicitly in main()
- Avoid senseless optimizations, profile and test
- Avoid risky coding practices
- Use type-safety, encapsulation, be const-correct

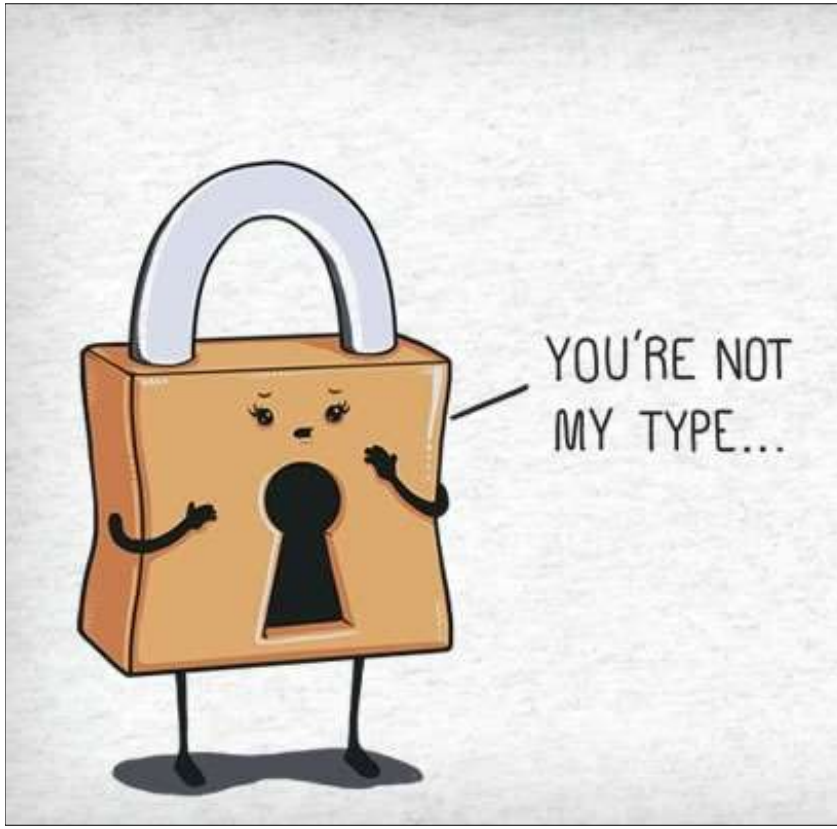
Safety-Critical C++ Concepts



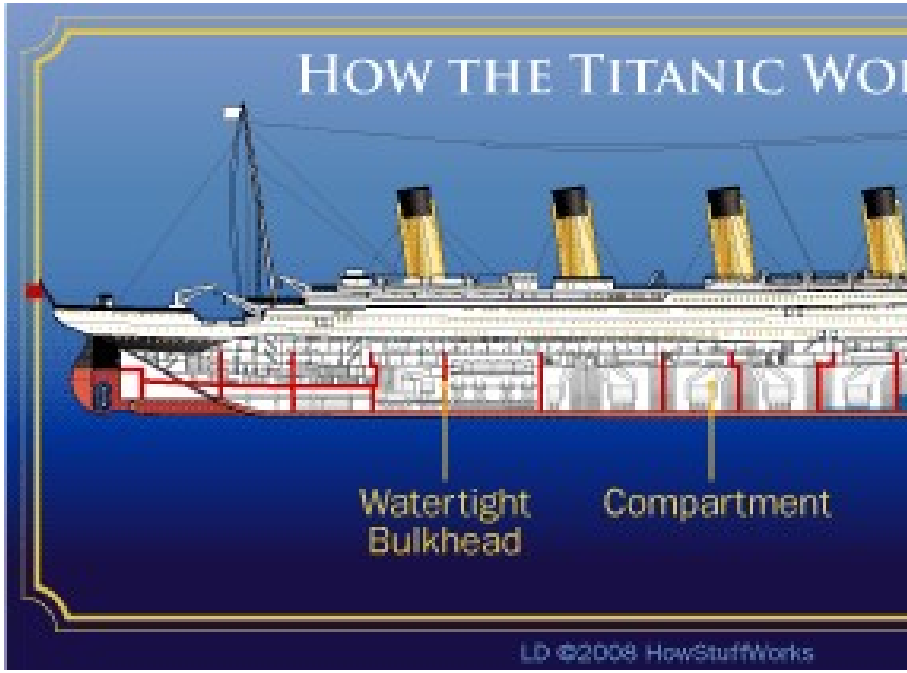
- Encapsulation
- Memory Management
- Thread Management
- Hard and Soft Real-time
- Static Analysis
- Single codebase on Linux, Windows, Mac, embedded
- Audit, Simulation, Playback
- SQA, Unit and Regression

C++ Type-Safety

- Locks out incompatible code
- Typical type errors will be found at compile time
- A major way of static checking
- We can still cast, when we must
- sizeof(ptr) unknowable
- Use intptr_t type



C++ Encapsulation



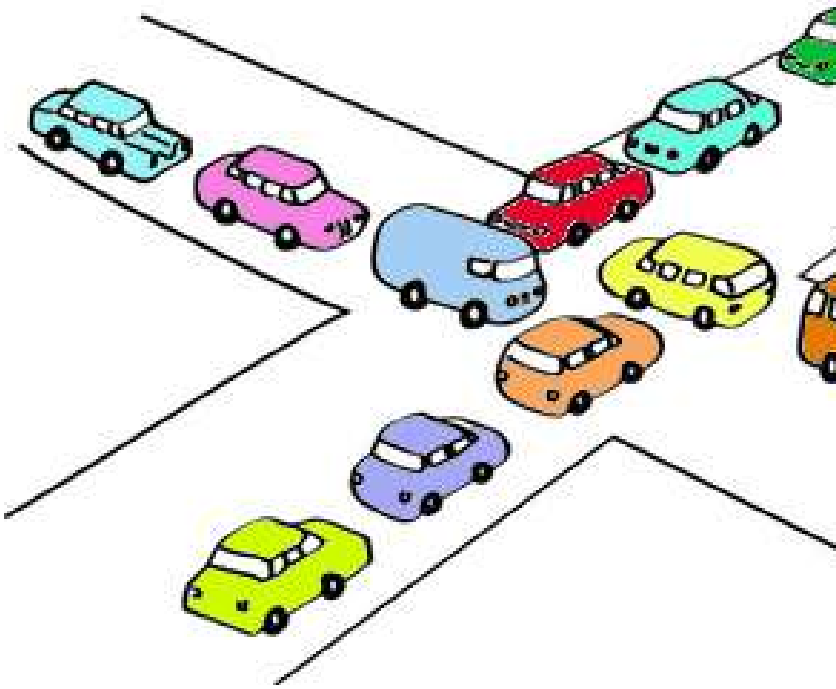
- Watertight compartments
- Classes and objects
  - **`private`**
  - **`protected`**
  - **`public`**
- Encapsulation is a form of data hiding
- Encapsulation can ensure consistency of state
- Don’t use inheritance where you mean encapsulation
- **`const`** is also a form of encapsulation

C++ Memory Management

- Stack: Temporary, Fixed Size
  - Failure consequence is stack overflow, a crash
- Static: Forever, fixed size
  - Failure consequence is program too big to load
- Heap: Can vary in time and space
  - Failure consequence is null pointer or exception
  - Fragmentation possible
  - Memory leak possible
  - **`unique_pointer<>`** leaves no garbage to collect
  - **`shared_pointer<>`** useful for removing leaks in legacy code

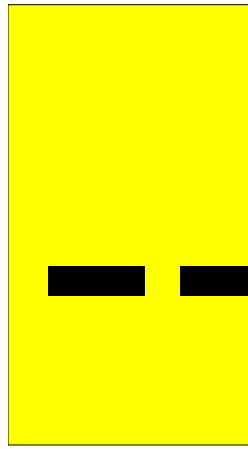
Don't tell anyone.
I actually forget lo

C++ Concurrency



- Message queues
- Threads, **`join()`** or **`detach()`**
- Mutex
- Locks
- Condition Variable
- Lockfree
- Double-buffering
- **`volatile`** and **`<atomic>`**

C++ Pointers



- Avoid garbage by using **`unique_pointer<>`**
- Avoid wild pointers by pointing to nullptr in constructor
- Avoid dangling pointers by nulling after release
- Hide pointers by making them private
- Where a pointer could never be null, use references instead
- Avoid unintended modifications by using **`const`**
- Trace in a debugger all code you write that does pointer math

C++ Casting

- We don’t want any, but sometimes…
- Indispensable for coping with legacy design issues
- C-style casts: **`(int) x`**
- C++ constructor casts: **`int(x)`**
- **`const_cast<>`** to cast away const
- **`static_cast<>`** is like a C cast
- **`reinterpret_cast<>`** made for **`void*`** casts
- **`dynamic_cast<>`** returns 0 if fails, for up-casts
- Use function templates to block integer casts

C++ Exceptions, Don't

- **`signal`** in a type-safe way
- **`return false`** is 10x faster and easier to trac
- Use to add simplistic error handling to legacy co
- C++ exceptions are termination-based, if a secon throw happens before the first throw is caught,
- If we don't like **`new`** because it can be slow and terminate unexpectedly, we don't like **`throw`** fo same reasons

Testing Methods

- Tracing
- Unit
- Stress
- Regression
- Monkey
- Screen scraping
- Keyboard/mouse macros
- Catch library

C++ Traps

- Infinite loop
- Recursion
- Casts
- Wild pointers
- Segfault, division by zero, FPE, fatal cache miss
- Initialization before main()
- Complexity and obfuscation
- Cohesion vs. spaghetti code

Agile Safety-Critical Mindset

“How could you not select a guy who wears a woman's hair band for sunglasses?”

“But, seriously, Geordi saved the Enterprise from certain doom in countless episodes. Sure, so did Scotty. But Geordi did not whine about it like Scotty, ‘Captain, I'm giving her all she's got… She can't take much more.’ Nope. Geordi just got 'er done. As ridiculous as that visor Geordi sported looked, it enabled him to see things that other crew members could not.”

LaVar Burton

LaVar Burto

A charmingl
session in th

Real Life C++ Examples

- Literally *Everything* Depends on C++
- Real-Time Systems
- Safety-Critical Systems
- Embedded Systems
- Financial Systems
- Critical Infrastructure
- Let's Look at Some Systems I've Touched…

Mt Huashan

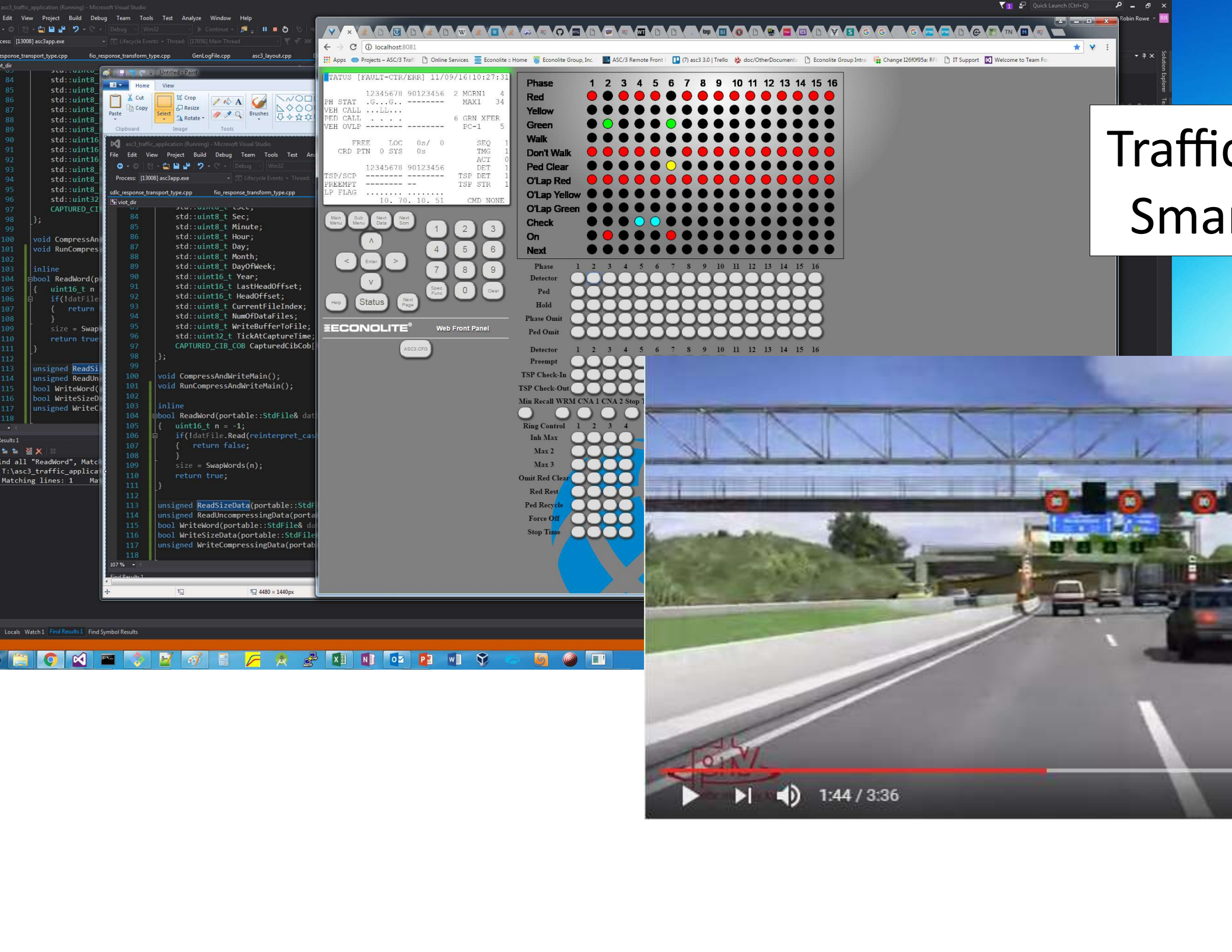

rip curl
Go
Be a HERO.


Barbie Vlog # 19 | Cosplay Costumes | Barbie
of Cosplay online.
Here are a couple of my first
0:05 / 1:25
CC
HD
YouTube


CALL OF DUTY


PUSH
POWER
DIRECTV
GUIDE
MENU
REC
SELECT
INFO
DIRECTV HD DVR

Casino Gaming


Traffic
Smar
1:44 / 3:36

USS Lin
Global
and Co
System

Thank you!

Robin Rowe

Gabrielle Pantera

Robin.Rowe@VentureHollyw