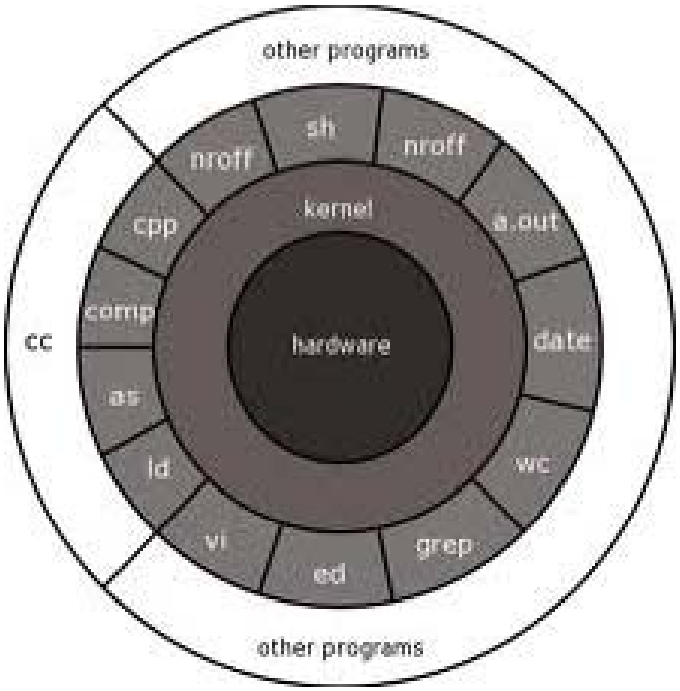
CPSC 875

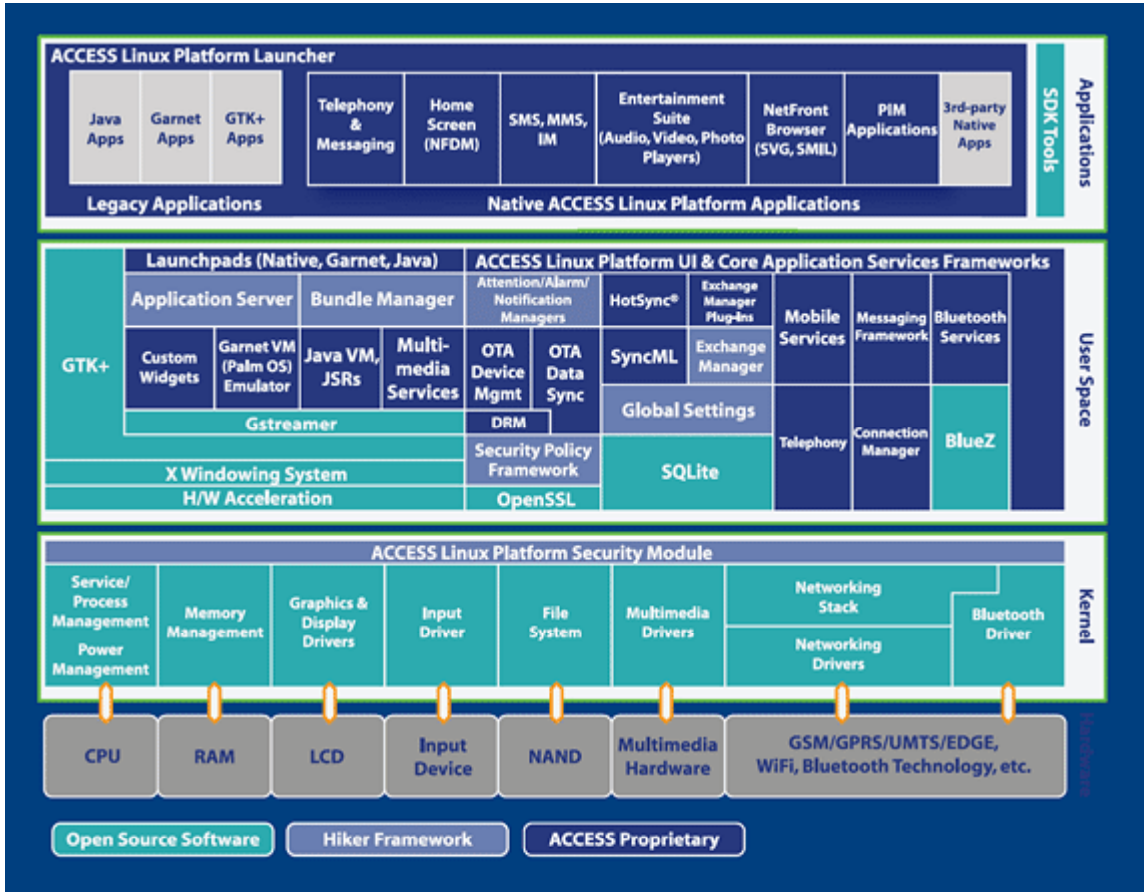
John D. McGregor

Class 6 – Design Concept

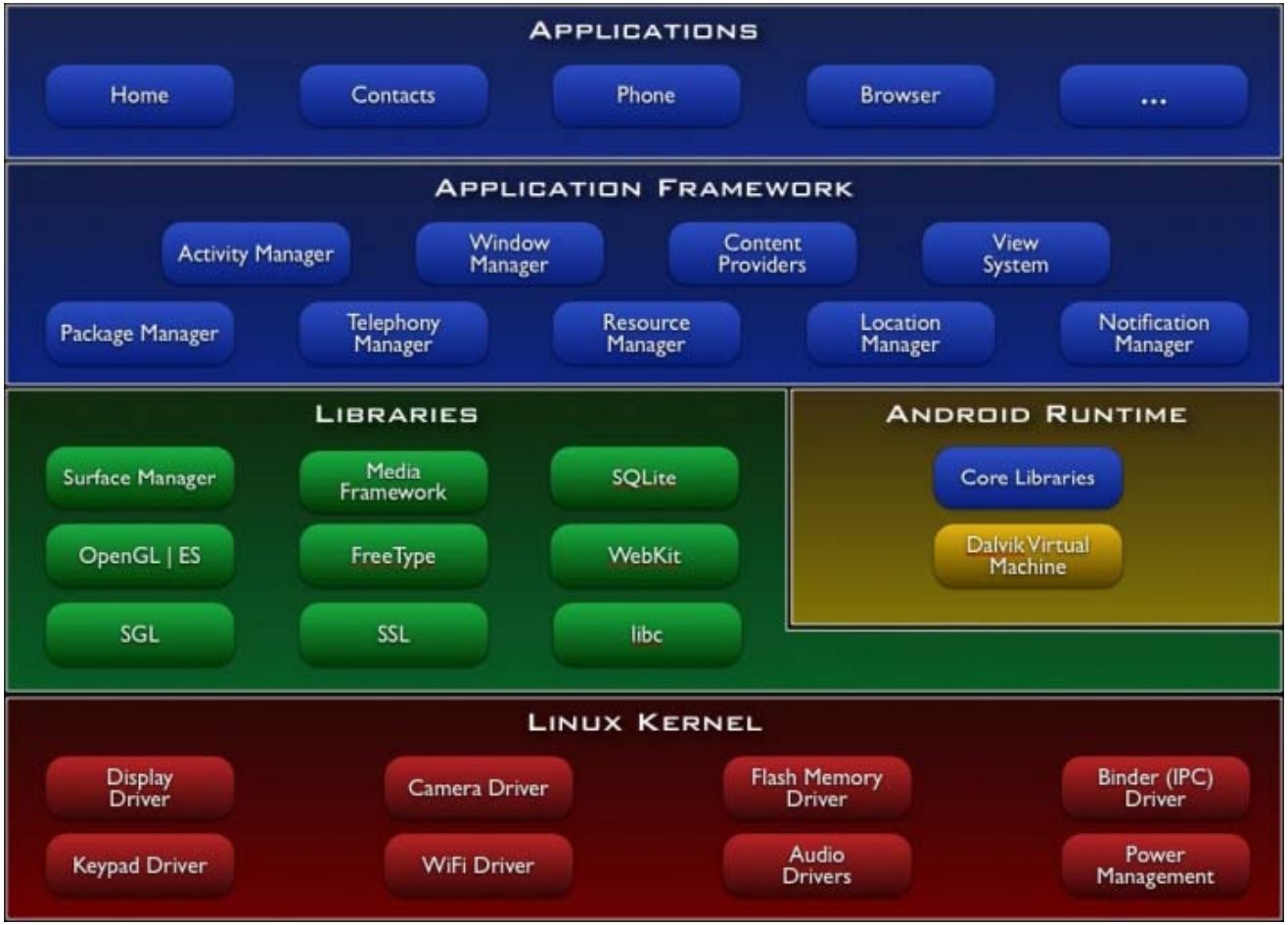
Unix


other programs
sh
nroff
nroff
kernel
cpp
a.out
comp
hardware
cc
date
as
ld
wc
vi
ed
grep
other programs

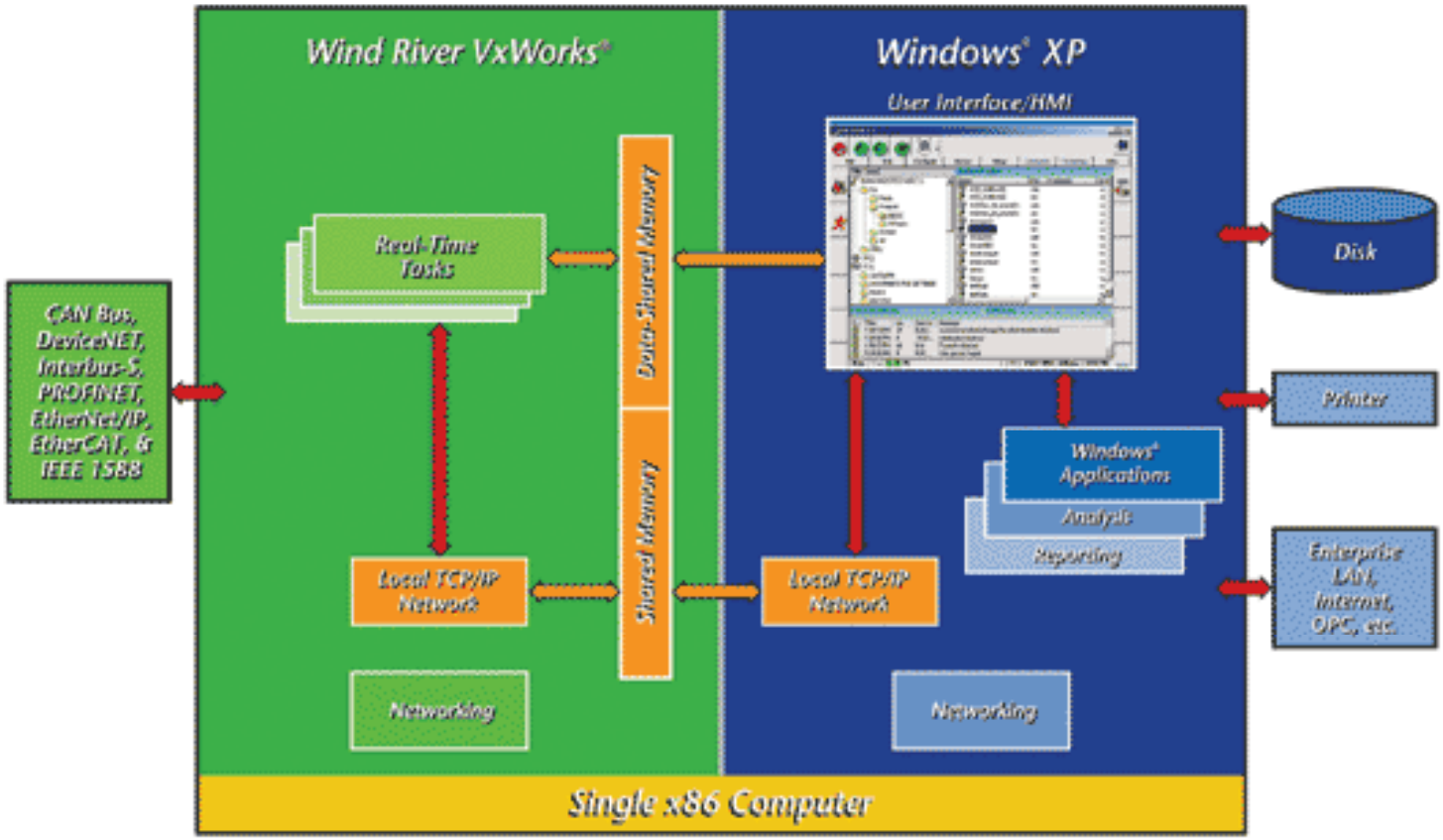
Linux


ACCESS Linux Platform Launcher
Java Apps
Garnet Apps
GTK+ Apps
Telephony & Messaging
Home Screen (NFDM)
SMS, MMS, IM
Entertainment Suite (Audio, Video, Photo Players)
NetFront Browser (SVG, SMIL)
PIM Applications
3rd-party Native Apps
SDK Tools
Applications
Legacy Applications
Native ACCESS Linux Platform Applications
GTK+
Launchpads (Native, Garnet, Java)
ACCESS Linux Platform UI & Core Application Services Frameworks
Application Server
Bundle Manager
Attention/Alarm/ Notification Managers
HotSync®
Exchange Manager Plug-Ins
Mobile Services
Messaging Framework
Bluetooth Services
Custom Widgets
Garnet VM (Palm OS) Emulator
Java VM, JSRs
Multi-media Services
OTA Device Mgmt
OTA Data Sync
SyncML
Exchange Manager
Gstreamer
DRM
Global Settings
Telephony
Connection Manager
BlueZ
X Windowing System
Security Policy Framework
SQLite
H/W Acceleration
OpenSSL
User Space
ACCESS Linux Platform Security Module
Service/ Process Management
Power Management
Memory Management
Graphics & Display Drivers
Input Driver
File System
Multimedia Drivers
Networking Stack
Networking Drivers
Bluetooth Driver
Kernel
CPU
RAM
LCD
Input Device
NAND
Multimedia Hardware
GSM/GPRS/UMTS/EDGE, WiFi, Bluetooth Technology, etc.
Hardware
Open Source Software
Hiker Framework
ACCESS Proprietary

Android


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

VxWorks/Windows



- We have requirements
- We have identified a starting point

Step 3: Identify Candidate Architectural Drivers

- Choose the top level qualities using the scenarios
- 12 was #1

Source: Vehicle collision warning system

Stimulus: Driver's vehicle is about to collide with another object

Artifact: Forward facing radar and collision prediction system

Environment: Normal operation

Response: Warn driver; apply brakes; close all windows; tighten seatbelts; adjust peoples' positions

Response Measure: Response should occur less than .5 seconds after the conditions for a potential collision are detected

- Quality attributes – time efficiency; reliability

#2

Source - driver

Stimulus - hit deer

Artifact - emergency response system

Environment - collision

Response - system notifies police; deploys air bags

RespMeasure - 1/2 second

Qualities -

#9

Source: Vehicle

Stimulus: Comes in the close proximity of another vehicle or vehicles

Artifact: Collision Prevention System

Environment: Normal

Response: Notify vehicles that will be affected and accordinly accelerate or de-accelerate.

Response Measure: 0.3 seconds

Qualities -

Quality attributes

- IEEE Std. 1061 subfactors:

Efficiency
- **Time economy**
- **Resource economy**

Functionality
- **Completeness**
- **Correctness**
- **Security**
- **Compatibility**
- **Interoperability**

Maintainability
- **Correctability**
- **Expandability**
- **Testability**

Portability
- **Hardware independence**
- **Software independence**
- **Installability**
- **Reusability**

Reliability
- **Non-deficiency**
- **Error tolerance**
- **Availability**

Usability
- **Understandability**
- **Ease of learning**
- **Operability**
- **Comunicativeness**

Reliability

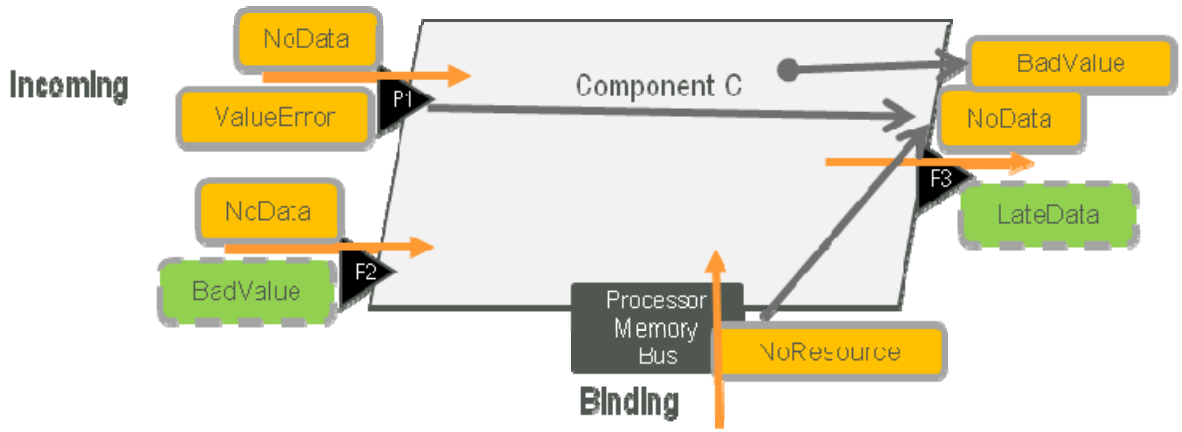
- Software Reliability: P(A|B)
  - **A**: Software does not fail when operated for *t* time units under specified conditions.
  - **B**: Software has not failed at time 0.
- Ultra-high reliability requirements for safety-critical systems (Draft Int'l Standard IEC65A123 for Safety Integrity Level 4):
  - Continuous control systems: **< 10^{-8}** failures per hour
    Airbus 320/330/340 and Boing 777: **<10^{-9}** failures/h
    This translates to 113,155 years of operation without encountering a failure
  - Protection systems (emergency shutdown): **< 10^{-4}** failures/h
    UK Seizewell B nuclear reactor (emerg.): **<10^{-3}** failures/h

What does it mean?

R (for 1h mission time)	Failure intensity
0.386	1 failure/h
0.9	105 failures/1000h
0.959	1 failure/day
0.99	1 failure/100 h
0.994	1 failure/week
0.9986	1 failure/month
0.999	1 failure/1000 h
0.99989	1 failure/year

Definitions

- Error – an incorrect result; sometimes used to refer to a mistake made during development
- Fault – some errors are manifest in the artifacts of the system
- Failure – when a fault is executed and the incorrect result is propagated to where it is visible



operational profile

- We must know which operation are used the most
- We operate the program according to that profile
- Record failures
- Do the math

- The **robustness** of software is related to how badly things go wrong when it fails to do exactly what it is supposed to do
- Reliability growth models

Design for reliability

- Simplicity - as complexity increases, probability of faults being injected increases
  - Avoid designs that require dynamic allocation
  - Avoid designs that require reflection
  - …
- Use mature technologies not fads
- Why C++ and not C?

Styles and patterns

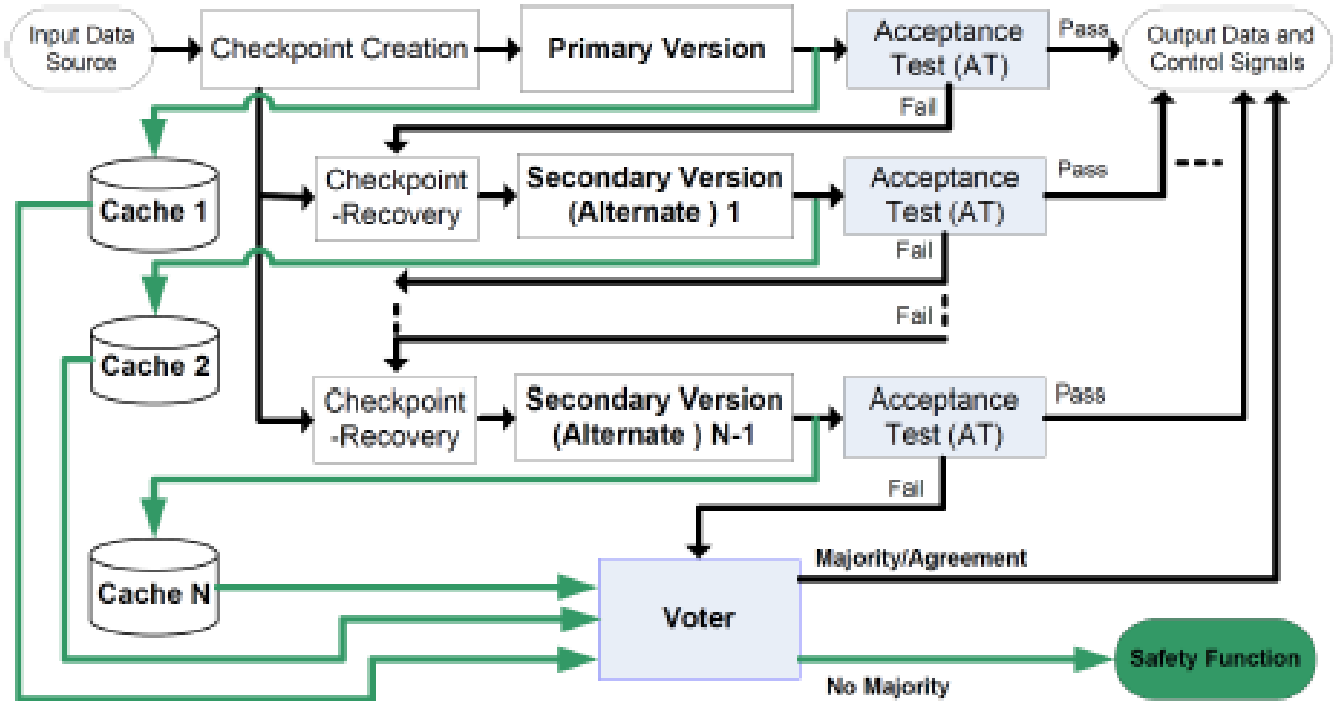
- An architecture style and a pattern are very similar
- A pattern may have more information, particularly more information about trade-offs among attributes.
- Which patterns enhance which qualities?

Fault Tolerance

- We can have faults
- We can even execute the faults as long as the effect does not propagate
-
-

  http://www.ijcaonline.org/volume18/number2/pxc3872900.pdf

Voting pattern



Avoid bad habits

- Stop writing lousy code
  - http://cwe.mitre.org/ - common weaknesses
- Design for failure
- Keep it Simple
- Test… test… test….
- Get in the trenches with Ops
- Safety first

http://www.javacodegeeks.com/2011/06/lessons-in-software-reliability.html

Time efficiency

- Avoid context switches
  - Function, thread, process
- Concurrency
- Parallelism