



How to develop software ?

Mostly: don` t...

Jerzy Prekurat
jerzy@jerzy.org



Definition

- Software Development Conference:

Long winded discussion how to do better something, that – in most cases - should not be done at all.



Example – `Sleepy Eye`

- Large integrator (EDS):
 - Developed program for ODBC data transfer
 - Broken `keep-alive` loop
 - Program periodically went to sleep
 - Solution: write program to wake it up
- World would be a better place, if those guys did nothing (or – did not exist)



Definition

- Software:

Detailed recipe how to perform a task, preferably simple enough for a dumb thing (like a computer) to implement.

First programmer - Imhotep



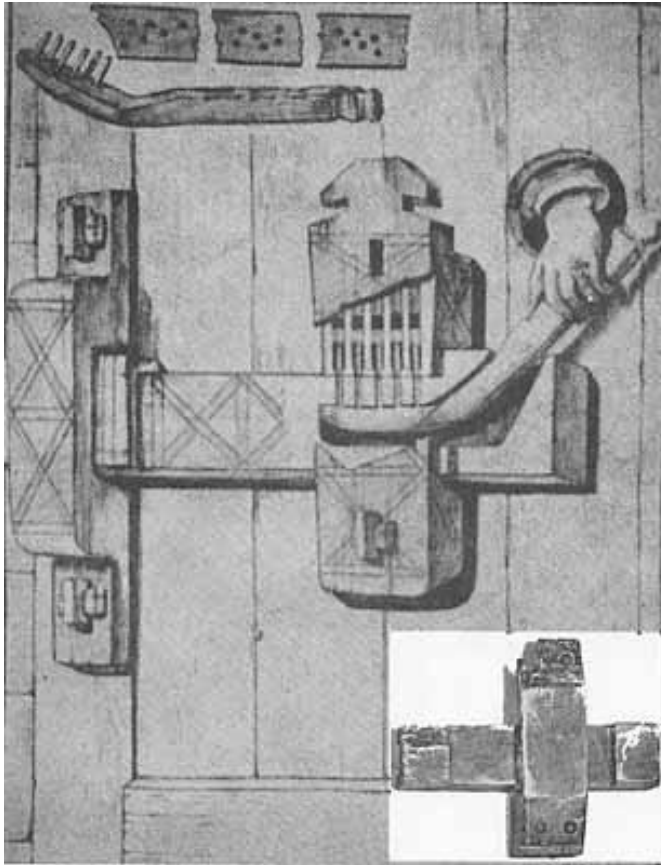
- Circa 2600BC
- Developed software to build a pyramid
- Language: hieroglyphs
- Compiler: none
- Interpreter: 10,000 Nubians

Pyramid v. 1.0



- Stepped Pyramid of Djoser
- A bit rough at the edges
- More pyramids were built...
- ...but software was irretrievably lost before the end of the Old Kingdom

Other lost goodies...



- Egyptian `Yale` lock (reinvented in 1848)
- Flash of genius working within immature structure...
- ...just like some software today
- Well, most have it`s "genius" bit set to 0



Monte Carlo story

- Perfect method to quickly approximate results in minimal time.
- Accuracy improves with additional calculations.
- I had to develop it twice (in Fortran)
- I counted 47 implementations by 1984, then - quit counting...
- What is wrong with this picture ?



Memory management

- “I am going to write memory management by Tuesday, then...” Does it sound logical ?
- “I am going to design a screw (rediscover America, reinvent the wheel etc.) by Tuesday, then...” How does this sound ?

Any similarities ?



Immature structure

- Lack of established mechanism to preserve the knowledge and to build on accomplishments of others
- Current status of Software Development

How do we grow up ?

The nut-case...

...and the screw to fit the nut

- Screw - invented by Archytas of Tarentum around 400BC...
- ...remained quite useless for over 2000 years...
- ...until the Industrial Revolution...
- ...everybody – including Archimedes and Heron – thought screw is a good idea, but...
- ...it did not fit the nut...



The nut-case continues

- Antoine Thiout – circa 1750 – lathe with screw drive (just like a framework...)
 - From now on craftsmen could produce much faster something that should not be created in the first place – unique screws
- Joseph Whitworth – 1841 – British standard screws (55 degree – round crest)
- William Sellers – 1864 – American screws (60 degrees – flat crest)
 - Result: British screws did not fit American nuts until 1948...



The anti-nut case

- Resistance to thread standardization: we need the screw to meet exactly the requirements of our product...
 - Science necessary to calculate the exact parameters of screws did not exist until 1930`s
 - See: software requirements documents...
- When you design a car – there is no time for reinventing screws, hinges and the like...
 - ...unless you happen to be GM, then you need Torx



Software screw – TCP/IP

- TCP stack was designed to recover from congestion related packet loss
- Radio links – interference causes most of the loss – requires different recovery mechanism
- Modification of Swansea stack – 3 days
- Writing new protocol (major vendor) - >\$250k...
- ...then – for compatibility – new protocol was used to wrap TCP



Kali VPN

- Kali is a flexible, secure router/fw/VPN
 - IPsec with 256-bit AES encryption
 - X.509 and CLR support
 - Packet filter firewall
 - Services: DHCP, NTP, PPP,SSH, custom services
 - Mock transaction tester
 - Packet blaster link test
 - OSPF
 - Connectivity: xDSL, X.21, V.35, up to 7 ethernet
 - Radio capability (802.11x)
- All that can be implemented for <\$300 per unit and more flexible than any other device



Beauty of little Kali

- CBN needs sophisticated features, but not complexity and “people pleaser” gimmicks
 - Need: multiple interfaces; don't need: web based management on port 80
 - Need: 256bit AES; don't need: XAUTH extensions
- Cost:
 - Kali – 7 interfaces: \$400
 - Cisco – 3 interfaces: \$15,000
- Flexibility:
 - We can add what we need – tests etc.
- Control and more



How did we get here ?

Incremental development

■ Advantages:

- Low cost
- Immediate payback...
- ...pay as you go
- Immediate feedback...
- ...learn as you go
- No big mistakes – no time to make them
- Incremental risk
- Open Source leverage, many modules are waiting out there

■ Modular design comes as a necessity

■ Problems:

- Never enough time...
...we could make it better, if...
- Never enough resources...
...we could build more, if...
- Ad-hoc architecture
- Small mistakes
- Periodic re-engineering

■ Management challenge: programmers always want to “boil the ocean”



How did we get here ?

Multi-purpose development

- Advantages:
 - Very low cost...
 - ...build once, sell many times...
 - ...in different products
 - Stability – debugged in many contexts
 - Effective life cycle management
 - Generic specifications
 - Adaptability to new tasks
- Open Standards design comes as a necessity
- Problems:
 - Consistency
 - Versioning
 - No “paper fortress” – cannot hide bad design behind “requirements document”
 - Expertise – not tools
 - Hostile to bad habits
 - Constant back-porting
- Management challenge: leader has to know what he is doing (rare)

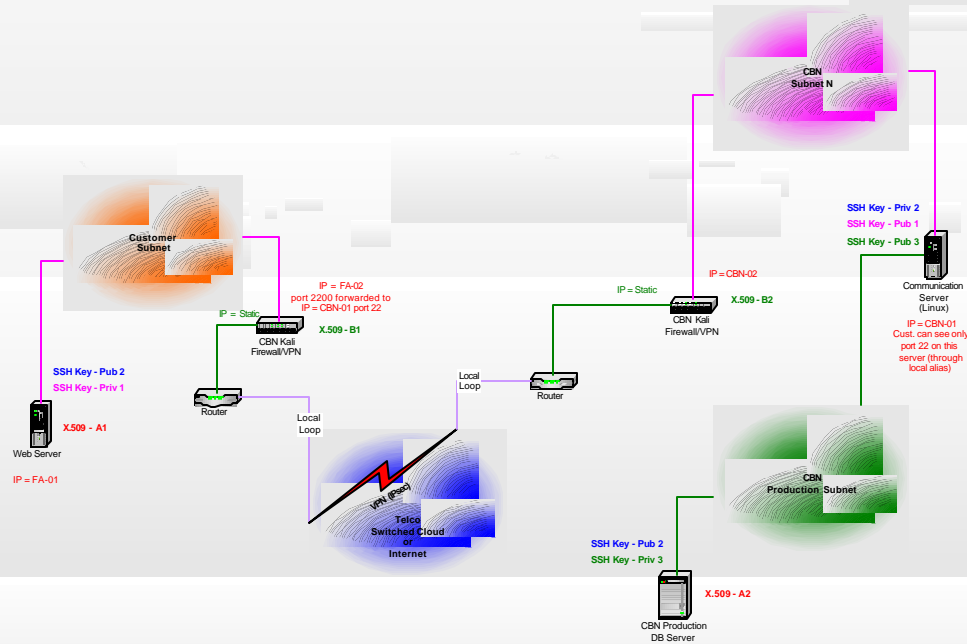


Case for Kali: Freedom of choice

- Need: dial-up server
 - Outdated technology, mostly discontinued
 - Available devices >\$2000
- Student added PPP server function to Kali
 - Less than 3 days of work.
 - Cost <\$300 per device, \$8500 savings on 5 devices needed
 - Faster than ordering equivalent gear
- Technology will be reused
 - By support group (~5 more devices)
 - Available for other projects

Exotic example – file transfer

Data Exchange Connection Architecture v. 1a (Proposed)



Design principles:

1. Client on external network cannot see CBN network (no trust).
2. Client on CBN network cannot see customer network (no trust).
3. All transactions appear to be local - port 2200 on local host.
4. All "magic forwarding" accomplished on a pair of Kali devices.



Case study - Kali:

100 nodes in Guatemala

- Commercial products (cost to CBN)
 - Entrust CA - \$250,000
 - Central site devices, fail-over Cisco 3030 >\$40,000
 - Remote nodes – Cisco 3015 – \$1,400,000
- Total - **\$1,690,000**
 - Likely too expensive to implement
 - Common solution: **cut down security**
- CBN Kali products (cost to CBN)
 - CBN CA - \$20,000
 - Central site devices, fail-over Kali 4801 - \$1,000
 - Remote nodes – Kali 4501 – \$30,000
- Total - **\$51,000**
 - Feasible on most contracts
 - We can afford **secure network** anywhere



Summary

- We delivered more than we promised...
- ...early...
- ...now we are victims of our own success
 - Kali met all short-term objectives, including very exotic ones...
 - ...but all components need improvement to meet long-term objectives
 - Kali was built for small scale deployment...
 - ...but its current deployments already exceeded small scale
- How many software projects are in this position ?