CONTROL SYSTEM EVOLUTION

and the Importance of Trial and Error

Philip Duval and Mark Lomperski, DESY, Hamburg, Germany
Jaka Bobnar, Cosylab, Ljubljana, Slovenia

Disclaimer

- Not a status report (*per se*)
- Information you probably already know
- Not focused on any particular control system problem
- Is focused on how you go about solving control system problems …

- Consider this a blog …

Contents

- Goals and Problem solving
  - The God Complex / Einstellung
  - Group influence: The Mathew Effect
  - Trial and Error
- Evolution
  - Darwinian
  - Software
- Trial and Error Examples
- Conclusions

Goals and Problem solving

Suppose you have a non-trivial problem to solve …

- The God Complex

"No matter how complex the problem, you believe that your solution is correct."

- Archie Cochrane

  - after all: *You're an expert!*

- Einstellung
  - predisposition to solve a given problem in a specific manner even though better or more appropriate methods of solving the problem exist.

a tendency to rush to "*the*" solution !

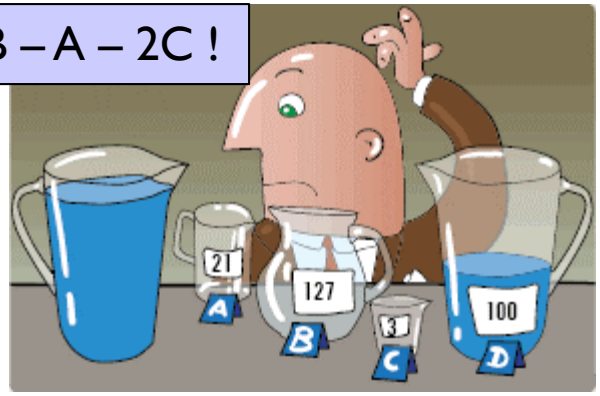
Einstellung

- e.g. (Abraham Luchins)
  3 water jugs: A, B, and C.

  A: 21 units, B: 127 units, C: 3 units. How to get 100 units?

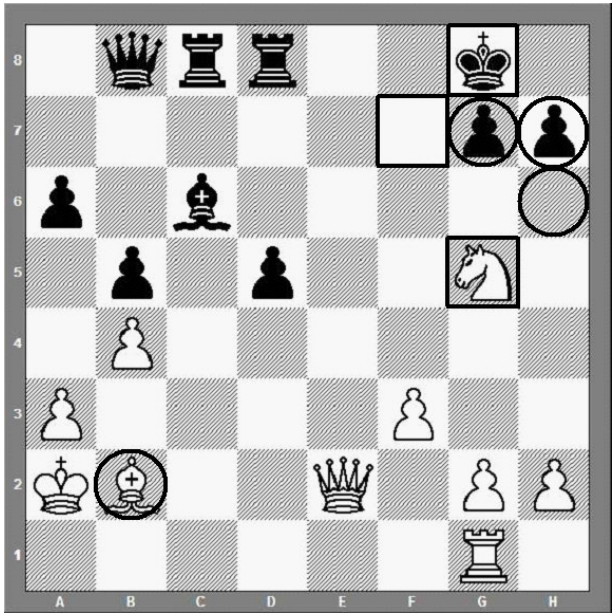
  A: 15 units, B: 39 units, C: 3 units. How to get 18 units?

B – A – 2C !



- It's white's move:
  - How many moves needed for checkmate?

'smothered mate' in 5 !



Group Influence: The Mathew Effect

- The *Mathew Effect*
  - ***accumulated advantage***
  - 'the rich get richer,

    and the poor get poorer.'

> For unto every one that hath shall be given, and he shall have abundance: but from him that hath not shall be taken even that which he hath.
>
> - Mathew 25:29

- Your opinion often depends on others' opinions !
  - e.g. two versions of web site to download garage band music (experiment by Duncan Watts):
    - version #1: **could not see** how many times a song had already been downloaded.
    - version #2: **could see** how 'popular' a song had already been.

Trial and Error Loop

- No matter what approach we try …
  - Decide what constitutes a solution and

```
while (true)
{
    • try something
    • if (problem is solved) break;
    • tweak the something
}
```

note: if you set the bar too low, you might end up back in the loop sooner than you think …

Evolution

- Evolution *naturally works via trial and error.*
  - Darwinian Evolution
    - Replicating units : *genes*
  - Software Evolution
    - Replicating units: *memes*

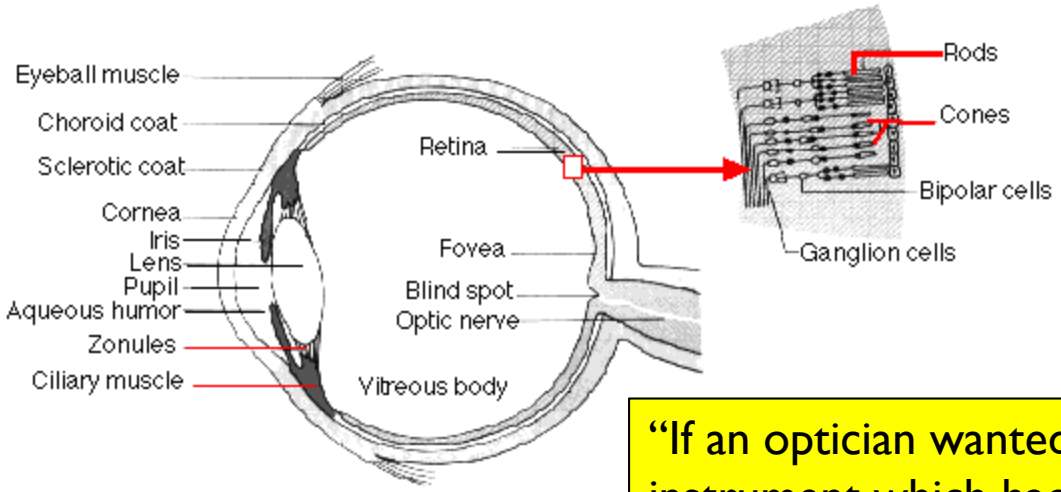
chance vs. design

  - If you can *replicate*, then you *survive* into the next generation.
  - Implicit *drive* to be *better* (your chances of survival are then improved) => **evolution**.

Evolution and trial and error

Darwinian evolution and natural selection

- the **eye** : fantastic *finished product* with a clumsy design !



Refactoring **NOT** an option !

"If an optician wanted to sell me an instrument which had all these defects, I should think myself quite justified in blaming his carelessness in the strongest terms, and giving him back his instrument."

- Hermann von Helmholtz

Software Evolution

- We *can* refactor *bad* design decisions or clumsy *spaghetti code* !
  - but is it *always* a good idea?

> If a complex system is running *flawlessly*, is it a good idea to *change a running system*?
>
> - If it ain't broke, don't fix it !

  - sometimes better to tweak than refactor ?

Software Evolution

- Manny Lehman (1980):
  - 3 Categories of Software:
    - S-Program – written to perform a specific task

      printf("hello world\n");
    - P-Program – implement a set of procedures which determine what the program can do.

      play chess
    - E-Program – perform a real-world activity. Adapt to circumstances and environment in which it runs.

E-Programs necessarily evolve !
Acc. Control System applications/libraries are E-Programs!
(as much as we wish they were P-programs …)

Software Evolution: Lehman's Laws

- 1.(1974) "Continuing Change" — an E-type system must be continually adapted or it becomes progressively less satisfactory
- 2.(1974) "Increasing Complexity" — as an E-type system evolves, its complexity increases unless work is done to maintain or reduce it
- 3.(1974) "Self Regulation" — E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal
- 4.(1978) "Conservation of Organizational Stability (invariant work rate)" - the average effective global activity rate in an evolving E-type system is invariant over the product's lifetime
- 5.(1978) "Conservation of Familiarity" — as an E-type system evolves, all associated with it, developers, sales personnel and users, for example, must maintain mastery of its content and behavior to achieve satisfactory evolution. Excessive growth diminishes that mastery. Hence the average incremental growth remains invariant as the system evolves.
- 6.(1991) "Continuing Growth" — the functional content of an E-type system must be continually increased to maintain user satisfaction over its lifetime
- 7.(1996) "Declining Quality" — the quality of an E-type system will appear to be declining unless it is rigorously maintained and adapted to operational environment changes
- 8.(1996) "Feedback System" (first stated 1974, formalized as law 1996) — E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base

Error and Trial …

- April 21, 2010
  - Released:

update (DAT No 5958) which incorrectly identified a vital Windows XP system file (svchost.exe) as a virus !

Whenever you deploy: have a serious look at the costs of error …

Control System Evolution

- Some *memes*:
  - Software:
    - sockets
    - threads
    - processes
    - signals
    - …
  - Architecture:
    - client-server
    - publish-subscribe
    - producer-consumer
    - database vs. device-server view
    - …

Building blocks …

Trial and Error : Control Systems

Trial and Error at the protocol level:
the *loaded server problem* re-visited …

- **Contract coercion (TINE)**
  - Magnet server(s)
    - Central Magnet Server + set of group/section servers
  - jddd/ddd panel clients + MatLab et al. creating >80% CPU load on some servers
    - tend to acquire information from 100s of PSCs *one-at-a-time* !
    - tend to 'synchronously poll' the server
  - Coerce a client's synchronous request for **one** thing into a monitor request for **all** things …

Contract Coercion

Device server model

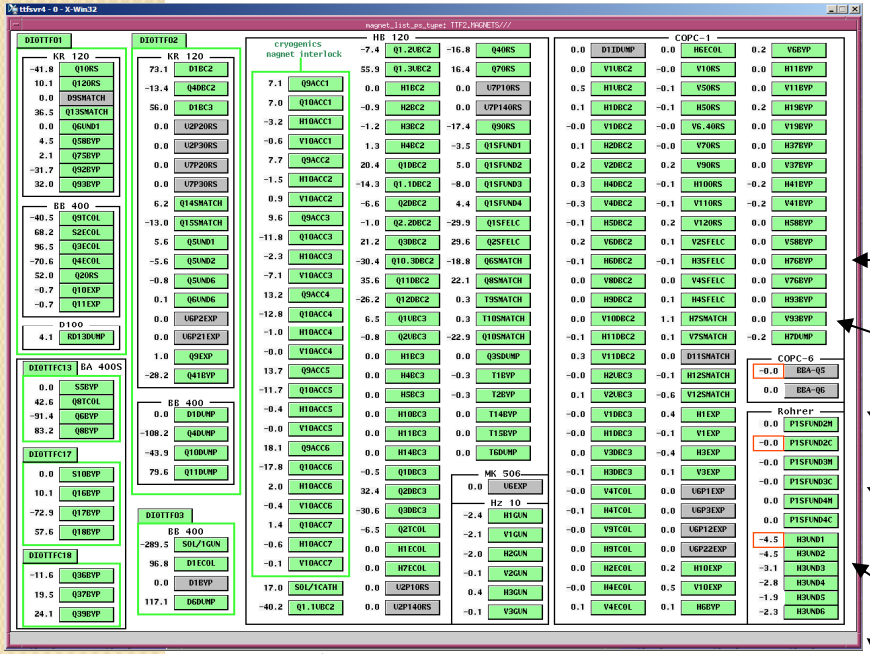
- **Multi-Channel Arrays (MCA)**
  - An *array of all values* of all devices for a given *Property* with a well-known order.
    - same units, settings, etc.
  - Registered Property can declare itself to be an MCA.
  - Strict OO Device Servers can declare device *groups* with MCA characteristics.
  - e.g. **BPMs, BLMs, IonPumps, Temps, PSCs**, etc.
  - Request for *Current.RBV* for device 'Q3FL2SASE4' knows that 'Q3FL2SASE4' is element #37 in an array
    - renegotiate the request to send the entire array !
    - inform client to retrieve element #37 to get the value it wants
  - **End Result:**
    - the server is only ever handling a single request for the entire array !
    - No one changes any code at the client side !

Presented PCaPAC 2010

FLASH DDD PS Panel with TINE Multichannel Polling

10 connections/property (was ~260)

50 contracts (was > 1000)



STEERER/SHGROUP (57)

STEERER/SVGROUP (48)

QUAD/QDGROUP (40)

DIPOLE/DIGROUP (19)

UNDULATOR/UNGROUP (7)

MAIN/MNGROUP (6)

SEXT/SXGROUP (3)

SOL/1GUN (1)

SUN (multi-server)

TTMAG-AB/GROUP

(PC104, 41 elements)

TTMAG-CD/GROUP

(PC104, 41 elements)

Trial and Error

- Contract Coercion and Trial and Error
  - Initial local testing: solving threading issues, etc.
    - deploy and work-around … ?

2010

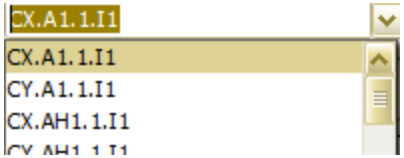
  - What happens when the 'element list' changes ?

"Q3FL2SASE4" is now element #38 and no longer element #37 !

    - *reset MCA* signal

2010

  - Only ever want one element as a combo box selection ?
    - MCA link *idle time*



2013

  - What happens when one of the array elements is in an error state and the others aren't?

Do they all claim 'hardware error'?

    - *format elevation*

2016

pass array of 'value-status' pairs !

Contract Coercion: *Cost of Error*

- *Trial-and-Error* exists in the **control system libraries** !
  - ~ *open-heart surgery*
  - any errors are *systematic* !

- *But* …
  - lots of *testing* prior to real deployment !
  - *mainstream* applications should be okay
  - if an error makes it past the unit tests …
    - likely to affect only a *small percentage* of components
    - rollback if necessary …
    - => fix the problem and expand the unit tests …

Plug and Play

Trial and Error at the services level

- TINE has an Equipment Name Server (ENS)
  - Servers *plug* into the **ENS**
  - Clients *acquire* addresses from the **ENS**
  - Must guarantee a *unique address* for each control system element !

not only can we NOT allow an "I'm Spartacus" problem, but …

⇔

if I think I'm controlling "/XFEL/Magnets/QD.448.B2" then I sure as hell better be controlling "/XFEL/Magnets/QD.448.B2" !!!

  - Must accept only valid names !

"My Cool Server !" is not an allowed name !

  - Must work automatically (no administrator required)!

but: only an administrator can assign an *Importance Level* …

Plug-and-Play: *Cost of Error*

- *Trial-and-Error* exists in a **system service**.
  - *semi*-critical
  - control system systematics already have a *fallback mechanism* if the service fails !

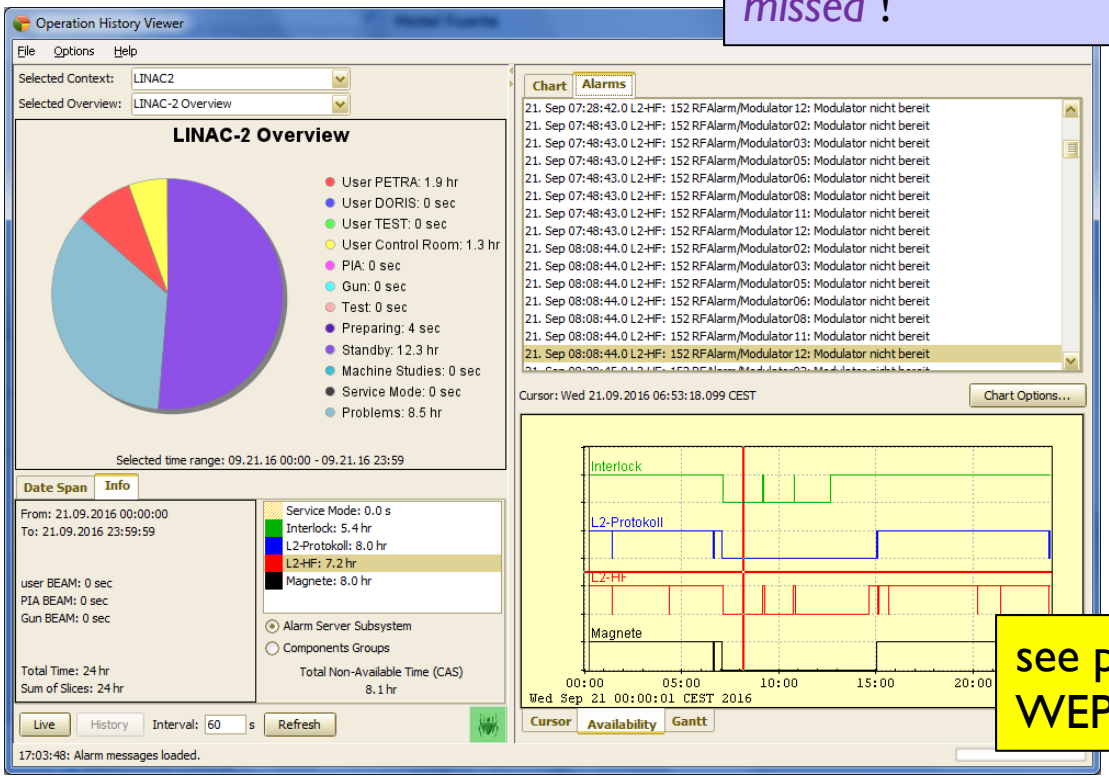
- *But …*
  - lots of *testing* prior to deployment
  - no *catastrophe* on error !
    - most people won't notice
    - some might be annoyed (*and they will report the error*!)
  - error most likely on *plug* (server-side) and not *play* (client-side).
  - *rollback* the service until the problem is fixed.

Automated Availability Statistics

Trial and Error at the application level

- bean-counting
  - machine states
  - subsystems with fatal alarms

A maintenance day at the linac …
fatal alarms get the *blame* for non-availability !
they must be *real* and they *can't be missed* !



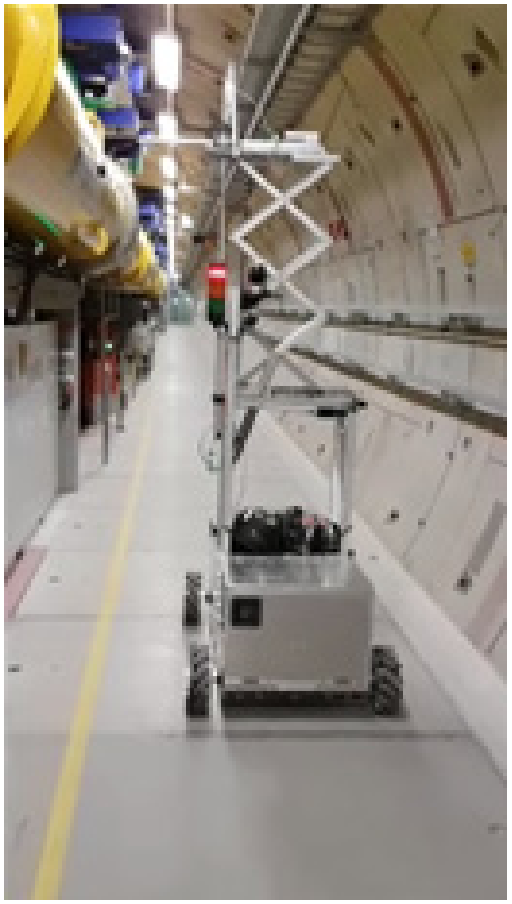
see poster:
WEPOPRPO18

Automated Availability : *Cost of Error*

- *Trial-and-Error* exists in a **control system application**.
  - *non*-critical
  - only users of the application will be affected.
    - how popular is the application?
- *Trial-and-Error* exists in the reliability of **system services** (*alarms*).
  - *non-critical* to operations
  - *non-useful* if information not reliable !
- *But …*
  - This particular application is being designed to allow post-corrections of statistics !

And some hardware …

- **MARWIN**
  - Automatic radiation measurements and monitoring in the tunnel
  - Lots of trial-and-error !
  - Costs of error ?
    - isolated from operations
    - (well … it could go crazy and start attacking equipment …)



Conclusions

- Unless you're under an extreme time constraint …
  - Resist the psychological pressures to *play it safe* …
    - Be aware of the cost of error post-deployment …
  - Don't rush to implement *the* solution to some new problem !
    - beware of incremental improvement toward a dead end …
  - Engage in healthy second-guessing !
  - Teamwork *can* be unproductive if not counterproductive !

The reasonable man adapts himself to the world.
The unreasonable one persists in trying to adapt the world to himself.
Therefore all progress depends on the unreasonable man.

- George Bernard Shaw