RECENT TRENDS IN ACCELERATOR CONTROL SYSTEMS

igor.verstovsek@cosylab.com

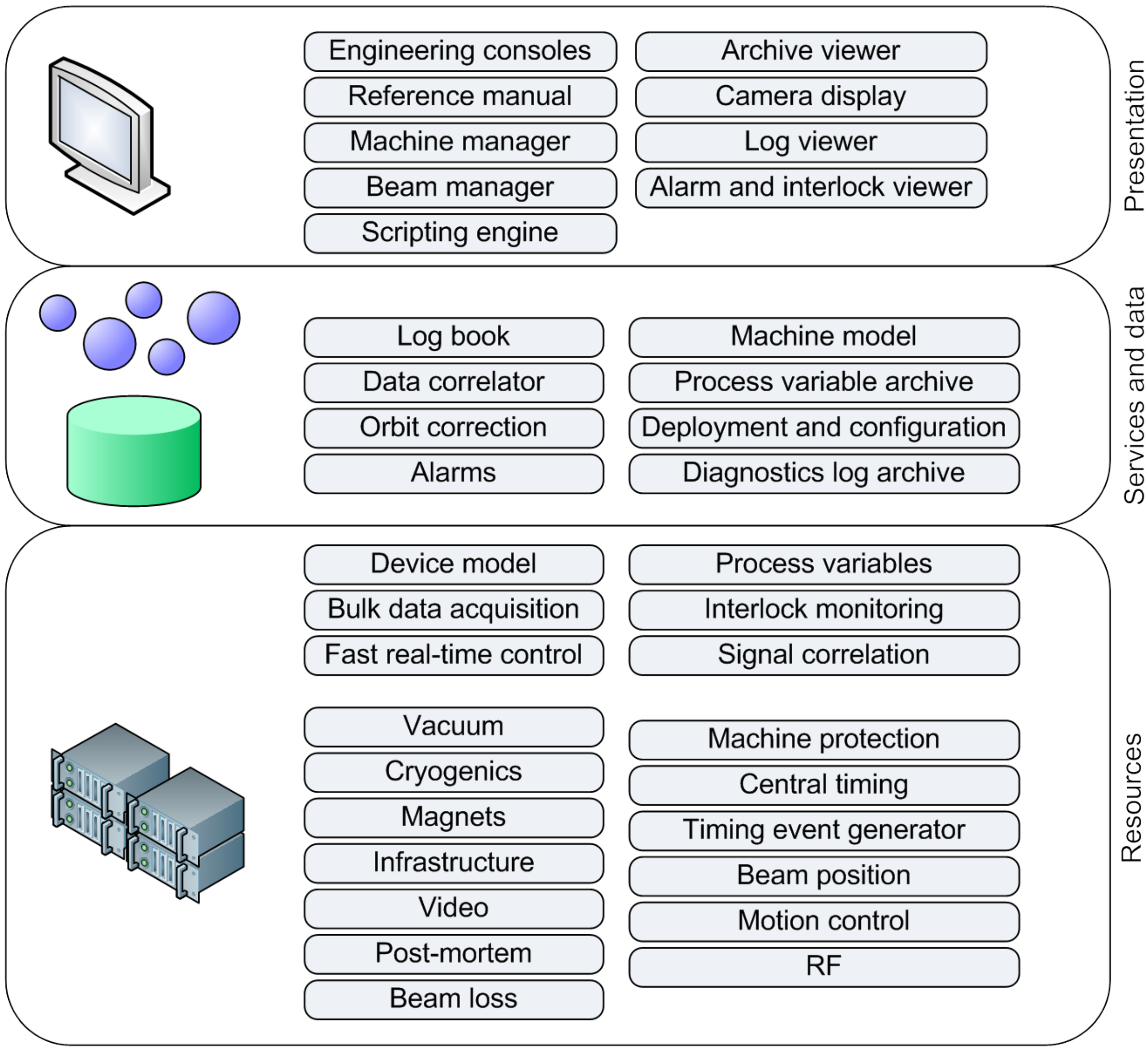
COSYLAB

Let‘s first define what is a CS

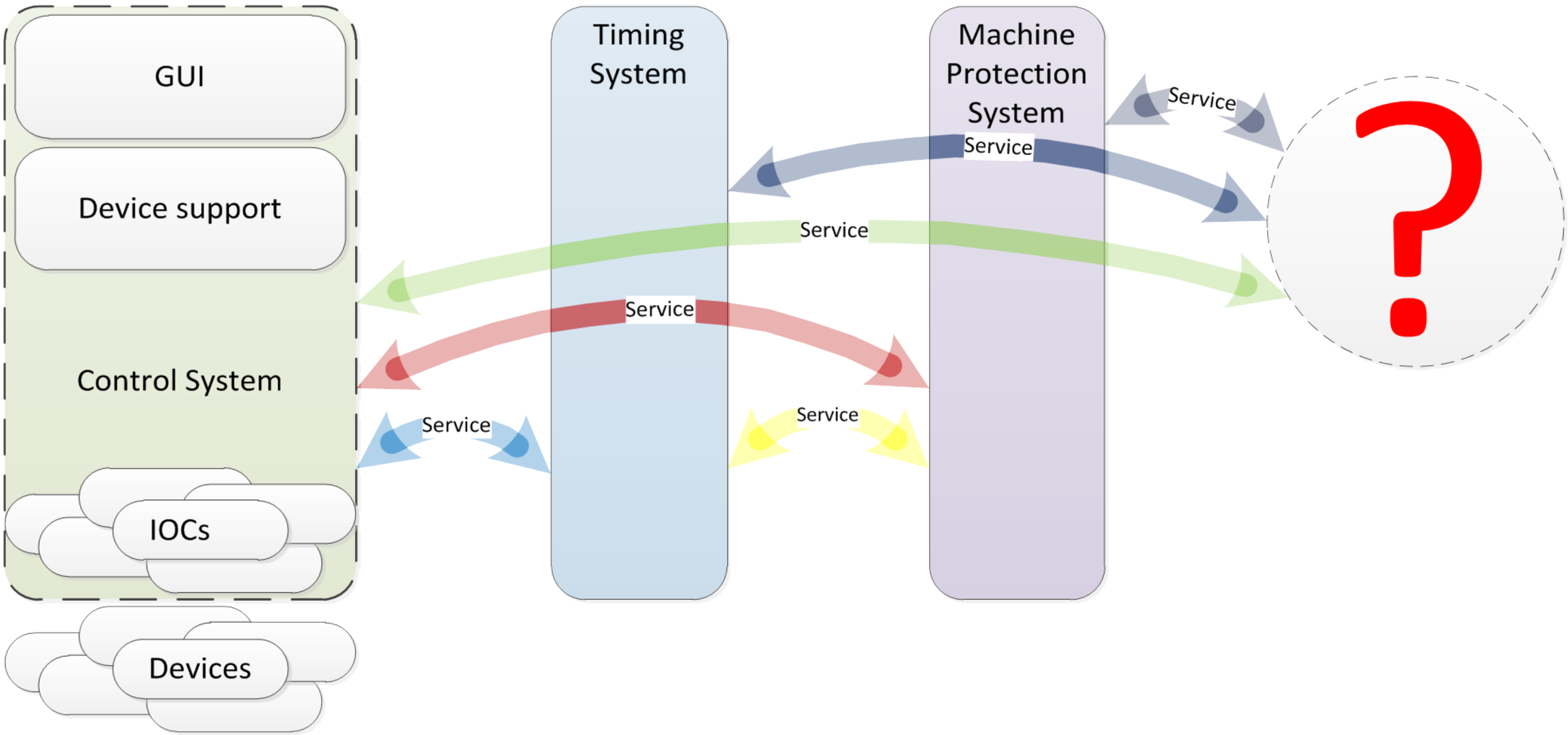
- Control System (CS) is not a shrink-wrap package with an installation wizard, but rather a service
  - Engineering according to specifications
  - Configuration of packages like EPICS, TANGO, FESA, TINE, DOOCS, MADOCA, LabView…
  - Outsourcing or in-house software and hardware development
  - Installation

Main CS Components

- What has to be in real time?
- Interconnection of components and services
- Which components are getting more important?



Define subsystems with care



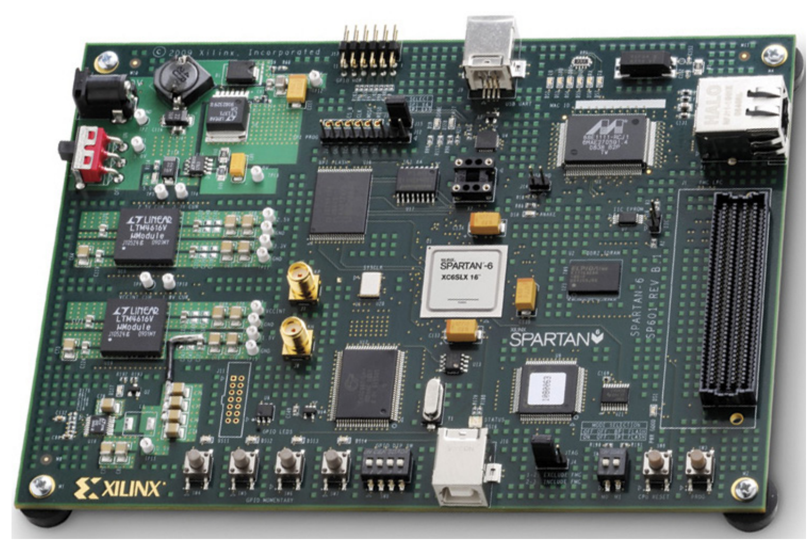
- List ALL needed services
- Define systems without duplicating services
- Understand connections between systems
- Define interfaces between systems

Role of CS in the project

- **Relatively low technical risk**
- **Higher organizational risk**
  - Collaboration across all the departments
  - Control system comes late in the project
  - Integrates with most of other subsystems
- Control Systems are an **engineering** discipline like all the others, but with an even more complicated cycle
  - Write specifications
  - Architecture
  - Design
  - Prototyping – **fun part**
  - Test procedures
  - Implementation (coding)
  - Documentation
  - Testing
  - Debugging
  - Acceptance
- Iterative development (evolution through upgrade phases)

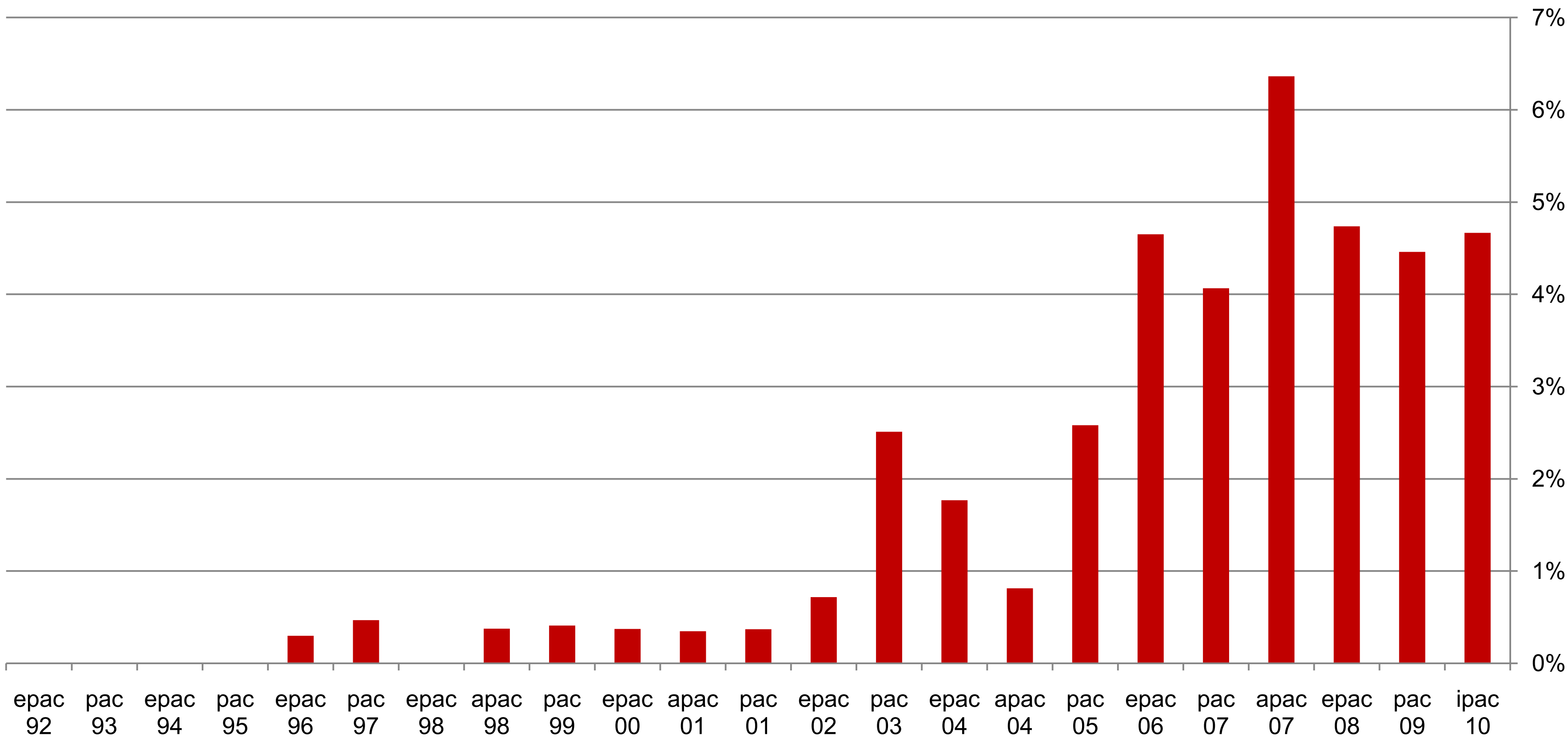
Trends – the example of FPGA

- FPGA: Field Programmable Gate Array
  - Integrated circuit designed to be configured by the customer after manufacturing
    - “reduces hardware development to configuration”
  - Obvious benefits
    - Many inputs and outputs, parallel processing, full synchronization, real time, flexible,…
  - Applications
    - LLRF, MPS, timing, DAQ,…



Trends – the example of FPGA

String “FPGA” in conference articles



Trends – the example of FPGA

- Can (& do) replace custom hardware, but
- Complexities and potential pitfalls of software engineering appear in hardware
- Risk of postponing key design decisions because of false sense of flexibility (cost of changes becomes prohibitive late in the project)

Mastering complexity requires BRAIN power (more then CPU power)

- Advances in technology often give false sense of complexity reduction. Examples
  1. FPGA development environment. For few 100 euros a PC card with free, well supported tools. Feels like an easy start, but it's a marginal win. True challenge is in domain expertise and system knowledge.
  2. System 2.0 syndrome. With new tools and technologies, we'll fix ALL the shortcomings of the system 1.0 … result: a proven, working system is replaced by an over-architected, heavy framework with late delivery.

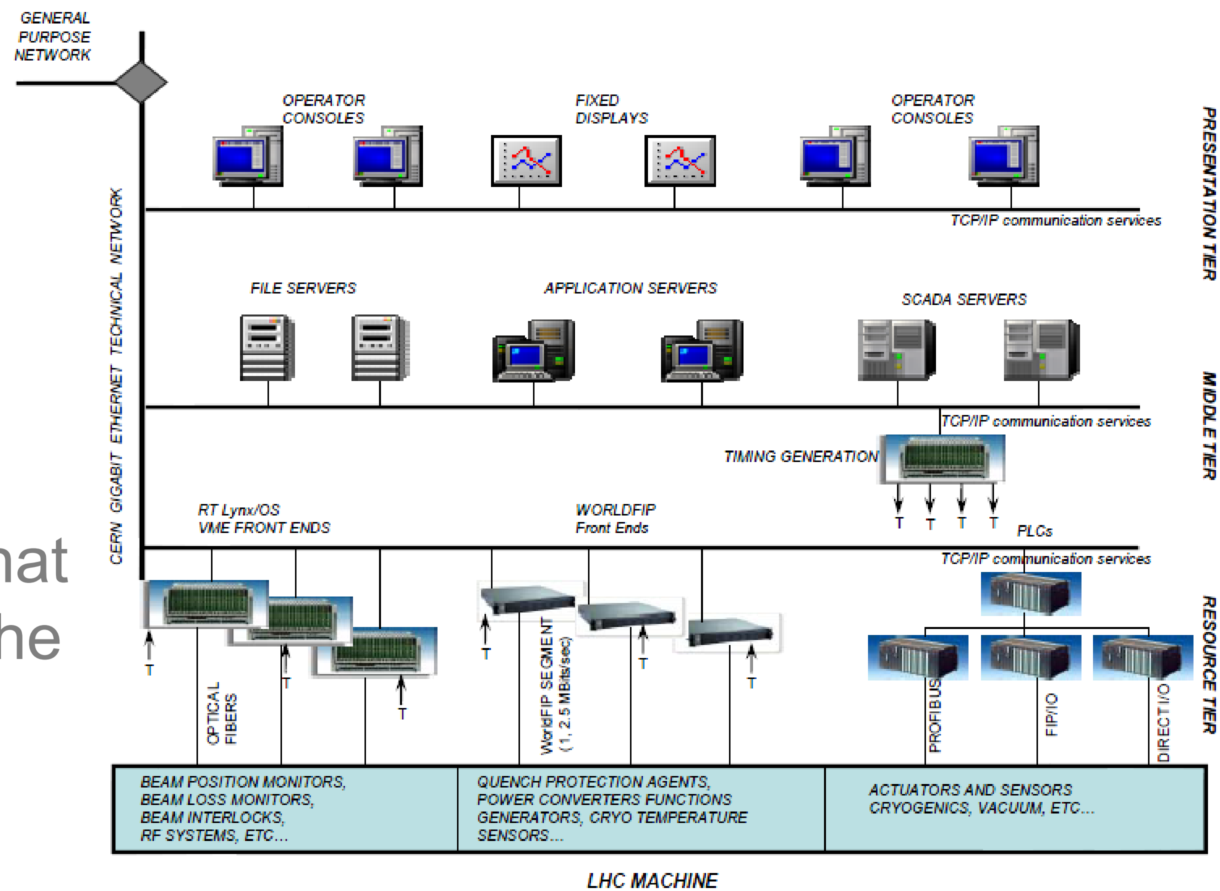
→ Prudence, use of proven techniques

Elder Mathias (Canadian Light source) : *Be realistic on what is needed to commissioning the machine versus what is needed for optimal performance.*

Middleware or software bus

- What is their purpose, are they really different
  - **Yes** in terms of technical implementation
  - **Probably yes** in terms of performance
  - **No** in terms of what they provide for the CS

EPICS, CMW/FESA, TANGO, TINE, DOOCS, MADOCA,...



Why is EPICS so popular?

- A very strong user and developer community
- A large number of supported devices, and a relatively small set of interfaces
  - Universal motion control record]
- Lightweight on dependencies
  - Does not depend on relatively complex middleware
  - few central services that would be single-points-of-failure.

Hardware platforms

- VME, ATCA, cPCI
- Criteria for evaluation:
  - Vendor support, maturity, longevity, maximum transfer rate, topology, form factor, availability, software support, user base, etc.
- Why don‘t all the labs make the same choice?

Hardware platforms

	VME	ATCA	cPCI
Vendor support	High/Declining	Low/Growing	Medium/Stable
Maturity	High	Medium	High
Longevity	Medium	High	High
Max. transfer rate	VME: 40MB/s VME64: 80MB/s VME64x: 160MB/s VME320: 320MB/s	1Gbps, 10Gbps (Gigabit Ethernet); 250MB/s/lane (PCIe)	PCI: 133MB/s PCIe: 250MB/s/lane (up to 16 lanes)
Topology	Master-slaves	Star Dual star Full mesh	Master-slaves

Hardware platforms

	VME	ATCA	cPCI
Form factor	6U (64 bit) 3U (32 bit)	12U (ATCA) 2U (μTCA)	3U
High availability	Medium	High	Medium
Software support (Linux, EPICS)	High	Medium	Medium
Cost	High	High	Medium
Users	SNS, SLS, Diamond Light Source, NSLS II, …	XFEL (LLRF), ITER, TPS (considering)	ALBA, TPS, CERN (LHC collimation), LANL, ORNL, ITER (planned)

Hardware platforms

- Main criteria for selecting hardware platform should be
  - Usability
  - Longevity
  - NOT “top performance” or coolness factor

Acceptance by majority in the industry.

Jean-Francois Gournay (CEA) *: stay with well-improved solutions as much as possible (we use the same analog IOs and binary IOs VME boards - still manufactured - for 20 years.*

Some more dilemmas

- Components off-the-shelf (COTS) vs in-house
  - Cheap COTS that must be modified to fit our needs,
  - or expensive specialized components
  - The name of the game is development cost: COTS is cheap only if there is a mass market behind it: high volume or high margins

- Open source vs. commercial

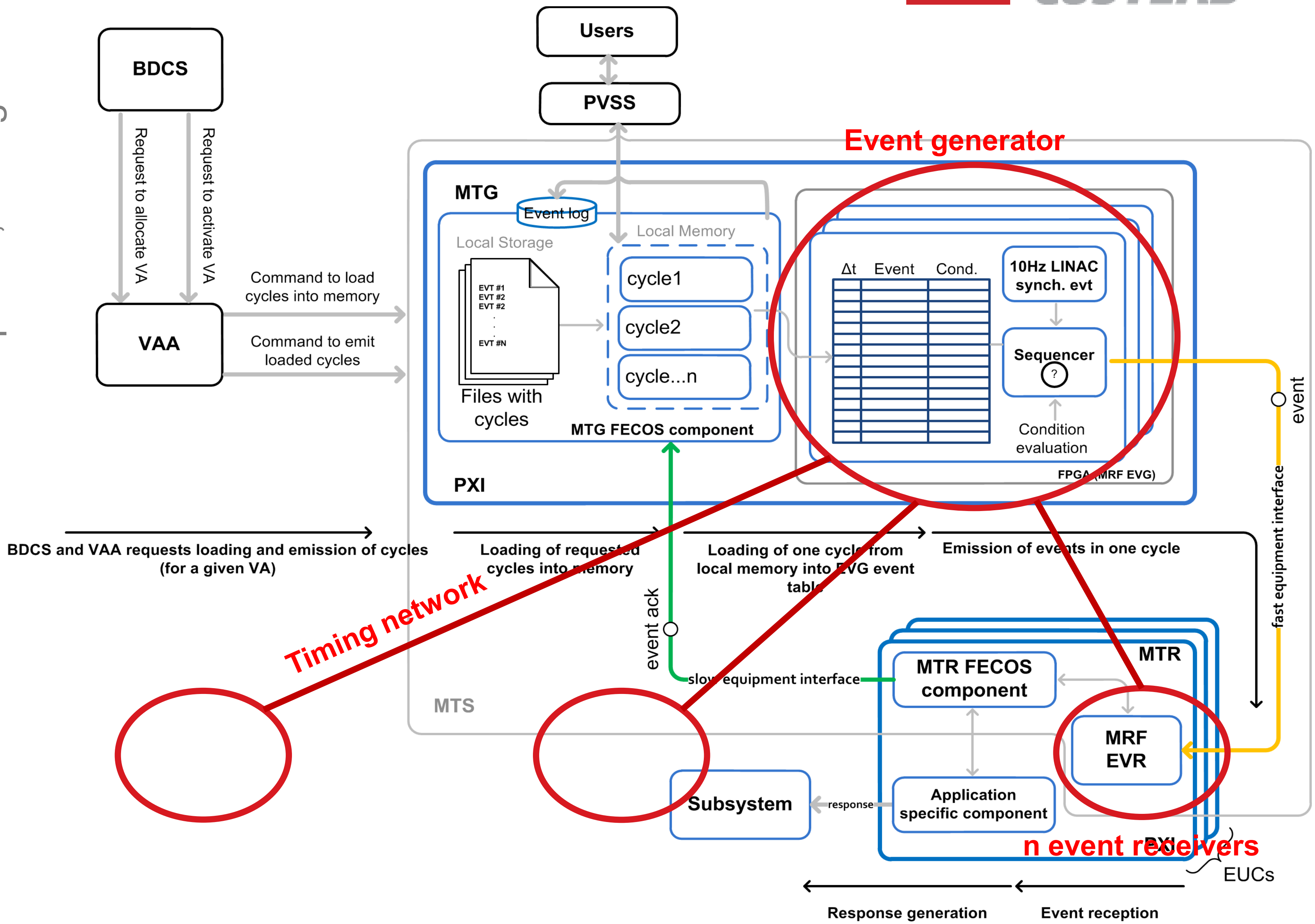
So it's all just Plug & Play?

- Or is it Just work and no play?
  - Never possible to buy the entire CS off the shelf
  - Integration work, specific to every accelerator
    - Even on identical accelerators (say, medical)
    - Possible to even buy system integration services
  - Everybody knows the control system will work, but it‘s still work that must done by somebody

- Black, white or magic box?
  - Prepackaged components, bundled together
  - E.g. Micro research Finland timing platform

Example: timing system

- requirement analysis,
- architecture design
- development, testing…

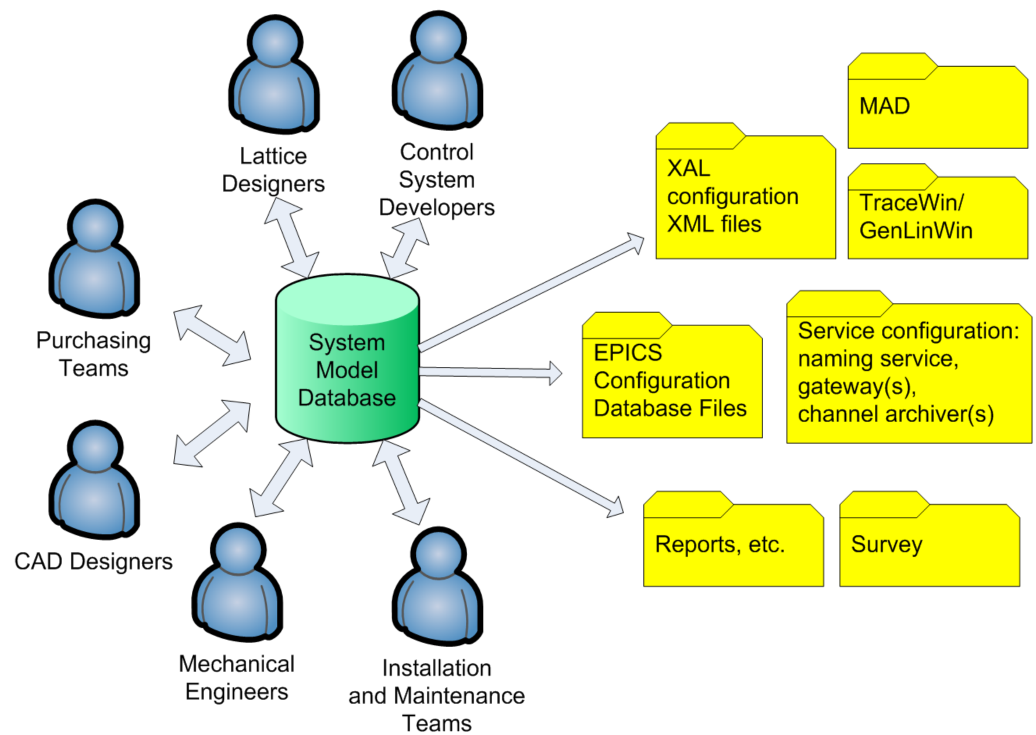


See poster, MOPC148

Top-down approach … and why it never works

COSYLAB

- Automatically generate as much of the control system's components as possible
- Principal input: a high-level description of the system (e.g., the accelerator's lattice)
- Use of system engineering tools and model-driven architecture
- The database(s) contain:
  - inventory information (equipment and its location, reference to manuals, reference to purchasing information, …)
  - cabling, connectivity and topology information
  - control system process variables, processing rules, …



Poster: Klemen Zagar et al, WEPC151

Future challenges

- Large international projects with in-kind contributions: not technical, but managerial challenge.
  - How can CS help
  - How should we design the CS to solve these issues

Development process – our experience from 20+ projects

- Start with requirements very early
  - They will change later in any case, no matter how long you wait for the “final” requirements!

- Standardize development
  - Applies to the whole cycle: design, implementation and testing procedure.
  - More important than standardizing components.

Matt Bickley [JLab]: *The aspect that I think has been most helpful has been standardization of hardware and software[...] Another decision was the choice to develop and rigorously adhere to standard testing and implementation procedures.*

Development process – our experience from 20+ projects

- Vertical prototypes from the beginning
  - With integrated software and hardware
  - E.g. vertical column (MedAustron), Control Box (ESS), Fair Host Machine (GSI/FAIR),…
- Iterate frequently
  - Yearly cycles
  - First specific requirements usually come when people comment on the first prototype!

Should we expect clear technological "winners"?

- Accelerator CS is a very broad field with specific needs for every job
  - Timing needs
  - Safety needs
  - Reliability uptime needs
- Many installations are experimental by nature

Many arguments for very diverse approaches that blur the overall picture

- Computing power/$ grows faster then project size
- Increasing expectations in power and flexibility of the CS
- Every added (cheap) CPU increases the entropy of the system

Increasing challenge of managing the added complexity

Conclusion

- CS has shifted over the years
  - From research to engineering
  - From performance to integration challenges
- But architecture and platforms are more or less stable
- So CS integration task is how to integrate everything into the CS in-time, on-budget, and with a low-risk by using an increasingly large number of off-the-shelf components.

THANK YOU!

Igor Verstovsek, igor.verstovsek@cosylab.com

COSYLAB

Tel.: +386 41 934550

Web: www.cosylab.com

COSYLAB