SURVEYING SOFTWARE TECHNOLOGY FOR ACCELERATOR CONTROL SYSTEMS

Thilo Friedrich, MAX-lab, Lund and KTH Stockholm, Dept. of Machine Design, Sweden
Prof. Martin Törngren, KTH Stockholm, Dept. of Machine Design, Sweden

Abstract

Virtually all accelerator based research facilities nowadays use a mixture of software libraries, tools, protocols and development techniques to address the facilities' various control system[*] requirements efficiently. Many of these technologies are open-source and shared between laboratories to various extents. Motivated by the planning of MAX-lab's new light source project, the MAX IV facility, we have conducted a state-of-the-art survey of these technologies, which will serve as a knowledge base for upcoming design decisions. This paper provides a summary of the topics and conclusions of our survey. In this scope the survey compares software technologies with respect to user features (scientific analysis and operation requirements), quality requirements (integration, performance, services, reliability, security, safety), and other issues. Control system design goals are beneficial long-term effects on future improvements, development costs and maintenance costs.

THE MAX IV PROJECT

MAX-lab [1] is currently in the late proposal phase for a new synchrotron light source facility called MAX IV [2], built in Lund, Sweden. The new facility provides synchrotron radiation of high quality over a broad spectral range, stretching from IR to hard X-ray regions. Additionally, a Short Pulse Facility (SPF) provides intense, short x-ray pulses in the femtosecond domain.

The MAX IV facility design includes three 3rd generation storage rings operated at different electron energies. The main MAX IV light source is a new low-emittance storage ring, operated at 3.0 GeV with 20 straight sections, which is optimized to approach theoretical limits for hard x-rays in its energy domain. The existing MAX II and MAX III storage rings are transferred to the new MAX IV site and receive an upgrade. MAX II, operated at 1.5 GeV, covers the soft x-ray region; MAX III, 700 MeV, hosts UV beamlines. A Short Pulse Facility provides intense, short x-ray pulses in the femtosecond domain. A 3 GeV LINAC will serve as injector for the storage rings in top-up mode, and as electron source for the short pulse beamline. The LINAC is constructed such that it can be upgraded to a seeded FEL facility in a future second phase.

The over-arching, emerging requirement for the development of the MAX IV facility's IT infrastructure is the necessity of an overall lean, resource-efficient design. MAX-lab has in the past been dependent on primarily cost-efficient solutions for IT development and maintenance, and will be so in the future. Still, as a user facility, MAX-lab endeavours to provide a competitive level of research quality for the MAX IV experiment users, and a reasonable perspective for continuous future improvements accompanying the long-term plans.

The composition of software technologies can provide us with useful toolkits to implement the desired features with acceptable quality properties. Hence we surveyed project related state-of-the-art software technologies.

DOMAIN SPECIFIC SOFTWARE

We have surveyed some domain-specific software frameworks in respect to the foreseeable requirements of the MAX IV IT infrastructure. The application domain includes the digital part of information processing, including data acquisition systems and the integration of, or exchange with, scientific analysis software. Our initial *framework-related requirements* are the availability of an integrative layer for desirable hardware platforms, operating systems and programming languages, a communication system with name resolution, application development support, an archive system, an alarm system, behaviour scripting support and administrative tools. We chose to look at open-source frameworks used at light sources: EPICS [3], TANGO [4], TINE [5] and DOOCS [6], the latter two being used in a multi-accelerator and linac-FEL environment. Realizing that all these frameworks would provide the framework-related requirements, we tried to discover technical arguments relating to our specific needs. In the following, we will outline some characteristics we perceived as notable.

For *application development* support for non-programmers, "jddd" offers a wide range of possibilities for user-built synoptic displays (geographical, functional, logical views) [7]. Enabling user customization for sophisticated applications is demonstrated by the ACOP+COMA concept [8]. Following an ambitious workbench approach, CSS [9] aims at an extendable, homogenous provision of various services. Beyond permanent *archiving services*, notable (quasi built-in) framework features are command archiving, temporary archiving [10], local archives, an event (post-mortem) archive infrastructure [5], and snapshot management tools, e.g. [10]. The communication systems' common *data types* are predefined [3, 4], and additionally customizable for TINE [5]. The overall throughput *performance* benefits from built-in multi-casting from each node [5], interesting especially where high data load sources have several data sinks. An application example is the advanced, feature-rich *video system* used at PITZ for beam studies [11]. System *scalability* for our needs is less influenced by network performance, but rather by

[*] here used synonymously to 'domain specific IT infrastructure'.

system structuring properties such as hierarchy concepts, object orientation, browsing services, redirection, gateways, aliasing, etc., as these impact practical complexity-handling. The *development and integration* of distributed control servers using varying technologies is supported by framework specific code processors (VDCT [12], POGO [4], Server Wizard [5], etc.). For hardware platform or field bus integration, generic interfaces exist (asyn [3], CDI [5], abstract classes [4], etc.). Driver availability can be sped up by using or customizing existing drivers, with the EPICS driver base [3] being the largest. *Interoperability* between protocols can be achieved by various methods [13]. *Commercial, domain-specific suppliers* support various frameworks reasonably [12, 14]. *Version* management and *automated deployment* can be realized with all frameworks. Systems for *safety* (accelerator, personnel, vacuum, equipment, etc.) and *synchronization* are recommended to use dedicated signal lines for hard real time tasks, and can be integrated for control, monitoring and archiving tasks reasonably well. For our foreseeable requirements, the dependability properties (availability, robustness, recoverability) of existing solutions appear as sufficient. *Usability* properties of the produced system (learnability, task efficiency, memorability, understandability, subjective satisfaction) are first of all application dependent, but can be constrained by the choice of application development toolkits. While malware and intrusion related *security* is more of a systems administration problem, write access restrictions help to prevent errors due to user mistakes or programming errors [5]. Issues related to beamline control can be addressed by e.g. synApps [3], GDA [15], Soleil's systems, BLISS [4]. Issues related to FEL routine operation (global feedback, synchronization, DAQ, FEL instrumentation and experiment equipment) are addressed within the DOOCS framework [6, 16]; others will follow (e.g. [17]).

Conclusions on Frameworks

It becomes clear that our possibilities will be limited rather by our resources than today's hard limits of the frameworks. In other words, we think that any framework would do us an excellent job, though in various respects. Further we conclude that choosing a control system framework for specific *technical* reasons, a *sufficiently detailed* requirements analysis with a *finalized validation* would be needed. Such can be the case e.g. for subsystems with known real-time demands. However, we do not see this possibility to be realistic for the integrative layer of the MAX IV project, given among other reasons the (intentional) vagueness for the FEL upgrade, and its derived requirements on the IT infrastructure. While a choice motivated by technical properties may be a sound for other facilities, it is currently not in our reach.

Realizing the general strength of the discussed options, we are increasingly shifting our focus to other aspects, risks and opportunities related to development of the systems and software engineering for MAX IV [18]. The development processes and their relations to various software technologies are outlined in the next section.

DEVELOPMENT PROCESS SUPPORT

Here, we describe the anticipated control system and software development for the early project stages. Further we describe design guidelines intended for cost reductions on the mid-term or long-term time scale.

We intend to establish a multidisciplinary control system group as a pool of specialists with appointed contact persons for technical groups or project specific communication. Project management tools are matter of further investigation, e.g. [19].

The MAX IV project challenges MAX-lab with a significant organizational growth, project complexity and successive *information management* issues. We investigate using a Product Document or Live-cycle Management System (PDM/PLM) for the electronic data organization, e.g. [20]. For IT systems development such can serve as a central document repository (specifications, user manuals, 'published' applications, CAD, measured data, etc.), providing document access, consistency and change management. Project specific "sandboxes" are desirable for restricted remote access by involved stakeholders (out-sourced projects, research groups, etc).

We are presently developing an information structure able to structure systems and software *requirements* [21] and *specifications* consistently, prioritized, traceable and in relation to the various stakeholders' perspectives. The requirements structure needs to be suitable for concurrent top-down, bottom-up and middle-out developments, reflecting starting points such as feature requests and use cases, equipment integration requests and service-centred approaches. Currently the Borland CaliberRM [22] software is used. The requirements database is intended to be maintained only by a limited number of system architects in the control system group. Word documents containing specifications can then be exported using templates.

We consider it advisable to start the implementation of a *control system simulator* at an early time within the MAX IV construction phase, before the installation of actual hardware systems. A control system simulator consists of control system framework components of all kinds (applications, services, local control servers, etc.), except that local control servers are internally either dummies or connected to a software-based dynamic physics model of the accelerator machinery. Physics modelling can be based on MATLAB, including the accelerator model, with all frameworks. The simulator development would be advantageous for staff training, software verification and validation, as it enables a more realistic, complex system environment.

Beyond systems *verification* of applications, services or control servers by test code, test scripts etc. in a standalone fashion, a control system simulator can enable testing complex communication situations, service functions and data processing of more realistic data. Errors and failures can be injected in controlled ways, and their propagation and treatment can be tested and practiced. The probability for delays during machine commissioning can be reduced.

In particular for accelerator control applications (or application toolkits) used for commissioning it is desirable to have the first *validation* iterations by users (operators, accelerator physicists) early on. Preferred means of validation are reviews of requirements documentation and architectural design documents by users. Succeeding iterations using a control system simulator allow to validate the application design, the functionality, and to elicitate new requirements. For the FEL upgrade, a control system simulator may be useful to minimize disturbances of the on-going operation.

A potential for cost reductions lies in *consequent standardization* to reduce the number of systems types, which the staff has to cope with, in order to reduce staff training, development and maintenance diversity. We consider to offer a generic, but feature-rich toolset to the beamlines, addressing common experimental or scientific requirements. The standards would further be applied to the accelerator domain. The standardization candidates:

- Hardware acquisition and building blocks: Only defined commercial-of-the-shelf IO modules are used in defined building blocks, incl. software interfaces.
- Libraries for control and data formats of similar beamline devices are standardized across beamlines, e.g. optics libraries, interpolation tables, etc.
- generic control servers (e.g. for x-ray mirrors, vacuum)
- a generic, feature-rich multi-axis scan system addressing problems specific to the facility's accelerators (e.g. handling top-up interruptions)
- a generic data acquisition and management system
- a high-level system behaviour scripting application, useable both for beamlines and accelerators
- GUI builders for non-programmer staff
- standard services for histories, system snapshots and configuration management
- standard building, versioning and deployment system

Problems with standardization guidelines could emerge from conflicts with an otherwise very desirable, creative laboratory culture, where individuals push smaller projects with great personal initiative, and thereby naturally choose hardware, software etc. of their personal preference. Similarly, this applies to outsourced subsystems (avoidance of solution specification). The best ways to compromise or convince in this respect is subject to discussion.

By providing users with software tools for building control applications and scripting physical behaviour [23] which are suitable (syntax, entity structure) for non-programmers, *user autonomy* can be enhanced, shifting workload from the control system staff to users on a medium- and long-term time scale. This also passes things to scientists or machine operators, who are better acquainted with their equipment and professional intentions. Finally, we have to consider changes in our user community, resulting in more visiting research groups who are less trained with our experimental facilities; hence, such tools have to enable good support for incorporating help and sanity tests.

CONCLUSION

For MAX IV, major design considerations are still in flow, and further contemplation based on recent perspective changes is needed to consolidate in a comprehensive program. We now consider project risks related to domain-specific software frameworks as relatively low, and expect them in other domains. The elaboration and validation of development guidelines for the development approach, standardization, user autonomy, etc., appears to be a prospective approach for technology decisions and to expose potential project risks which are probably higher than domain-technology related risks.

A great help in our on-going learning process has been a variety of contacts with many experts in the domain, who shared their knowledge, experiences and advice with MAX-lab. We wish to express our sincere gratitude and appreciation of these efforts, and hope for further opportunities to learn from the wealth of experience in the community.

REFERENCES

[1] www.maxlab.lu.se
[2] www.maxlab.lu.se/maxlab/max4/
[3] www.aps.anl.gov/epics/
[4] www.tango-controls.org/
[5] TINE as an accelerator control system at DESY. Bartkiewiecz, Duval. Meas. Sci. Technol. 18 2007. also: tine.desy.de
[6] doocs.desy.de; see: Status of the FLASH Free Electron Laser Control System. Rehlich. ICALEPCS 07
[7] ”jddd”: A Java DOOCS display for data display for the XFEL. Sombrowski et al. ICALEPCS 07
[8] The run-time customization of Java rich-clients with the COMA class. Bacher et al. ICALEPCS 07
[9] Control System Studio. Hatje et al. ICALEPCS 07
[10] Status of the Tango Archiving System. Pierre-Joseph et al. ICALEPCS 07
[11] Status of a versatile Video System at PITZ, DESY-2 and EMBL. Weisse et al. ICALEPCS 07
[12] www.cosylab.com/solutions/particle_accelerators
[13] The Babylonization of Control Systems Part II. Duval et al. ICALEPCS 03.
[14] Libera Brilliance, Software, on www.i-tech.si
[15] www.gda.ac.uk/
[16] The Data Acquisition System of the FLASH facility. Agababyan et al. ICALEPCS 2007.
[17] FERMI@Elettra. Conceptual Design Report.
[18] Systems Engineering. Coping with Complexity. Stevens et al. Prentice Hall 1998.
[19] Management System tailored to Research Institutes. Verstovsek et al. ICALEPCS 07.
[20] www.metataxa.com/icat.asp (lab. tailored tool)
[21] Software Requirements. Styles and Techniques. Soren Lauesen. Addison-Wesley 2007.
[22] www.borland.com/us/products/caliber/rm.html
[23] A Graphical Sequencer for the Soleil Beamline Acquisitions. Abeillé et al. ICALEPCS 07.