Development of Software for Instrumentation in Power and Research reactors and Accelerator based Research

Padmini S., M.P. Diwakar, P.M. Nair, Swati Gaur, S.S. Pande and C.K. Pithawa
Electronics Division, E& I Group

and

Nilesh Gohel
Research Reactor Maintenance Division, Reactor Group

Ms. S. Padmini is the recipient of the DAE Scientific & Technical Excellence Award for the year 2009

Abstract

The Electronics Division is mainly concerned in building systems for reactor instrumentation and nuclear physics experiments. Software forms a significant component of these systems. Applications for reactors have large overlapping functional requirements in addition to specific requirements. This has been captured into a reusable “Real Time Data Acquisition Framework.” This application framework provides a platform and imposes high-level architecture for building embedded systems. The variability in these systems is taken care of by configuration points. Using this approach, the following systems have been developed: Flux Mapping System for TAPP-3&4, Digital Recording System for AHWR Critical Facility, Radiation Data Acquisition System, Fire Alarm System and RRS Data Acquisition System for Dhruva. Nuclear data acquisition software has been designed to dynamically exploit the number of available processor cores, making it scalable to meet throughput requirements of individual experiments. This paper describes some of the software technologies used, the software development process and the major applications that have been developed in the last few years.

Introduction

Software is a significant component of the instrumentation projects that have been undertaken by the Electronics Division, for Power Plants / Research reactors and accelerator based research. Systems developed for nuclear reactors have to address concerns of high reliability, real time performance, safety and interoperability. Nuclear data acquisition systems have to be designed to be scalable, handle large volume of event data and filter meaningful data through sophisticated software triggers. These systems need to have the ability to exploit advances in multicore architecture, to meet throughput requirements and allow programmability to the user. In order to meet these challenges, some of the important decisions were concerned with the choice of technology to be used. Attempts were made to identify, evaluate, select, and absorb software innovations for improved performance. These include Modern Languages, Frameworks and software engineering practices. While adopting these technologies, a transition pattern was followed where these technologies are first adopted for nuclear physics applications and after a certain level of maturity is reached, they are applied for safety critical applications.

The paper is organized in three parts: Technologies that have been used to address the above challenges, applications that were developed using these technologies
and improvements made in the software development process.

**Real time Data Acquisition Framework**

Designing a distributed application, requires careful consideration of performance, scalability, reliability, security, maintainability, interoperability and extensibility. The concept of an architecture-based reuse has been used, which provides a common platform for a family of related applications. Variability between these applications has been analyzed and configuration points are provided, to take care of these variations. Handling the variant aspects is done by the Component configuration infrastructure, that supports dynamic, runtime (re) configurations and (re) deployments of objects. By reusing an appropriate architecture, not only development and testing effort is saved, but time to complete is reduced and architectural mistakes are avoided.

**Multicore and Task-based Programming**

It is a well-understood maxim in the technology industry, that software and hardware must evolve in parallel, and be well matched to achieve desired functionality and performance. Physical limits of semiconductor technology and improved manufacturing technologies, are driving processor architectures towards multi and many cores. Most desktop machines today have at least a dual-core microprocessor. Servers and advanced workstations are designed with quad-core and octal core microprocessors.

Traditional sequential code, does not take advantage of multiple cores, because the serial instructions run on only one of the available cores. Improving the throughput requires splitting algorithms into multiple threads, coordination among the different units of code, sharing information between them, and collecting the results. Threads created directly with native threading package are logical threads, that are mapped by the operating system onto the physical threads of the hardware. By using native threads directly, the developer becomes responsible for matching the parallelism available in the application, with the resources available in the hardware. Creating too few logical threads will result in underutilization of available hardware resources. Creating too many logical threads will cause the operating system to incur considerable overhead, as it must time-slice access to the hardware resources.

We have used a lightweight concurrency model using the Tasks Parallel Library (TPL) in .NET Framework that is based on actor model. This model could express the logical parallelism in applications with large number of tasks, which are scheduled by the runtime library on to its internal pool of worker threads. Usage of tasks made it possible to focus on the logical parallelism in the application, without worrying about managing the parallelism. In addition, since tasks are much lighter in weight than threads, parallelism could be expressed at a much finer granularity. This design is scalable on many core architectures.

**Modern Languages**

Multi-paradigm is the paradigm of the future. The level of abstraction of programming languages is increasing with features like garbage collection, type safety, threading, exception handling, extensible data types, and code security provided in the language itself. The three major trends for programming languages are declarative, concurrent and dynamic. Recent developments in industry have resulted in big influence on languages. Many functional languages like Erlang, Scala, Haskell, Clojure, F# etc. are becoming popular, as they make concurrency a lot simpler. There is no sharing of state and mutating of state in these languages. F#, a functional language with strong typing, very succinct, type inferencing, pattern matching, active patterns, async workflows, units and measures, has some imperative constructs. Tools are getting sophisticated in finding and exploiting fine-grained concurrency when they are presented with abstract, declarative and functional codes.
In the systems designed, safe high-level programming languages have been used to the greatest extent possible. These prevent critical errors while facilitating development and use of accurate and efficient software-development tools. C++, an object oriented language has been used, for embedded system development on QNX RTOS. A multi-paradigm programming language C# on the .NET platform is currently being used, for development of Operator Consoles for Dhruva control room. With C# 4.0, explicit functional language extensions such as lambda expressions, closures, continuations and type inference have been used. LINQ technology built into C# encouraged declarative programming style. Currently F# is being explored for use in application areas like scientific computations, UI etc.

**Software Development Process**

An iterative incremental model of development was adopted for our projects. Classic software development processes follow the waterfall lifecycle where development proceeds linearly from requirements analysis through design, code & unit testing and system testing, with limited feedback on the results of the previous phases. In this approach, there is a lot of emphasis on “the specification” which is required to be complete. In a waterfall approach, many problems are resolved during the system integration activity.

The main advantage of iterative incremental is to bring more predictability to the outcome and the schedule. It brings higher quality products and satisfies the real needs of end-users, as there is time to evolve requirements, design and implementation. In the iterative process, the software that is developed comes first. The software architecture is driven by early lifecycle decisions. Planning is mostly based on risks and unknowns, which have to be addressed right from the onset.

**Agile Methodology**

The practices of traditional software development processes are inadequate to control projects with complex technology and sophisticated requirements. We have adopted agile development practices, which resulted in the ability to manage changing priorities, increased productivity, improved team morale and enhanced software quality. The Agile process implements empirical process control through iterations, frequent releases of working, tested software, emergence of requirements and architecture, self-organization of multiple small teams and collaboration with users.

Working closely with the users, the agile team delivers version of the software as early and as often as possible. Constant feedback is obtained from the code that is delivered. Automation is used for continuous building and testing the project. In order to preserve the design, the code needs to be re-factored. Work progresses in iterations: small blocks of time (a week or so) where a set of features are identified and implemented.

Software development infrastructure for agile process has been set up with tools and procedures for configuration management and automated unit testing, profiling, code coverage & dynamic testing. The tool set also includes re-factoring tools, tools for automated build, test, release & deployment activities and acceptance testing tools.

**Applications**

**Flux Mapping System for TAPS 3 & 4**

The Flux Mapping System (FMS) of 540Mwe PHWR is a system, which is the first of its kind, used in Indian PHWRs. It is used to compute a detailed flux/power distribution of the reactor core using modal synthesis method. The
The main function of FMS is to monitor the in-core Vanadium self-powered neutron detector signals, located at different locations in the core, generate neutron flux profiles, 14 zonal powers and other related information, compute, and send zone power correction factors to the Reactor regulating system. FMS also performs two additional functions namely: display of detailed flux maps and computation of burn up of Co\textsuperscript{59} isotope and the build-up of Cobalt \textsuperscript{60} isotope.

The system is designed as a three-tier distributed fault-tolerant system with dual redundant Ethernet optical links between nodes. The three tiers are: Nine Input Scanning Processor (ISP) sub-nodes, dual redundant Flux Mapping Processors (FMP) nodes, and dual redundant Operator Console (OC) nodes. ISP nodes are located in the reactor building, receive signals from SPND detectors and send the scanned data through dual redundant fiber optic link, to the FMP nodes located in the control equipment room. Each of ISP and FMP are PC-architecture-based, rugged embedded nodes.

The nodes of FMS are interconnected through dual redundant fiber-optic links. Fiber optical communication eliminated EMI interference. This network architecture has resulted in drastic reduction of cabling from reactor building to control room. In order to achieve the deterministic behaviour in the Ethernet based network, master-slave protocol was used, which avoids the potential packet collision.

The embedded nodes (ISP, FMP) run on QNX RTOS to meet safety standards. QNX Neutrino is ideal for embedded real-time applications, as it scales to very small sizes and provides multitasking threads. It has priority driven pre-emptive scheduling, and fast context switching. QNX Neutrino achieves this through fundamental principles: Microkernel architecture and Message-based inter-process communication.

Each of the FMS nodes periodically performs self-test, to identify faulty components. Watchdog continuously monitors system integrity. The redundant FMP nodes change their roles to active (Master) and stand-by (Listener) role in alternate scan cycle. This ensures that all the software modules in each FMP node are active all the time. This helps in prompt detection of faults in the FMP nodes. Redundant network links are also used in the same way, for early detection of link problems. This resulted in achieving high availability of the system.

**PC Based Systems for Research Reactors (Dhruva and AHWR Critical Facility (CF))**

In Dhruva, a need for upgradation of some of the instrumentation was felt, in the data acquisition and processing systems, due to either obsolescence or for augmenting the facilities provided by the existing systems. Taking care of the retrofitting requirements of obsolete systems / instrumentation, the following systems were designed, developed and deployed at Dhruva: Radiation Data Acquisition System, RRS Data Acquisition System, Fire Alarm System and Operator Console for Alarm Annunciation System. Significant improvements that have been obtained using this approach are: completely redundant systems, convenient paperless large data storage, operator friendly interface and the facility to connect to other systems.

A digital recording system for AHWR CF was developed and installed, to record parameters from Reactor regulating system, Neutronic channel instrumentation & Process instrumentation and to monitor the actuation time of shutdown devices like Shut off rods and Moderator dump & Control valves. On-line estimate of critical height of moderator level based on inverse LCRM count could be computed which was very useful in first approach to criticality.

**Ethernet CAMAC Crate Controller**

CAMAC is currently the most widely used instrument standard for nuclear data acquisition and accelerator
control applications in accelerator labs in India. A new CAMAC controller (ECC) has been developed with an embedded processor and Ethernet as the communication medium to the host computer. Control applications need several physically distributed crates with regular scanning of all the parameters with control from a centralized PC. On the other hand, nuclear physics experiments need a high throughput with a large number of parameters. ECC is designed to achieve these objectives using single cycle mode, scan mode and list mode operations.

The ECC is designed with Single Board computer having PC architecture, which allows standard Ethernet services, such as TCP socket based communication protocol. The processor runs a version of real-time operating system QNX. The micro-kernel architecture of QNX allows for low memory footprint and gives deterministic performance. The multi-threading architecture permits the overlapping of the operations of data acquisition and delivery of data through the communication channel. This results in improved system performance even on a single processor system. Judicious use of multiple threads with lockless algorithms enables the system to take advantage of multi-core support in QNX to provide a scalable solution.

**Conclusion**

It has been observed, that when software development takes into consideration advances in software technology, it results in radically new and powerful solutions. We have been able to embrace promising technologies, modern programming languages and good software engineering practices. Framework based approach not only resulted in speeding up the system development, but also cut down the time and effort required for testing and maintenance of these systems. Agile adoption improved productivity, reduced project risk, and enhanced software quality through collaboration, test-driven design and continuous integration.

**Acknowledgement**

The authors would like to acknowledge the encouragement and support provided by Shri M.D. Ghodgaonkar and Shri P. K. Mukhopadhyay. We acknowledge the contribution of Reactor Instrumentation Section, ED. in the design and development of hardware for FMS. We acknowledge the contribution of Data Processing Group, ED. in the design and development of hardware for CAMAC Controller. We are also thankful to the user community for their collaboration, which helped in making the transition to agile a success. We are thankful to Shri Kevin Fernandes, ED for shaping the UI and additional support provided in commissioning of the systems.