Paper presented on the EHPG meeting of the OECD Halden Reactor Project, Storefjell, Norway, 12-15, March 2007.

Investigating the case of Open Source applications within nuclear power

Olli Ventä, Björn Wahlström
Technical Research Centre of Finland (VTT)
+358 20 722111, olli.venta@vtt.fi, bjorn.wahlstrom@vtt.fi

Abstract

Open Source has since the creation of the first version of the Linux operating system become a realistic alternative also for commercial applications. Open Source software has so far not been used within nuclear power at least to a level where serious application notes would be found from a web-search. However, going through some of the requirements that are placed for example on I&C for nuclear power plants there seem to be advantages if at least some parts of the software components would be based on Open Source technology. Some of these advantages include, but are not restricted to, a possibility to assess the likelihood of software failures and an easier upgrade to new generations of hardware. The paper addresses benefits of using Open Source based applications in different functions within nuclear power plants. The intention is to open up a broader discussion of strategies within the nuclear community in the development of I&C and other computer based applications for present and future nuclear power plants.

1. INTRODUCTION

Open Source[a] software has not yet been applied within the nuclear industry to a notable extent. The reasons are many, such as the requirement to use proven technology, the perceived hacker status of Open Source and the need to license all safety related systems. The situation has however changed and there are today many reasons to believe that at least some computer based application at the nuclear power plants could benefit from Open Source technology. The rhetorical question regarding the use of proprietary or Open Source software in safety related computer system has been posed, but no real discussion has so far emerged [1].

Taking a broader look on software development it can be noted that the complexity of computer systems has grown tremendously over the last decade. Typical software can include millions lines of code and require development resources that are in excess of many thousands of person-years. In view of these figures, there is no possibility for any single person or even group of persons to comprehend all details of these systems, which means that efficient models of co-operation in the development projects have to be created. The traditional model, which has been used within the global companies, is the top-down model of large projects that are managed using scientific management methods. The Open Source model, which has demonstrated its viability, can be seen as a bottom-up model, where application needs steer the process and committed persons serve a loosely patched development organisation. The Open Source model of development has been seen as a completely new model of organising innovation processes [2].

A closer look on the Open Source projects reveals that their participants are users of the software that is developed. This fact seems to explain at least a part of the success, because this provides a remedy to one of the problems that has been seen in the development of software, i.e. the software developers do not know what the users would like to have. A common counter-argument to the Open Source model of development is that free riders would get access to the innovations without any benefit for their developers. This argument seems to be based on a rather utilitarian model of people, which may not be completely realistic. For

[a] The term Open Source is in this connection used in a general meaning of software that is distributed under a user license, which gives free access to the source code (Public Domain, Freeware, etc.).

example, research in the economics and sociology of Open Source projects has revealed that the developers typically are driven by many other motives [3] of which personal gain seems to be down on the list. Another finding is that the code architecture and its option value actually may mitigate free riding [4]. The Free Encyclopedia is a similar project in another area [5].

This paper investigates possible benefits of using Open Source software for computer applications within the nuclear industry. Concrete suggestions for further work within software development at the Halden Reactor Project are developed. These suggestions can be applied more generally also for nuclear power plants.

2. THE OPEN SOURCE MOVEMENT

The history of the Open Source movement can be found at many places on the Internet and will therefore not be repeated in this connection [6]. To summarise the present state of the development efforts, the Open source movement has grown from a hobbyist activity in the beginning of the 1990ies to a serious business, where today thousands of servers all over the world operate with Open Source software. To understand this success one has to go beyond the technical issues and consider also the driving motives of the developers and the users of Open Source software [7]. One indication of the seriousness of Open Source within the software industry is a recent special issue on Open Source software in the journal Management Science [8].

When assessing future development of Open Source software, the prospects appear to be bright. An ever increasing area of commercial applications is built on platforms such as Linux (operating system), MySQL (database) and Eclipse (software development environment). The driving force of this trend is quality, maintainability and reliability or shortly the dependability of the software. These objectives are also common themes in the ICT-programme of the 7th Framework Programme funded by the European Union [9]. The ICT-programme has also been given a focus on global standards, which covers the whole spectrum of software products from the hardware up to the upper layers of applications.

Considering the rationale behind Open Source software development the first thing one has to note is that the products developed are a public good. They can be copied, shared and distributed anywhere all over the world at a transaction cost that is practically zero. The initiative of developing some specific software under the Open Source license typically emerges from a small group of individuals that wants to develop a software product to meet their own needs. Presently there are in excess of 80000 active projects in the world [10]. The projects have over the years developed co-operative practices, where a varying number of core individuals take the largest load of development to get testing support from a very much larger group of people.

3. COMPUTERS USED IN THE NUCLEAR INDUSTRY

The nuclear industry has been using computers for different purposes from the very beginning. Already in the early 1970ies computers were used in control and instrumentation and in many analysing instruments [11]. Process computers as a major information source for the control room operators became a standard for the nuclear power plants already in the early 1980ies. Today the development of computer based system has reached a point, where there are no real alternatives to digital and software based systems for control and instrumentation [12].

There have been many hurdles in transferring from analogue to digital technology. Most importantly the inherent features of the digital technology have not always been comprehended, with the consequences of time delays and exceeded project costs. Sometimes diverging views between utilities and regulators on what should be considered as a sufficient evidence of safety have been contributing to the difficulties. Especially the question of the extent of diversity needed as a protection against common cause failures has shown to be difficult.

There are many practical challenges in applying computers and software in safety relevant systems at the nuclear power plants. One challenge is connected to the short product life-cycles within computers and software, which can be contrasted with plant life times that are an order of magnitude longer. Another challenge is connected to a low number of nuclear projects, which have made it is difficult to find vendors committing themselves to the area. A third challenge is more generally connected to the use of emerging technologies, which for example may introduce new security problems.

4. DEVELOPING SOFTWARE FOR SAFETY CRITICAL APPLICATIONS

The construction of safety in the nuclear industry relies on two important principles, *defence in depth* and the protection against threats using *independent safety barriers*. The application of these principles leads to the introduction of *redundancy*, *diversity* and *separation*, which ensure that no single failure will threaten safety. Unfortunately these design principles are difficult to apply for digital software based systems, because the behaviour of a computer is essentially deterministic, which means that software errors always have the possibility to cause failures of the systems.

Methods and tools to write high quality software have emerged over the years and have been applied in the nuclear industry with success also for safety applications. One very commonly used method is the application of the so called *waterfall model* for software development, which divides a software project into phases of requirements development, design, coding, tests, integration, installation and commissioning [13]. Between the phases a thorough *verification* and *validation* is used to ensure that software errors are detected and corrected.

In other design applications a promising technology has emerged, which is very similar to the Open Source paradigm. This concept has been called *design patterns*, which loosely can be defined as a good solution to a specific problem. This concept has its base within *object oriented programming*, which provides a structured approach to software design. The design patterns provide problem – solution pairs that can be reused and they are exchanged freely to build up repositories in various areas. One example is an emerging community of people that are interested in the application of design patterns for safety critical software.

The difficulty with the process of producing safety critical software is the amount of work, which is needed before a good confidence can be reached that the system can be considered safe. This difficulty is multiplied if rapid technical development forces the design to be recreated always when earlier hardware and software platforms have become obsolete. For safety critical system it would, at least in principle, be necessary to give quantitative reliability estimates for digital software based system to enable them to be included in a probabilistic safety assessment.

5. SOFTWARE DEVELOPMENT WITHIN THE HALDEN REACTOR PROJECT

The Halden Reactor project has been involved in software development for the nuclear industry since the late 1970ies. In the beginning this development was focused on the development of computerised control rooms, but was later diversified to include all sorts of operator support systems as well as methods and tools for the development of high quality software.

The activity directed towards software quality and reliability was initiated in the late 1970ies. This activity has lately been focused on the specifications elicitation phase of software development. Present activities at the Halden Reactor Project also include software for the experimental facilities and activities, which are used to validate the applicability of proposed technical solutions.

The software activities within the Halden Reactor Project have encompassed a large span of development projects ranging from simple specialised modules to large generic systems. The developed software has mainly been used in-house, but some of the signatory organisations have expressed interest in getting experimental versions of the developed software for their own purposes. To our knowledge the possibility of developing software under some Open Source license has not been discussed within the Halden Reactor Project.

6. THE POTENTIAL OF OPEN SOURCE IN THE NUCLEAR FIELD

From a technical point of view there should not be any restrictions, at least in principle, to use Open Source software for selected applications in the nuclear field. Such a transfer would most likely involve a rethinking of many work practices that are in use today, but in our view there are many arguments in favour of such a solution. The most important argument is that use of Open Source would make the licensing process in various phases of software based systems much easier. These arguments are expanded below.

6.1 General software quality based arguments

The first argument is connected to the potential of reaching a higher quality using the Open source development paradigm. Already a superficial assessment makes it believable that more resources could be allocated both to development and testing if selected software modules are developed under an Open Source license. It is also likely that the people involved in such a development would be highly motivated in their efforts.

One important benefit of the use of Open Source software is the availability of the source code. In the licensing process one may for example apply different manual and computerised inspection tools for the source code. Another issue, which may ease the licensing process, is also that Open Source software tends to be more modular and versatile than proprietary software [14]. Open Source based applications can also benefit from increased simplicity, because non-relevant modules can simply be taken off the code.

Perhaps the largest benefit of Open Software will emerge from a development process that is governed by application needs. Interfaces between modules and between different vendors would be built on open standards, which would imply a large potential for re-usability, even in the case that a specific vendor or product would disappear from the market. The move to new versions is also likely to be easier and possible to carry out in a more carefully planned process.

6.2 An approach towards common cause failures

The Open Source software carries due to its larger inspectability an opportunity to make more accurate assessments of the likelihood of common cause failures. This argument builds on the recognition that a software error has to be triggered by the path of execution to cause a failure. Consequently a common cause failure implies that the same software error is hit at the same time in all applicable redundancies.

A common architecture of a digital software based system is to separate between system and application software. The system software is reused in many different servers and projects, which makes it easier to test thoroughly and to collect operational experience. The application software is written using standardised module calls to make each module as simple as possible. By the selection of suitable modules it is possible to design functions with a small number of module calls, to make the application software easy to inspect.

The argument for not requiring diversity in some function can then be built through the following chain of arguments:

- The path executed by the system software can be made very stable through deliberate design and thorough testing.
- The modules used in the application software can be made simple to ensure that inspections and tests can verify their correctness with a high degree of certainty.
- The execution paths of the modules can be tested separately to ensure a good coverage of the tests.
- The physical conditions at the signal level (analogue noise, time fluctuation in sampling) will make it unlikely that two redundancies will go through exactly the same execution path,
- The introduction of diversity has its own drawbacks, which may increase the susceptibility for failures introduced by maintenance or false triggering of protective actions.

6.3 Computer security

The importance of computer security has got an increased recognition over the last couple of years. So far the nuclear industry has not experienced any serious computer security incident. However, the event in January 2003 when the Slammer worm brought down computers at the Davis Bessie nuclear power plant gave a general warning to the nuclear industry.

Open Source as such does not give any benefit over proprietary software, but the possibility to analyse and tailor the software for specific needs gives a certain benefit. The software at nuclear power plants will be divided into security zones, between which strict rules of transfer will apply [15]. The software based systems that are the most important for plant safety will according to this principle be strictly isolated from the outside world with only one-directional transfer of information to the control room and archiving computers. The verification that this information transfer is truly one-directional can be very hard if the transfer protocol and the software are proprietary.

For future software based systems at the nuclear power plants there is a mounting pressure to include some outside connectivity. The primary need for such connectivity is one-directional for various diagnosing purposes, but also the updating of software at the nuclear power plants would benefit if it could be done remotely. If one would like to respond to these needs it would be necessary to build specialised solutions and to validate them thoroughly.

6.4 Risk informed decision making

Risk informed decision making in the licensing process has larger possibilities to be applied when the source code is available. It may for example on probabilistic grounds be possible to relax a deterministic safety requirement if it can be shown that a specific chain of event is very unlikely. Similarly it may on deterministic grounds, such as the architecture and structure of the software, be possible to claim that certain failure sequences are not likely. One may for example argue that pointer overflows are not likely to occur if the processor is reset between each timing cycle.

The collection of data from the development process of Open Source software seems also to be quite straightforward. The history of module changes is available to a large degree of detail, which makes it possible to get for example a record of software error development. Such data can then be used to build Bayesian belief networks to decide on the sufficiency of presented safety evidence.

6.5 Implementing and licensing software at nuclear installations

A successful implementing and licensing of software based systems for the nuclear power plants relies on a thorough understanding of the whole life-cycle of the software in consideration. In the case that Open Source applications are used it should be possible to support all phases of the life-cycle by Open Source products.

This means for instant that requirements specifications, compilers, linkers, loaders, application code generators and version management systems would be supported by Open Source based tools.

In a larger time frame, the most important argument for using Open Source products for nuclear power plants is the possibility to reuse application designs over the whole life cycle of the plant. If for example a plant is targeted for an operational life of sixty years it seems likely, at least with the present development pace of computer products, that it has to go through at least two large modernisations. Two large modernisations may perhaps be traded with one major and four minor modernisations depending on the situation, but there is still a large incentive to reuse as much as possible of the software that already has been licensed. With Open Source it would be easier, because not all of the verification and validation has to be redone and the plant would not be forced to depend on companies that implemented the original software.

7. AN ACTION PLAN

We believe that there are many arguments in favour of establishing and Open Source policy for software produced at the Halden Reactor Project. The arguments given below are based on a rapid walk through of work at the Halden Reactor Project and literature from the Open Source field. Before a decision is taken, it would be important that these arguments are expanded and weighted carefully. The sections below introduce some of the major arguments and a possible route forward.

7.1 Benefits of Open Source to the Halden Reactor Project

The perhaps most important benefit for the Halden Reactor Project of an Open Source software production policy, is the increased quality of software produced. The increased quality is expected to emerge through the need to expose early plans and initial codes to outsiders and through their commenting and testing efforts. Open Source is likely to make it easier for Signatory organisations to apply developed software for experimenting and testing purposes. Targeting software development on Open Source applications, it would be easier to dig deeper into difficult research questions connected to risk based decision making, software reliability and common cause failures. One could for example envisage the need for various software tools to support both the development and the licensing processes. Last but not least, an Open Source software development policy can, properly managed, generate new ideas for the interaction between research and the more practically oriented development activities at the Halden Reactor Project.

A summary of these arguments would then be that an Open Source policy is expected to give:

- Better quality of software products developed by the Halden Reactor Project.
- An efficient interaction between research work and practically oriented development.
- New areas for research in the software implementing and licensing processes.
- New forms of co-operation between the Signatories and the Halden Reactor Project.
- Better feedback from application fields and better possibilities for learning for the staff participating in software development projects.
- New possibilities for networking within the software community.
- Easier application of the developed software for real applications in the nuclear field.

7.2 A small exploratory project

It is proposed that the Halden Reactor Project starts a small exploratory project to investigate the feasibility of adopting an Open Source policy for software developed in-house. Such a project could for example investigate the following issues:

- Of the software that has been produced over the years, which would have profited of being developed under an Open Source license?

- How would the Signatories view a change in policy to produce software under an Open Source license?
- Would there be additional costs involved in selecting the Open Source path for producing new software?

The most suitable way to proceed is to present the project report at a suitable Halden Programme Group meeting to be discussed and assessed. If the decision is favourable a recommendation to the Halden Board would be given and this policy written into the next three year programme.

7.3 Strategic considerations

In a larger time frame, if the Open Source path would be considered appropriate for the Halden Reactor Project, it would be important to ponder a few strategic questions. Perhaps the most important is to select the software development projects, which would give the largest impact. In this selection the following criteria may be of help:

- The problem exists in the nuclear community and it has not yet found a good solution.
- Similar problems exist in other industries (process industry, off-shore, transportation, etc.).
- The problem has no clear organisational owner with whom a competition would be expected.
- There is a possibility to use research findings to take a major step forward for a solution of the selected problem.

In developing future strategies it would also be important to create appropriate work practices that make it easy for agents within the software community to assess proposed projects with respect to characteristics such as:

- What is the viability of the proposed project?
- Has the core group members the necessary skills (technical, administrative) to make it likely that the project will succeed?
- What are the likely rewards (influencing, learning, acclamation) for people taking part in the project?

8. CONCLUSIONS

It is clear that a commitment to Open Source implies a change of present thinking both for the nuclear field and within the Halden Reactor project. We do not argue that this transfer would be free of problems, but that such a transfer in the long run would be beneficial. Looking towards the future it is very clear that standards applied on a world wide scale will be a crucial component in avoiding unnecessary conversion activities and in reusing existing software. The emergence of open standards would be a benefit for everybody.

It is important that research organisations always reconsider policies, approaches and practices. The Open Source model of co-operation has demonstrated its efficiency for application oriented software. A large part of the activities at the Halden Reactor Project is directed towards the production of new and innovative software and therefore it would therefore be important that the Open Source model is assessed thoroughly. We actually claim that adopting an Open Source policy would provide for the Halden Reactor Project an opportunity to become a kind of knowledge broker between the nuclear and the software development communities [16]. We also believe that the risks by adopting such a policy are small as compared with the benefits that could be gained.

We actually argue that both the nuclear field and the Halden Reactor Project would be able to join forces for a mutual benefit in promoting Open Source solutions. We even argue that the nuclear field has a history of sharing knowledge, which is very similar to the Open Source movement, but that this good practice almost has been forgotten in the harsher business climate of today.

This paper has made a plea to the Halden Reactor Project to start a small exploratory project to investigate the feasibility of the Open Source model for in-house software production. If this model is considered feasible the Open Source policy should be written into the plans for the next three year programme.

9. REFERENCES

[1] Björn Wahlström, Risk assessment and safety engineering; applications for computer systems, SAFECOMP 2005, the 24th International Conference on Computer Safety, Reliability and Security, Fredrikstad, Norway, 2005.

[2] Eric von Hippel, Georg von Krogh, Open Source software and the "private-collective" innovation model: Issues for organization science, *Organization Science*, Vol.14, No.2, pp.209-223, 2003.

[3] Sonali K. Shah, Motivation, governance, and the viability of hybrid forms in Open Source software development, *Management Science*, Vol.52, No.7, pp.1000-1014, 2006.

[4] Charles Y. Baldwin, Kim B. Clark, The architecture of participation: Does code architecture mitigate free riding in the Open Source development model, *Management Science*, Vol.52, No.7, pp.1116-1127, 2006.

[5] http://www.wikipedia.org/.

[6] http://www.opensource.org.

[7] Steven Weber, *The Success of Open Source*, Harvard University Press, 2004.

[8] Georg von Krogh, Eric von Hippel, The promise of research on Open Source software, *Management Science*, Vol.52, No.7, pp.975-983, 2006.

[9] http://cordis.europa.eu/fp7/ict/.

[10] http://sourceforge.net/.

[11] IAEA, Nuclear power plant control and instrumentation, STI/PUB/301, 1972.

[12] IAEA, Managing Modernization of Nuclear Power Plant Instrumentation and Control Systems, TECDOC-1389, 2004.

[13] IAEA, Verification and Validation of Software Related to Nuclear Power Plant Instrumentation and Control, TRS-384, 1999.

[14] Alan MacCormack, John Rusnak, Carliss Y. Baldwin, Exploring the structure of complex software designs: An empirical study of Open Source and proprietary code, *Management Science*, Vol.52, No.7, pp. 1015-1030, 2006.

[15] IAEA (2004), Guidance on the security of computer systems at nuclear facilities, unpublished manuscript.

[16] Sulayman Sowe, Ioannis Stamelos, Lefteris Angelis, Identifying knowledge brokers that yield software engineering knowledge in OSS projects, *Information and Software Technology*, Vol.48 pp.1025–1033, 2006.