Domain-based Simulation Modelling to Enable Continuous Testing for Software Development in the Chemical Industry

Adam Ziolkowski[1] and Joost Noppen[2]

1 **School of Computing Sciences**
University of East Anglia
Norwich, United Kingdom a.ziolkowski@uea.ac.uk
2 **School of Computing Sciences**
University of East Anglia
Norwich, United Kingdom j.noppen@uea.ac.uk

Abstract

It is widely considered that the adoption of iterative software engineering methodologies and in particular continuous testing helps ensure high quality software and reduce bugs. The successful application of continuous testing however rests on the assumptions that testing is cheap, fast and easily repeatable. Software development for control systems in the chemical production domain generally cannot satisfy that constraint as evaluating the correctness of a recipe program requires its execution on a live production environment which can take multiple days to complete, usually comes at great expense in raw materials and can sometimes create a reliance on safety systems to manage risk. As a result testing in the chemical domain becomes a bottleneck that prevents true iterative cycles taking place. This in turn leads to a linear waterfall-like process with all its inherent problems and limitations.

To help resolve this problem, we propose a generic simulation framework, based on a domain model of core components of chemical productions plants. This simulation can be used in place of the live plant during a first phase of testing. Only once an engineer is satisfied that the software is performing as expected on the simulation, will that live plant hardware need to be involved. This will help greatly in reducing the bottleneck in the testing phase by allowing this to be quick and automated while reducing the risk and cost involved.

1998 ACM Subject Classification B.3.3 Performance Analysis and Design Aids

Keywords and phrases Continuous Testing, Industrial Automation, Domain Modelling, Simulation, Testing

1 Introduction and Problem Statement

One of the major cornerstones of modern day software engineering is the use of iterative software methodologies incorporating continuous testing. The ability to have short cycles that allow for continuous feedback on development has lead to software systems with less bugs, lowered costs and more predictable delivery [8]. Of particular importance in this context is the use of automated testing practices which effectively allows this continuous testing with minimal impact from a time and monetary point of view after the initial testing rules and conditions have been defined.

However the effective application of continuous testing during software development assumes the ability to perform, execute and evaluate test scenarios at minimal costs, a condition that cannot be fulfilled in some industrial settings such as the aviation, chemical production or nuclear energy domains. In these instances, testing is often challenging and


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time consuming involving analogue and digital hardware paired with software and often has further safety implications. While this can be considered a broad problem, this paper will focus on introducing continuous testing to the chemical production industry.

Chemical production plants are typically composed of a complex array of analogue and digital computerised systems that interact with each other to control the production process of chemicals. As shown in Figure 1, these plants typically include a form of low level software, running on a real time processor and directly influencing the plant hardware. These systems can be thought of as a combination of plant elements such as container vessels, actuators for mixing chemicals, temperature and pressure sensors, etc. This is paired with one or more layers of higher level control software which can include human machine interfaces, allowing operators insight and control of the production processes.

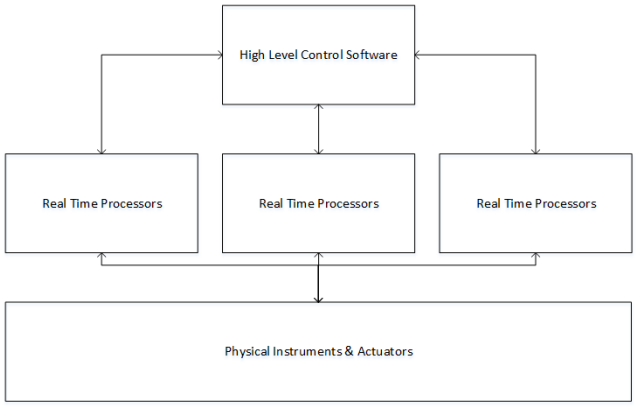

Figure 1 An abstract overview of a typical chemical plant architecture.

In modern chemical plants the production process of a chemical is defined using a so-called recipe program, consisting of varying steps, such as moving a chemical into a vessel followed by raising its temperature as shown in Figure 2. This is defined in a software development environment tailored to the plant configuration and control infrastructure. The development of a recipe program fundamentally follows the traditional phases of software development and many of its best practice principles can be observed.

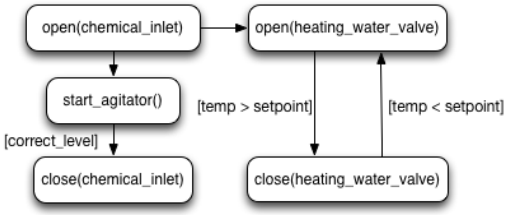

Figure 2 An representation of a recipie program.

However a divergence can be observed between iterative development practice and the development of recipe programs in testing. As recipe programs will be executed on plant control systems, currently the only way to assess their suitability and correctness is to perform live tests on the plant using actual chemicals. The chemical process and work flow then need to be observed and evaluated, after which the results can be fed back into the recipe program development cycle. Such live tests typically take multiple days to complete, during which

time the plant cannot be used for actual production. In addition raw materials are used as part of testing driving up the cost even further. Such testing regimes are thus extremely expensive, require downtime to the plant and if unsuccessful can cause further downtime, a loss of product or unnecessary reliance on safety systems to mitigate risk. These issues are also noted by Ljungkrantz et al. in their work [9].

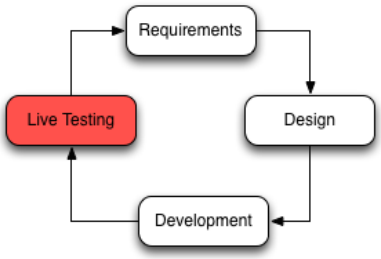

Figure 3 Live testing as a bottleneck in the recipe program development process.

The increased cost and time implications of testing and optimising the software on the live plant can be seen as creating a direct bottleneck in the adoption of iterative processes, as shown in Figure 3. This bottleneck encourages engineers to often adopt a rather more linear process, where an attempt is made at implementing all needed functionality before live testing is performed on a plant. The development process of recipe programs therefore more closely resembles that of the waterfall method rather than embracing the principles of iterative development methodologies. The fundamental reason for this development practice occurring is the inability to evaluate recipe programs in a fast and cost-effective manner. An accurate simulation of the behaviour of the plant and its control software would allow for continuous testing and feedback at a much lower cost, however to this day this has not been achieved. Due to the variety of configurations that can exist in production plants and their pairing with control systems from a range of vendors, a bespoke simulation would need to be created for every recipe program, thereby negating the cost gain from more in depth testing.

In this article we propose to address this problem by defining a simulation framework that is vendor-independent and can be configured easily for specific production plant configurations. Our proposed approach leverages the principles of domain modelling in software engineering [11] to capture the core building blocks of production plants and enable the simulation of their behaviour, given a recipe program, without involving the physical plant. The reduction in cost and time will enable faster development cycles and continuous testing within the chemical domain. The remainder of this paper is organised as follows: In Section 2 we present our proposed approach. Section 3 highlights our current results as well as our research method and in Section 4 we cover related work. Section 5 concludes.

2 Proposed Approach

Continuous testing is often not performed during the engineering of software for control systems due to a lack of a flexible testing solution without the need for a physical plant and real chemicals (hereafter referred to as off-line). This often leads to a reliance on live plant testing which can be extremely expensive and long running, often spanning multiple days. This research aims to explore if it is feasible to propose and implement an alternative testing approach, which would aid in reducing this bottleneck and thus enabling an easier and deeper adoption of continuous and automated processes.

This efficient off-line testing regime, where tests can be run faster and at less expense, will need to account for the actual behaviour of a plant. As such it is proposed to introduce a plant simulation framework to account for this behaviour. Simulation tools, such as SimSci Pro/II [13], are already available to assist with the design and engineering of chemical product facilities, however these are designed to perform studies of distinct plant areas, rather than modelling the plant software as a whole.

By performing off-line testing, a more iterative development methodology, such as the one represented in Figure 4 could be adopted. It is important to note that by using a simulation such off-line testing can be performed at a far greater speed that live testing. This allows for engineers to perform continuous off-line testing and tuning of complicated parameters throughout development, only moving to more costly and time consuming step of on plant testing once the software has been stabilised within the simulation environment.

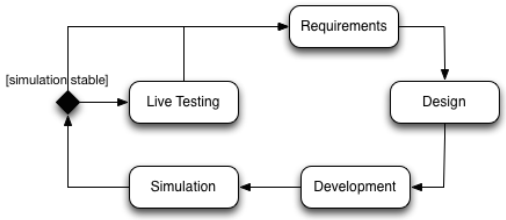

Figure 4 The newly proposed testing regime, using a domain model based simulation.

The fundamental challenge that has to be overcome in constructing such a simulation, is that each chemical plant has a different set of equipment, including but not restricted to values, vessels, measurement instruments and actuators combined with differing software and hardware control system architectures provided by different vendors. To overcome this, a generic mechanism needs to be created, allowing every single plant configuration paired with every kind of control infrastructure to be accounted for.

We therefore propose to create a domain model for the chemical industry that allows modelling the different configurations and behaviours of both the plant equipment and processes. This domain model, after having been instantiated with details of a plant, will become an integral part of the simulation, as shown in Figure 5 a & 5 b.

To enable the instantiated simulation to be used in the course of testing control software, the simulation will be executed using the control software (Figure 5 c) that is run on the live control system. While the majority of languages used to write control software are standardised by IEC 61131-3 [2], there remain implementation specific details specified by the vendor of the control system. It is proposed that for the purposes of simulation, an intermediate language is used to execute the simulation. A translation mechanism (Figure 5 d) will then be used to convert source code from a specific vendors implementation to this language. Initial investigations suggest that XML interchange methods, proposed by the PLCopen organisation [10], may prove helpful in doing this and merit further examination.

In addition to the execution instructions for the simulation, some other inputs need to be defined by the engineer to allow the software to be properly executed and tested. One of these is a set of ground truth values and rules for the simulation (Figure 5 e). These are very similar to the assert statements used by testing frameworks such as JUnit [6]. These will be defined using the expected set points and conditions that correspond to the correct operation of the plant. The starting state of the simulation (Figure 5 f) also needs to be introduced. This could for example define the starting level of a vessel or the raw materials present in the system. Using the provided information, the simulation engine will be able

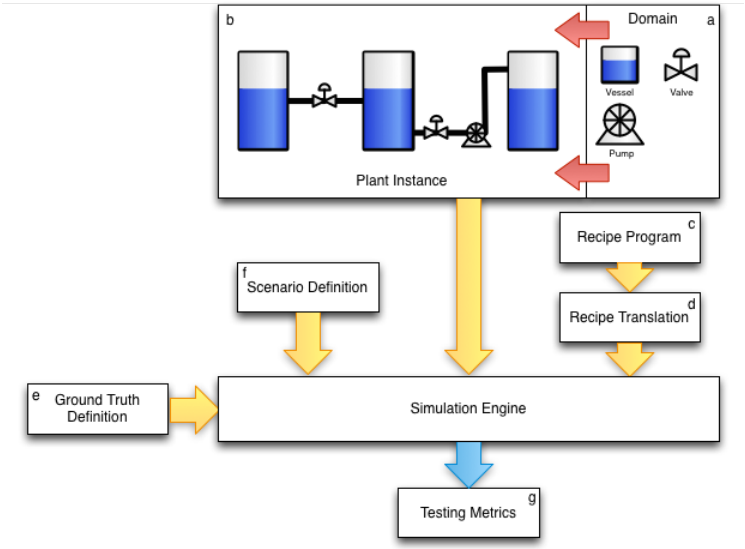

Figure 5 Simulation architecture used for off-line testing.

to detect behaviour that is outside of the design parameters and in turn produce testing metrics for the code base (Figure 5 g).

We hypothesize that by using the proposed simulation approach, integrated into a suitable workflow such as the one proposed in Figure 5, a more iterative approach such as the one shown in Figure 4 could be adopted. This approach promotes frequent testing, with the simulation allowing for risk free and low cost testing and giving an initial platform to tune complex system parameters. In addition, the basing of the simulation on a domain model would considerably reduce the effort required to produce the simulation model. This could be applied both for the development of software for a new plant, modifications to existing software or to explore hardware changes.

3 Current Results and Research Agenda

Our current work has involved performing an in depth study of the chemical domain. This has involved investigating associated software and control system architectures and has comprised diverse plants at our industrial partner. This has allowed us to identify the elements that are common between these systems and has given us the initial basis for a chemical domain model. This initial work has additionally enabled us to assess the feasibility of being able to make this domain model exhaustive.

A simplified overview of our domain model can seen in Figure 6. This shows various elements that are comprised in the chemical domain such as vessels, pipes, actuators and various sensors. In addition to these common plant and control system elements, it is important to note that we believe it useful for the model to also consider control system elements such as the input/output interfaces as this gives us further information for use in the simulation. In addition to these structural elements, further behavioural models will need to be introduced describing their functionality.

The future research efforts are envisaged to focus on the following areas:

- With the goal of creating a first instance of a simulation, a subset of a real world chemical plant will be chosen and a simulator constructed on top of the chemical domain model

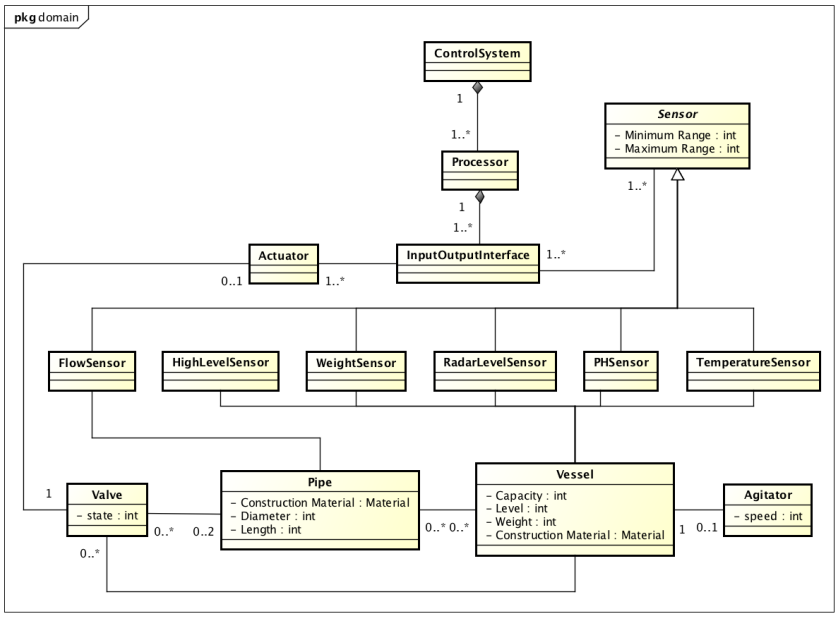

Figure 6 Abstract overview of the chemical production industry domain model.

elements.

- To establish the accuracy and relevance of current and future versions of our simulation, a number of recipe programs that contain known behaviour, both wanted and unwanted, will be defined. These will be divided into two sets, a training set against which the simulation will be developed and an evaluation set that will be used to judge the performance of the simulation. These recipe programs will be executed against the instantiated simulation and the results compared to those expected.
- The observed behaviour and outputs of our simulation will be compared to the actual behaviour recorded by our industrial partner. The decades of output data relating to plant state that our industrial partner has made available will also be used to evaluate the performance and effectiveness of the simulation framework.
- The simulations will be repeatedly instantiated with increasingly advanced plant processes and configurations. This will allow for them to be extended beyond the scope of our industrial partner, to cover the entire chemical domain. With every extension of the domain model we also plan to evaluate the accuracy of the simulation by comparing it's behaviour and outputs to the historical data available from our industrial partner.
- Our work will result in the creation of a tool to test plant software against an instantiated simulation model. An initial evaluation will establish the accuracy against wanted and unwanted behaviour, involving both experts at our industrial partner, in addition to the comparison of outputs against a large quantity of data from their production plant.
- We intend to perform a further in-depth study to generalise our findings within the wider chemical domain and to help evaluate the effect that such a domain based simulation framework has on the software engineering process. We intend to do this through observations and interviews with software engineers developing chemical plant recipes. In particular we aim to see whether what effect the introduction of a domain model for simulation has on the efficiency and accuracy of the engineering process.

4 Related Work

In existing work, there have already been efforts to attempt to analyse the software engineering methodologies used, and the application of iterative so principles to industrial automation applications such as those that we have been focusing on in the chemical industry.

An analysis of the methodology used in the development of industrial automation software was performed by Dubey [4] who argues that the development processes of such software is shown to proceed according to four phases. That representation shows progress is made in a linear fashion through the phases and Dubey concludes that there is a strong coupling between the phases resulting in an inability for changes to be introduced once the requirements have been finalised. Our hypothesis is that one of the contributory reasons that Dubey observed a linear process being followed was the bottleneck in testing software we described above.

Brusaferri et al. [1] explain that while the development of control software is not performed on the live plant, testing of such software is often carried out on the actual plant. They note that this is not desirable as it can lead to a substantial period of time before the plant can resume operation and can cost a large amount. In addition, they make clear that such practices may cause conditions in which the plant may sustain damage.

Work by Hametner et al. [5], notes that while the agile Test-driven Development methodology is good for producing software that does not interact with external hardware and processes, some adjustments are necessary for using such methodologies in the context of industrial automation. Their work explored the use of different UML models to help model industrial plants and processes and whether these would allow test cases to be automatically produced. Their work did not however, consider how these tests could be executed off-line as we have proposed above. Work by Ritala et al. [12] also investigated the suitability of using UML to model industrial automation processes. They argue that previous version of the language were not considered suitable for such modelling activities, but that new versions of the language has recieved improvements and now offered potential for use in this area.

Some interest has been shown in the application of model driven engineering in the chemical industry. An instance of this can be seen in the work by Kandare et al. [7]. They propose defining a model of a desired system using a modelling language known as “ProcGraph”. They explain that code generation methods can then be used with the constructed model to produce code targeted at a given vendors platform.

A major area that currently appears to be lacking attention is the integration of modelling techniques with reuse concepts. When exploring how the analysis of domains takes place, Prieto-Diaz [11] notes that one of the major applications of domain analysis and modelling is to help create reuse opportunities. This leads us to believe that the basing of our approach on a domain model will aid in proposing a generic solution for the chemical domain.

Model checking and theorem proving tools, such as the Alloy [3] tool, are able to verify systems based on a full formal model of a system. We believe however that in our instance, where we are considering non uniform behaviour resulting from the interaction of analogue elements with a digital control system, that our simulation approach will prove more flexible.

5 Conclusion

This paper has highlighted the difficulty in adopting iterative software engineering methodologies when developing software for chemical plant control systems. Specifically, the current methods and tooling available to engineers do not provide the ability to perform effective off-line testing of new or modified software. As such both testing and tuning of complex

parameters is often performed on a live chemical plant, which can introduce unnecessary risk, take an extended period of time and can involve a large expense in wasted input materials.

We have proposed the creation of a generic simulation approach, based on a domain model of core elements of chemical productions plants. This will allow engineers to effectively and efficiently produce a model which, when execute with the control software, can then be used for off-line testing of the plant software. Once the engineer has had the opportunity to use the simulation to fix any unwanted behaviour and tune various parameters for best performance, the software will then need to be tested on the live plant. This will allow a truly iterative software engineering process to be adopted, as the bottleneck and cost of the live testing stage will have been removed from the iterative cycle.

Finally we have detailed the methods through which we propose to evaluate the approach proposed. In the first instance, we will use real world recipe programs paired with records of previous behaviour and large amounts of plant state outputs. Further evaluation will aim to gauge the effectiveness and efficiency of the approach using observations and interviews.

References

1 Alessandro Brusaferri, Andrea Ballarino, and Emanuele Carpanzano. Enabling agile manufacturing through reconfigurable control solutions. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8. IEEE, 2009.

2 International Electrotechnical Commission et al. Iec 61131-3. *Programmable Controllers-Part*, 3, 1993.

3 Daniel Jackson. alloy: a language & tool for relational models. `http://alloy.mit.edu/alloy/index.html`, 2015. Accessed: June 2016.

4 Alpana Dubey. Evaluating software engineering methods in the context of automation applications. In *2011 9th IEEE International Conference on Industrial Informatics*, 2011.

5 Reinhard Hametner, Dietmar Winkler, Thomas Östreicher, Stefan Biffl, and Alois Zoitl. The adaptation of test-driven software processes to industrial automation engineering. In *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pages 921–927. IEEE, 2010.

6 JUnit. JUnit - Abou. `http://junit.org/`, 2016. Accessed: May 2016.

7 Gregor Kandare, Giovanni Godena, and Stanko Strmčnik. A new approach to plc software design. *ISA transactions*, 42(2):279–288, 2003.

8 C. Larman and V. R. Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, June 2003.

9 Oscar Ljungkrantz, Knut Åkesson, Martin Fabian, and Chengyin Yuan. Formal specification and verification of industrial control logic components. *Automation Science and Engineering, IEEE Transactions on*, 7(3):538–548, 2010.

10 PLCopen. PLCopen for efficiency in automation. `http://www.plcopen.org/index.html`, 2016. Accessed: May 2016.

11 Rubén Prieto-Díaz. Domain analysis: An introduction. *ACM SIGSOFT Software Engineering Notes*, 15(2):47–54, 1990.

12 Tuukka Ritala and Seppo Kuikka. Uml automation profile: enhancing the efficiency of software development in the automation industry. In *Industrial Informatics, 2007 5th IEEE International Conference on*, volume 2, pages 885–890. IEEE, 2007.

13 Schneider Electric Software, LLC. SimSci PRO/II. `http://software.schneider-electric.com/products/simsci/design/pro-ii/`, 2015. Accessed: June 2016.