Applying Formal Methods

Joan Yoon
College of Arts and Sciences
Lewis University
Romeoville, IL 60446
Email: joanhyoon@lewisu.edu

***Abstract*—This paper reviews two pieces of literature that surveys examples of formal methods in current industries. With the use cases presented in each, it shows the challenges dealt with such verification tools despite realized positive impacts (in a general sense) on the development and maintenance of software and hardware. Looking at the Mondex example specifically as well, formal methods are very much reliable and relied on to preserve safety and security of the formulas and consequent end products.**

I. Introduction

Technology has come a long way, and is still growing at an impressive rate. With a majority of our devices and tools being technologically tied, it is obvious there is a heavy reliance on such tools. Technology plays a major part in all aspects of our lives, personal and business. Nowadays, people are tying multiple apps into their phones to even control their house systems with it, great technological advancements within transportation systems make it fun and fast to travel, and it even exists within our workspaces to manage data.

An important factor to note would be how it touches into the sensitive subject of technology and finances. Of course technology in this field makes it easily accessible, however it still means that there is a link to our very personal information, somewhere. We place a great deal of trust in the companies that arrange such services. We have confidence in them because of their name and because we assume that they have taken the necessary precautions and steps toward securing their business processes.

As presented in the case studies done on Mondex, for example, it is vital that the proper security measures are put in place on a foundational level, and are very stringent in its requirements for the integrity of customers' confidential information. As a result, when it comes to money-related electronic smartcards, security must be seamless from the direct design to the backend transaction system flows.

II. Literature Reviews

The papers reviewed gave a view into formal methods and how it has greater accuracy in verification than traditional, more manual methods. Its complexities make up for the broad scope of processes that software and hardware programs execute. Although some researchers may negligently focus on immediate results as opposed to long-term benefits of utilizing these tools, the overall results that both papers concluded upon observe better security and safety in real-life applications.

A. Woodcock, et al: "Formal Methods: Practice and Experience"

Formal methods are mathematical techniques for the development and verification of software and hardware programs. Woodcock and his associates in their paper "Formal Methods: Practice and Experience" go over the industrial application of formal methods, from the early stages to its current levels of growth. Investigation into several surveys with varying viewpoints actually shows common grounds of the challenges that may be faced when adopting it into the industrial industry.

Woodcock and associates then conducted an extensive research to understand the scope of the challenges met, its trends and advances. They surveyed several industrial projects utilizing formal techniques, and yielded the following results. Although the results provide some skewed views (e.g. mainly tester and architect roles in the project teams, majority of techniques used were for specification and modeling, executing, and inspection [4], the responses do represent the great part of the industry and can be applied to show the overall effect of formal methods. Based on this feedback, it was observed that there was a large reduction in time and costs, and also improved quality. The review cross-checking into a series of industrial formal methods projects further provides evidence that formal methods, from static analyzers, model checkers and theorem provers improve the process into software development and maintenance.

Despite the findings of its success, "verification technology and formal methods have not seen widespread adoption as a routine part of systems development practice, except, arguably, in the development of critical systems in certain domains" [4]. This may be because there is insufficient evidence for its cost effectiveness to convince project managers to take that initial risk. Some may see fault in the tools available at this time. Whichever the reason, the idea for the Verified Software Repository launched in order to challenge these weak beliefs and encourage the adoption of formal methods. This goes along with the conclusions given in the paper to "support Hoares vision of a future world in which computer software is always the most reliable component in any system which it controls" [4].

B. Armstrong, et al: "Survey of Existing Tools for Formal Verification"

Whereas Woodcock and his associates focused on the industrial industry, Armstrong and his associates focus on formal methods in the application of general digital systems in their paper "Survey of Existing Tools for Formal Verification." Claiming that traditional verification methods such as testing and simulation are limited for digital designs, they detail formal methods by looking into specific examples of models exemplifying tools for the following functions:

- to check abstract models
- to check hardware description languages
- to check the correctness of software
- to create a provably correct design

Each section of functions identifies specific tools and gives several examples. When checking abstract models, it is common to use custom languages and they are "targeted towards specific application domains ... [with] tools used to reason about high-level properties of a system that abstract out implementation detail" [1]. Verilog or VHDL design files can be used to check hardware descriptions so a separate model of the design doesn't have to be created; the hardware design is analyzed in conjunction with the specified properties. For formal, "after the fact" verification, it is best to analyze the design directly, even though this may be difficult based on applications not necessarily aligning or the lack of a specification. For actual design correctness verification, the "process starts with an abstract model or design that satisfies the required properties and then is iterated (refined) into a form that can be implemented" [1].

By presenting these descriptions, they recognize that there is no perfect tool that can encompass all applications, as they have varying inputs for behavior. Safety, security, and reliability is even more difficult to cover without a formal approach. Nevertheless, a conclusion is made with requirements needed to apply as an effective verification tool. They further emphasize their point by asserting that Rodin/Event-B is the best representation for the purpose of digital designs, and offer enhancement suggestions to perfect it for the application [1].

III. Application: Card Security

A. Mondex Overview

The Mondex was an electronic cash smart card developed in the 1990s by the National Westminster Bank and Platform Seven, "suitable for low-value cash-like transactions, with no third-party involvement, and no cost per transaction" [4]. The cards could be locked using a four-digit passcode, as requested by customers in a focus group discussion; although when the product was initiated for the public, many people didn't regularly use this feature, it gave ease of mind to users that they at least had this option. Value could be loaded onto the card from an ATM machine or a phone, which made it very convenient than carrying around different forms of cash [3].

B. Rationale

Being that its business case was to make sure electronic cash was not counterfeited, it had to be completely secure. John Beric, head of security for Mondex, described the aim of the design framework was to "[present] the criminal with the problem of turning the beef burger back into the cow" [3]. By this he acknowledged that they couldn't make it tamper-proof, but could make it very tamper-resistant by putting in place a working system of firm prevention barriers, sharp detection tools, and powerful recovery mechanisms.

1) Z Specification: It was a goal to achieve the highest standard of security, ITSEC Level E6 certification. The process towards fulfilling the checklist of requirements for this was lengthy, taking almost ten years to completely roll out. The software house Logica applied Z for satisfying the additional mandate to the strict requirements, tying the abstract security policy model with the concrete design. Through Z, they were able to successfully formalize the the architectural design, but faced challenges in the security aspect corresponding to the specification. A small change in design could have made this process easier, but sacrificing time for the predicted high costs, they instead chose to manually write a 200-page proof. The proof revealed a small flaw in one of the minor protocols. This essentially called for the complete development to undergo subsequent testing and further evaluations on that testing to confirm the validity of the certification, to which no errors using formal methods were found at that point.

In the 2006 Grand Challenge in Verified Software, Mondex was used as a pilot project in the testing of the growth of verification methods and to ultimately add to the Verified Software Repository. Eight groups used different formal methods to execute their verification: Alloy, ASM, Event-B, OCL, PerfectDeveloper, π-calculus, Raise, and Z. For the last method mentioned, it was found that the cost of mechanizing the Z proofs was only 10 percent of the original development cost. Overall, almost all techniques achieved the same level of automation.

2) Model Checking: Based on the different studies done on the Mondex application, it is clear that security was investigated very seriously at all levels. Reng Zeng and Xudong He narrowed their analysis by focusing on the SAM specification using SPIN to model check under the Promela program.

SAM is a formal software architecture model integrating multiple compositions, which may contain multiple elements. Each element has a behavior model that is in turn modeled in a high level Petri net, and a property specification that is defined by a temporal logic formula. Correctness of an element is determined if the behavior model satisfies the property specification. "SPIN uses Promela as its input language to model the behavior and uses linear temporal logic to specify the properties" [5].

From this formal method, three bugs were found in the Z specification: one for missing constraints about authenticity, two related with reasoning errors during refinement. Regarding the latter, it is significant to note that the specification done under SPIN avoids this bug by adding proper constraints [5]. With such results, it really shows how although the Z method used is a great initial base, the knowledge drawn from the results of the extended studies should be taken into consideration to provide the greater picture.

3) Rigorous Approach to Industrial Software (RAISE): Now let's look at it in the RAISE approach. This method is "based on stepwise refinement using the invent and verify paradigm, [meaning that] in each step a new specification is constructed and proved to be a refinement of the specification of the previous step" [2]. In this manner, it seems as if the loose ends would be covered.

Chris George and Anne Haxthausen's aim was to see if the original Z proof could be mechanically proved, and if so, could be confirmed. They also wanted to see how much could be automated. To produce the answers, they defined and investigated the following problems:

1) specifying the protocol in detail
2) proving that each operation satisfies two conditions:
    a) the value in circulation in the system will not increase
    b) all value is accounted in the system.

They used the RAISE specification language RSL which has a very wide spectrum. However, it was translated to SAL since model checking is easier than proofing, can conduct checks before proofs, and also to demonstrate liveness properties. They applied this language when developing their structure based on the original Z approach, although it was unable to match on a one-to-one basis. In a closed loop form, the security properties are expressed in terms of the abstract model (already proved correct here), the between model refines the abstract model, and the concrete model refines the between model - proving the security properties are satisfied. This case displays the achievement of mechanically proving the properties of the Z work and successfully specifying the complete protocol. However, it was difficult in that a lot had to be done manually again, and not even necessarily due to costs this time around. The level of expertise required to do such proofs is also quite high [2].

IV. Conclusion

Historically, we started off with bartering of items, and now we use more sophisticated metal and paper currencies. We have even evolved to the age of technology in this sector, where electronic payment is more preferred, from credit and debit cards, bitcoins, phone chips and mobile payment services. It did take time for each set of forms to be adopted fully, but the advancement is tremendous.

Trust of the consumers to the service is the key to continued success. Although electronic transactions are not a total replacement of cash, it is certainly a nifty alternative. Placing value on these little cards is what makes the invention so significant. The value is determined by the monetary amounts, and the links to individual fields of personal information. To secure each matter, we must look at the system structure of the card.

The Mondex study gives a good illustration of the importance of security on money-related smart cards especially. The functions of the card itself may have been limited, but the composition allowed it to attain the ITSEC Level E6 certification. Despite all of its extremely strict requirements, it is a great indicator when the card can potentially still be hacked that there are limitations to even formal methods of verification. This is also presented by how when using a specific method, some parts had to be completed in a painstaking, manual manner to make up for lack of expertise or project constraints. Between reward and return, the backend controls are usually complex in nature (i.e. extensive engineering for user friendly operation).

I want to conclude, however, that the studies do provide distinct evidence on how formal methods have an advantage over traditional ones. It offers long-term benefits of reasonable cost effectiveness, as well as efficiency in design and maintenance of the programs. Using these methods to construct the architecture of cash cards like Mondex would grant better assurance on aligning with the proper security protocols.

References

[1] Armstrong, R., Punnoose, R., Wong, M., Mayo, J. (2014). *Survey of Existing Tools for Formal Verification* (Rep. No. 20533). Albuquerque, NM: Sandia National Laboratories.

[2] George, C., & Haxthausen, A. (2007, February). Specification, Proof, and Model Checking of the Mondex Electronic Purse using RAISE. Macao: United Nations University International Institute for Software Technology (UNU-IIST).

[3] Ives, B., & Earl, M. Mondex International: Reengineering Money. Retrieved from http://wings.buffalo.edu/academic/department/som/isinterface/is_syllabus/mondex /mondex.html

[4] Woodcock, J., Larsen, P. G., Bicarregui, J., Fitzgerald, J. (2009, October). Formal methods: Practice and Experience. *ACM Computing Surveys (CSUR), 41 (4).*

[5] Zeng, R., & He, X. (2010). Analyzing a Formal Specification of Mondex using Model Checking. *Theoretical Aspects of Computing - ICTAC 2010 (6255).*