Developing An Extendible Framework For Continuous Integration In Embedded Systems

Uncovering Attitudes And Experience Regarding Continous Integration From Embedded Engineers

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Peter Tessier

November 8, 2024

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Rider Foley, Department of Engineering and Society
Charles Reiss, Department of Computer Science

INTRODUCTION

Testing is one of the most critical components of software development. Improper testing has caused loss of a space probe, loss of aircraft, collapse of bridges, and false nuclear alerts (Unwin, C., & Ould, M. A., 1986). Consumer trust in the product is also at stake (Selinger, 2021). Historically, software testing was done manually and was more of an after-thought. Such manual testing came with many problems: time consuming, requires a big human investment, prone to human error, and non-programmable, to name a few (Sharma, 2014). These problems can be alleviated with automated tests, performed via a script and with rapid and frequent results. Recognizing its benefits, the industry has made a huge push for test automation in recent years, with the global test automation market expected to reach $49.9 Billion by 2025, a 214% increase from 2019 (Testlio, 2024).

With these automated tests in place, it becomes possible to practice Continuous Integration (CI), where changes are made incrementally and immediately integrated into the main product. Such a practice considerably speeds up the release management and delivery process, which historically involved the Operations team to perform the burdensome work of integrating and deploying the long-disparate software after the developers finished their features (Syed, 2018). The benefits of CI have been long recognized by the industry, with CI adoption increasing from 16% to over 50% in 2020 (Testlio, 2024).

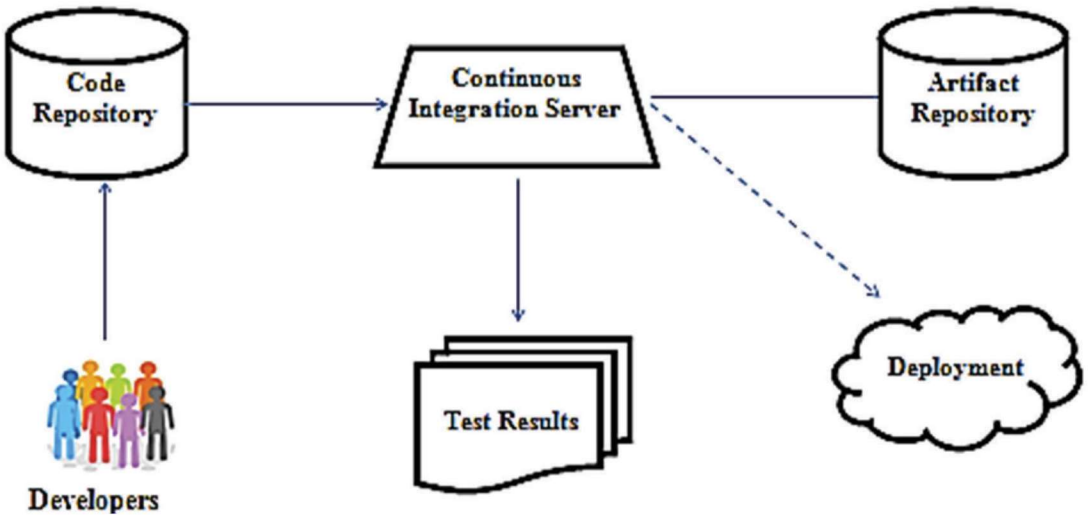

Figure 1. Continuous Integration Environment (Source: Syed, 2018)

While CI can be more effective than manual testing and deployment, it comes with its own set of challenges, especially in the realm of embedded software, which runs on often-specialized hardware instead of the cloud or general computers. Wind River Principal Technologist Woolley (2021) lists several such challenges:

- It is tightly coupled to specific hardware.
- It is written in lower-level languages such as C/C++.
- It interacts directly with hardware (e.g., peripherals).
- It requires specialized development and management tools.
- It tends to have a long lifecycle and stateful execution.
- It faces an increasing diversity of end hardware and software deployed in the field.

Authors Bajer, Szlagor, & Wrzesniak, (2015) add how it is more expensive to duplicate the hardware, causing a limitation on how many tests can be executed in parallel. They also explain the greater number of steps involved in deployment, often requiring physical connection

via USB or other cables. Despite these difficulties, both parties still emphasize the need for CI in embedded devices.

Beyond technical challenges, the success of automated testing also hinges on the development team's mindset. For CI to be effective, developers need to buy into the idea of integrating and testing their code frequently. In opposition to this, a 2024 survey indicated that one third of developers reported new test automation as “unfavorable” instead of “promising”, indicating a significant level of pushback (Dilmegani, 2024). For embedded engineers, the difficulty to adopt CI is only exacerbated by the inherently greater complexity in deploying to embedded devices.

This paper then addresses the question: What can be done to help embedded software engineers adopt Continuous Integration? This question is viewed from two angles. From the technical angle, the author attempts to alleviate the higher barrier to entry by designing a test automation system that can be extended to a variety of common embedded devices. The hope is that such a system will provide a welcome framework for automating deployment and testing, making it easier than starting from scratch. From the STS angle, the plan is to carry out interviews and a survey to get feedback straight from developers as to their attitude toward and experience with CI. The responses from embedded engineers with then be compared to responses from other types of software developers to see what solutions can be tailored to this group. Ultimately, the goal is to bring embedded software teams closer to a streamlined process of testing their products so that software integrity can be maintained.

DEVELOPING A TOOL FOR EMBEDDED CI

One of the largest barriers to adopting CI is finding the right tools to perform the necessary automation; in fact, 26% of companies in a 2024 survey indicated it as the biggest challenge they face (Dilmegani, 2024). This technical project seeks to alleviate this challenge for embedded engineers by producing a reusable, automated testing library of code that handles the majority of repetitive work for them, leaving the developer to focus on implementing the parts specific to their device.

In parallel to developing this library, a test automation system that depends on the library was developed for a specific embedded device with the codename “gouda”. The benefit of this strategy was that it exposed exactly what features would be useful in such a library because they were being used for that particular system. Essentially, any functionality that could be reused across multiple devices was factored into its own section, separate from device-specific functionality. As a bonus, the gouda-specific implementation could also be used as a working example for anyone learning to use the library. Below, the steps of the automation process are outlined.

First, the user navigates to the online server Jenkins (the most widely-used CI tool with 46.73% market share; Jenkins, 2024; Smart, 2011). Every new feature, after being “built” by Jenkins, is associated with a build number, which the user selects for each component. Components include the Root File System and Kernel, which are standard in embedded Linux (Ronsse, 2017).

After selecting the build number for each component, Jenkins connects to a raspberry pi (or pi for short). This deceptively powerful mini-computer was a natural choice as the “central intelligence” of the automation since it has the specs to execute all the logic necessary, as well as hardware peripherals to be able to deploy onto the physically connected gouda (Molloy, 2016).

The pi finds and downloads the specified components from Artifactory, an industry-standard server for storing binaries (Syed, 2018). During this stage, it implements a caching mechanism to speed the process by as much as 30 times, since it avoids re-downloading recently downloaded files, a common scenario.

Once it has the binaries (the "builds") downloaded, it begins to transfer this data via USB to the gouda's flash memory in a process known as "flashing" (Pravisani). Once the builds are loaded into the gouda, the pi must change the "bootmode" by simulating movement of a jumper. It accomplishes this by opening and closing the necessary circuits via General Purpose Input/Output (GPIO) control, another feature the researcher fleshed out.

Finally, a last GPIO powers on the gouda, which then boots up with the new components. Once it has booted up, the full suite of PyTest-powered automated tests are run against it, which tests for things like file integrity and driver performance.

The hope of this technical project is that it alleviates embedded engineers' burden of developing a implementing CI pipeline from scratch. It does this by providing a library that abstracts common automation tasks (like downloading artifacts, GPIO control, and flashing sequences), reducing the time and effort needed to set up a CI pipeline. It supports this library with documentation and the example gouda implementation, serving as both a guide and a foundation for further customization.

While this technical project seeks to remedy the burden of creating an embedded CI pipeline from scratch, the STS approach looks deeper into the social reasons for difficulty in adopting CI. Either by providing a technology or gathering developer insights, both solutions strive to help CI be integrated into embedded software workflows.

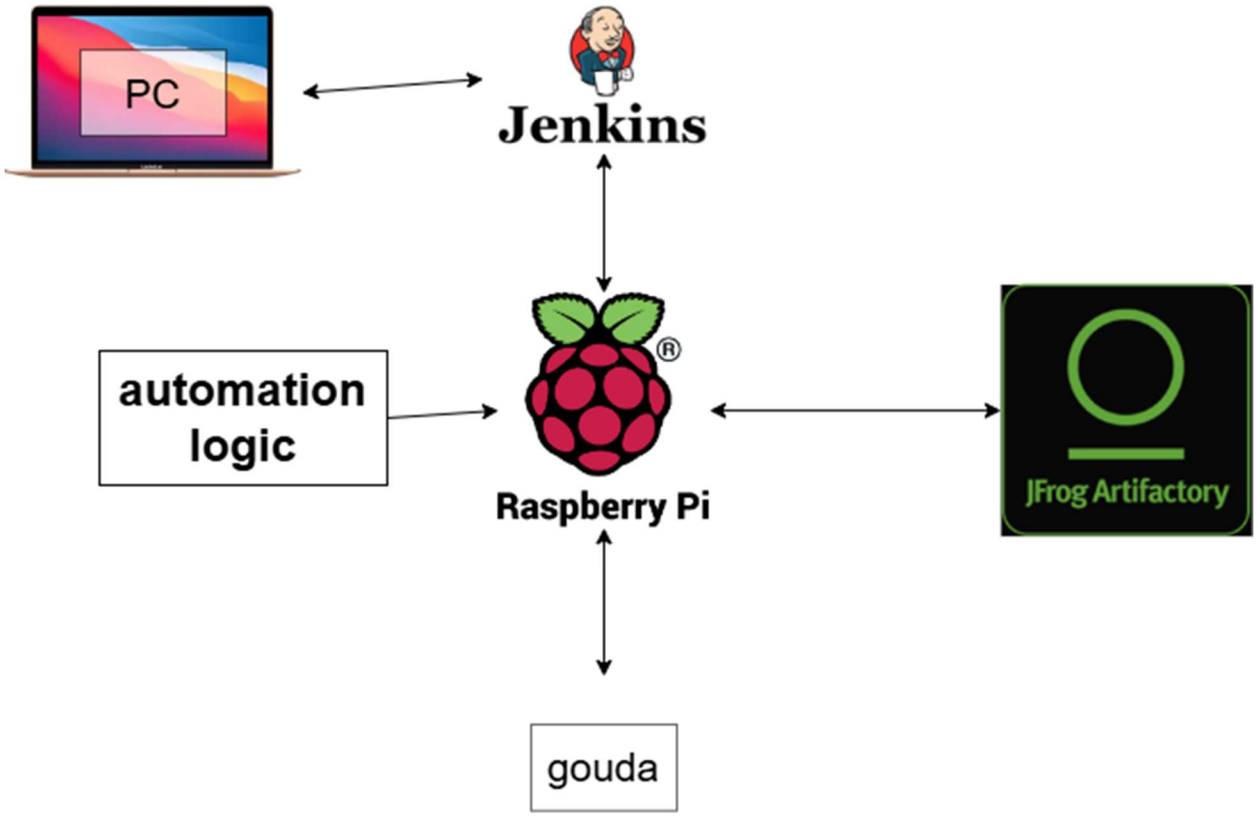

Figure 2. A relational diagram among components of the test automation system

SOCIOTECHNICAL ANALYSIS OF CI

What social factors influence developers' adoption of CI? Why do some developers resist company-mandated changes to enforce CI principles? Such resistance occurred when author and tech CEO Jeff Langr assumed a particular leadership role. He recounts how the codebase was a "mired mess", so he enforced Test Driven Development, a practice where all developers must write their own automated tests before committing any code changes, which supports CI. Some developers were upset and one left, saying it would never be his job to test. As a consequence of this poor testing environment, a high-security chat application was shipped with an "embarrassing defect" (Tarlinder, 2016).

Left with questions from stories similar to Langr's, Laukkanen, Paasivaara, and Arvonen (2015) attempted to get answers from stakeholders such as developers and managers by conducting 27 interviews in the networking and telecommunications firm Ericsson. They explored perceptions toward CI in the midst of a company-wide push to adopt such practices. They discovered that the number one challenge in accepting CI was lack of time, with stakeholders voicing concerns like "I don't think we have the capacity for both in the development organization: to do this at the same time as we do everything else we have committed." When the release date encroached, developers would even cut corners and skip testing in order to get the features out. Frustrations with learning the new technology, dropping everything to investigate build errors, and putting up with test automation failures illustrated more barriers to embracing the CI mindset. Interestingly enough, most stakeholders were in favor of CI as a general practice, but were unhappy with how it was implemented in Ericsson, specifically with the lack of communication and clear direction.

These struggles in accepting the new CI technology in the software development industry mirror the challenges that the healthcare industry faced in accepting new Healthcare Information Technologies (HIT), explained by Harrison, Koppel, and Bar-Lev (2007) in *Unintended Consequences of Information Technologies in Health Care—An Interactive Sociotechnical Analysis*. These authors assert that when HIT was introduced, many doctors and nurses placed the blame of undesirable consequences on the technology itself, when in fact sociotechnical reasons were at play. They frame their analysis in a theory they call Interactive Sociotechnical Analysis (ISTA), which focuses on how technology influences the stakeholders and vice versa in a recursive loop. This theory is natural for examining the relationship between developers and the CI technology, since the same disruption of habits and abuse of technology is observed when

CI is first introduced, with stakeholder and technology influencing each other. For instance, one parallel between the HIT and CI adoptions is how at Ericsson, key members from the 2 teams that spearheaded the push for CI were moved to other teams to help inform and integrate CI practices, just like how managers created tiered alarms in response to physician complaints about incessant alarms in the software system, described in point 5 of ISTA. Both were managerial responses to the technological influence, which in turn affected the technology.

Harrison, Koppel, and Bar-Lev conclude that it is not enough to simply make a good enough technology. No matter how well-designed a technology is, it is important to intentionally integrate that technology into specific cultures, monitoring what adjustments must be made, how the technology is actually used, etc. in order to avoid undesired consequences. This harmonizes with Star's (1999) claim that "infrastructure both shapes and is shaped by the conventions of a community of practice." In the context of CI in embedded development, this means it is not enough to simply craft a technical tool which makes it easier for embedded engineers to create a CI pipeline – work must be done to uncover how they use CI and their perception of it, or such a tool may be misunderstood and used improperly, if at all.

RESEARCH QUESTION AND METHOD

To then understand how CI technologies can be best suited specifically for embedded engineers, this research intends to answer: What experiences and perceptions do embedded engineers have with CI compared to other types of software developers? The method used to answer this is with a survey targeted at all types of software developers, with an emphasis on embedded developers. It is inspired by the research done by Hilton et. al. (2016) which involved a thorough analysis of 34,544 open-source GitHub repositories (involving 1,529,291 builds) and

442 survey responses from developers. Their goal was to uncover a number of questions surrounding CI, including how and why it is used. They discovered an overwhelming number of participants (94%) indicated interest in using CI in their next project, and the number one reason of not using CI is that "The developers on my project are not familiar enough with CI" (with 47% indicating so).

The research of this paper will re-use these questions to compare how embedded engineers' responses compare with the responses of Hilton's survey. The survey will also be open to responses from other types of developers to serve as a control group. See Table 1 for the full list of questions.

Question	Response Options	Motivation
What type of software developer are you?	Select all that apply • Front-End • Back-End • Full Stack • Middle Tier • Mobile • Desktop • Embedded • Database • Cloud • Security • SDET • DevOps • Data Science • Big Data • Game • Graphic • Customization • AI	Compare embedded developers to the others. It is not necessary to get a representative sample from each type of developer, but the list of options should be exhaustive.
How old are you?	Multiple choice • 20 or younger • 21-30 • 31-40 • 41-50 • 51-60	Since CI is a relatively new practice, it could be the case that younger developers are more in favor of it.

	• 61 or older	
Do you use Continuous Integration (CI)?	Yes/no	Compare the responses of embedded with the distribution from Hilton's survey. The list of software development roles was obtained from Merzlova (2023).
If you don't use CI, why?	Select all that apply: • The developers on my project are not familiar enough with CI • Our project doesn't have automated tests • Our project doesn't commit often enough for CI to be worth it • Our project doesn't currently use CI, but we would like to in the future • CI systems have too high maintenance costs (e.g., time, effort, etc.) • CI takes too long to set up • CI doesn't bring value because our project already does enough testing	
If you do use CI, why?	Select all that apply: • CI makes us less worried about breaking our builds • CI helps us catch bugs earlier • CI allows running our tests in the cloud, freeing up our personal machines • CI helps us deploy more often • CI makes integration easier • CI runs our tests in a real-world staging environment • CI lets us spend less time debugging	
If you do use CI, how challenging was it for	Likert scale of Very Challenging to Very Easy	

you to adopt this practice?		
Will you use CI for your next project?	Likert scale of Definitely to Definitely Not	
Has CI helped you with debugging?	Likert scale of Definitely to Definitely Not	
What is your perception of CI?	Likert scale of Very Positive to Very Negative	
(Optional) What were challenges you faced or currently face in adopting CI, if applicable?	Free Response	Get qualitative insight which may be representative of the sample.
(Optional) Please provide any additional insight into your experience with and perception toward CI.	Free Response	

Table 1. Survey Questions

The list is intentionally short so as to get as many responses as possible. A $20 USD gift card will also be raffled off as another incentive. Both these techniques were also used by Hilton to increase response rate. As for obtaining participants, the researcher will post to online forums such as Reddit and LinkedIn as well as email publicly available email addresses of embedded engineers found on company websites.

CONCLUSION

In conclusion, the problem of embedded developers having additional hurdles to embrace CI is approached in two ways. From the technical perspective, a test automation system was developed for a particular embedded device. This system provides features like GPIO utilities, various command-line tools, a functioning pipeline structure, and rich documentation which future embedded developers can use as a starting point to design test automation systems for

their own devices. From the social perspective, the survey is expected to uncover valuable insights from embedded developers as to their history and point of view regarding CI. By understanding how they understand and interact with the technology, pitfalls like those described by Langr, Harrison, and Laukkanen can be avoided.

With both of these deliverables, the hope is to empower a future of CI for embedded engineers. As Sharma, et. al. (2023) indicate in their recounting of the rise of CI, it was never enough to stop at technological innovation. Even with the perfect CI tool, developers who commit infrequently and write poor tests will reap no benefits from it. It also takes social work for all the stakeholders involved to help make the technology work as intended. With such effort, embedded products can be developed with much more reliability and ease.

References

30+ Test Automation Statistics In 2024. (2024, June 21). Testlio. https://testlio.com/blog/test-automation-statistics/

Bajer, M., Szlagor, M., & Wrzesniak, M. (2015). Embedded software testing in research environment. A practical guide for non-experts. *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, 100–105. https://doi.org/10.1109/MECO.2015.7181877

Dilmegani, C. (2024, October 3). Top 20 Test Automation Statistics QA Teams Must Know. AIMultiple: High Tech Use Cases; Tools to Grow Your Business. https://research.aimultiple.com/test-automation-statistics/

Harrison, M. I., Koppel, R., & Bar-Lev, S. (2007). Unintended consequences of information technologies in health care--an interactive sociotechnical analysis. Journal of the American Medical Informatics Association : JAMIA, 14(5), 542–549. https://doi.org/10.1197/jamia.M2384

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). Usage, costs, and benefits of continuous integration in open-source projects. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 426–437. https://doi.org/10.1145/2970276.2970358

Jenkins—Market Share, Competitor Insights in Continuous Integration And Delivery. (2024). 6sense. https://www.6sense.com/tech/continuos-integration/jenkins-market-share

Laukkanen, E., Paasivaara, M., & Arvonen, T. (2015). Stakeholder Perceptions of the Adoption of Continuous Integration – A Case Study. 2015 Agile Conference, 11–20. https://doi.org/10.1109/Agile.2015.15

Merzlova, K. (2023, March 14). 18 Types of Software Developers Positions Explained. SumatoSoft. https://sumatosoft.com/blog/different-types-of-software-developers-jobs-explained

Molloy, D. (2016). Exploring raspberry pi: Interfacing to the real world with embedded linux. *John Wiley & Sons, Incorporated*.

Pravisani, Y. (n.d.). *What is a flash programmer and how does it work?* SMH Technologies. Retrieved October 21, 2024, from https://smh-tech.com/corporate-blog/what-is-a-flash-programmer-and-how-does-it-work/

Ronsse, S. (2017, May 17). Embedded Linux Systems & the Yocto Project | Witekio. Your Embedded and IoT Software Partner. https://witekio.com/blog/embedded-linux-demystified-3/

Selinger, E., & Durant, D. (2021). Amazon's Ring: Surveillance as a Slippery Slope Service. *Science as Culture*, *31*(1), 92–106. https://doi.org/10.1080/09505431.2021.1983797

Sharma, H. K., Kumar, A., Pant, S., & Ram, M. (2023). DevOps: A journey from microservice to cloud-based containerization (1st ed.). *River Publishers*. https://doi.org/10.1201/9781032624310

Sharma, R. M. (2014). Quantitative Analysis of Automation and Manual Testing. 4(1). https://www.ijeit.com/Vol%204/Issue%201/IJEIT1412201407_46.pdf

Smart, J. (2011). Jenkins: The Definitive Guide. Taiwan: *O'Reilly Media*.

Star, S. L. (1999). The Ethnography of Infrastructure. *American Behavioral Scientist*, *43*(3), 377–391. https://doi.org/10.1177/00027649921955326

Syed, S., & Soomro, T. (2018). Achieving Software Release Management and Continuous Integration using Maven, Jenkins and Artifactory. *International Journal of Experiential Learning & Case Studies*, *3*. https://doi.org/10.22555/ijelcs.v3i2.2451

Tarlinder, A. (2016). *Developer Testing: Building Quality into Software*. Pearson Education. https://books.google.com/books?id=Csz3DAAAQBAJ

Unwin, C., & Ould, M. A. (1986). *Testing in Software Development*. Cambridge University Press.

Woolley, R. (2021). Deploying Embedded Applications Faster with Containers. Wind River. https://www.windriver.com/resource/deploying-embedded-applications-faster-with-containers