Managing Dermal Reference Guides in the Face of Software Evolution via a CI/CD Pipeline

Erik M. Fredericks, Abigail C. Diller, and Byron DeVries
Grand Valley State University
frederer@gvsu.edu, dillerab@mail.gvsu.edu, devrieby@gvsu.edu

ABSTRACT

Software evolution ensures that updates are provided due to changing use cases, discovered flaws, and updates to standards. However, such a process can be problematic when referencing APIs from external entities, given that function calls or remote service invocations often have strict requirements for access and use. As APIs change over time, the connecting software must either be updated to comply with new requirements or reference a static older version. Often, developers and engineers will use local reference guides to ensure that API information is close at hand. The problem we describe here lies in keeping those local references up to date in the face of evolving software, whether it is printed on a piece of paper or imprinted onto one's skin. We propose a CI/CD pipeline for managing such references throughout the software evolution process.

1 INTRODUCTION

Software evolution is the process of software changing over time, ideally for the better [1, 2]. For example, software evolution can include algorithm improvements that yield better or faster results, security updates to prevent bored teenagers or automated botnets from intrusion, and user interface updates resulting from the general ennui that developers face in their day-to-day lives in that they must be doing *something* useful. Such updates often impact the greater system in that changes need to be reflected in requirements, models, and API specifications. These changes then need to be further reflected in any other projects that use the updated software.

Reference guides are a very common method for looking up relevant or salient information related to a particular task. There exist many printable or electronic guides for any practitioner to quickly determine relevant information, from automatically-generated API documentation on project websites to pocket reference manuals. For example, the common ruler provides a handy reference to length measurements, or the Sphinx-generated Sphinx API documentation can tell you how to use Sphinx to automatically generate and publish your API documentation [3]. Such guides are useful, however must be kept up to date as well to reflect any changes made.

This paper focuses on those reference guides that are physically-manifested and cannot be easily updated to keep pace with constant changes. To this end, we propose a continuous integration/continuous deployment (CI/CD) pipeline for managing API changes for dermal reference guides. Specifically, we focus on how a permanent reference that has been etched onto one's skin can be kept up to date with the latest changes from the originating reference. For the purposes of this paper we use a specific application of the Unified Modeling Language (UML) as a case study, however acknowledge that our approach could be applied to any reference.

Figure 1 demonstrates the need for offline reference material. This figure shows a screenshot of the UML specification website in an unavailable state (accessed February 18, 2026), as we had a need to reference the formal class diagram specification. If this specification were available offline then, for example, we could continue our work. Assuming we had an offline reference, however, the issue of it being up-to-date persists, as we would not necessarily "know" if our offline reference was valid as the online reference may have been updated in the interim.

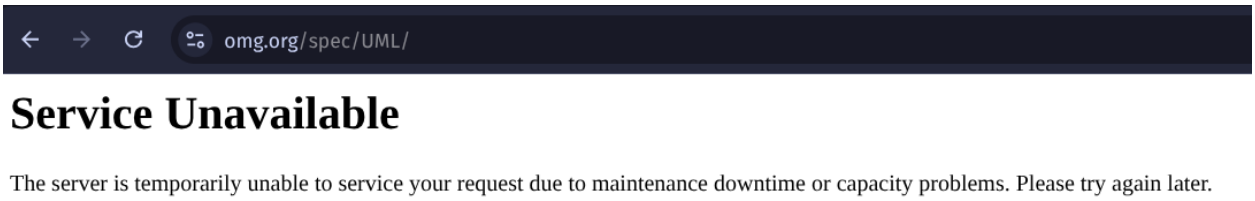

Figure 1: UML website down for maintenance - February 18th, 2026.

The rest of this paper is structured as follows. Section 2 provides relevant information on dermal reference guides and UML. Section 3 then discusses our proposed CI/CD framework with examples of it in action. Section 4 summarizes our efforts and presents avenues of future research.

2 BACKGROUND

This section provides background information on the concerns with dermal reference guides and a brief overview of UML.

2.1 Dermal Reference Guides

Dermal references are figures that are tattooed onto one's body to provide a quick and reliable visual representation for some task. For example, rulers of different units (i.e., metric and/or imperial) have been applied to arms, as seen in Figures 2 and 3. Such guides can be used in the event that a ruler is unavailable or out of reach.

However, an exhaustive Google Image search did not return a UML tattoo, as seen in Figure 4. Such a result implies that there exists a lack of research and industrial acceptance of UML-focused dermal reference guides. One ongoing issue that exists with dermal guides lie in the dermis. Specifically, tattoos tend to lose color over time due to sun damage or lose their crispness due to fading skin elasticity. In such cases, followup appointments to "refresh" a tattoo may be scheduled to ensure that the original reference remains crisp and legible. In the case of a UML

Figure 2: Screengrab (01:38) of Adam Savage's ruler tattoo – "[Plumbing parts] - That is absolutely what it is fantastic for" [4].

Figure 3: A picture of a precisely-scaled inch/centimeter ruler [5].

dermal reference, however, such touch ups may be able to be performed as part of a guide update.

2.2 Unified Modeling Language

UML is a design language that provides multiple specifications for modeling different aspects of programs at multiple levels of abstraction [6]. For example, class diagrams present information

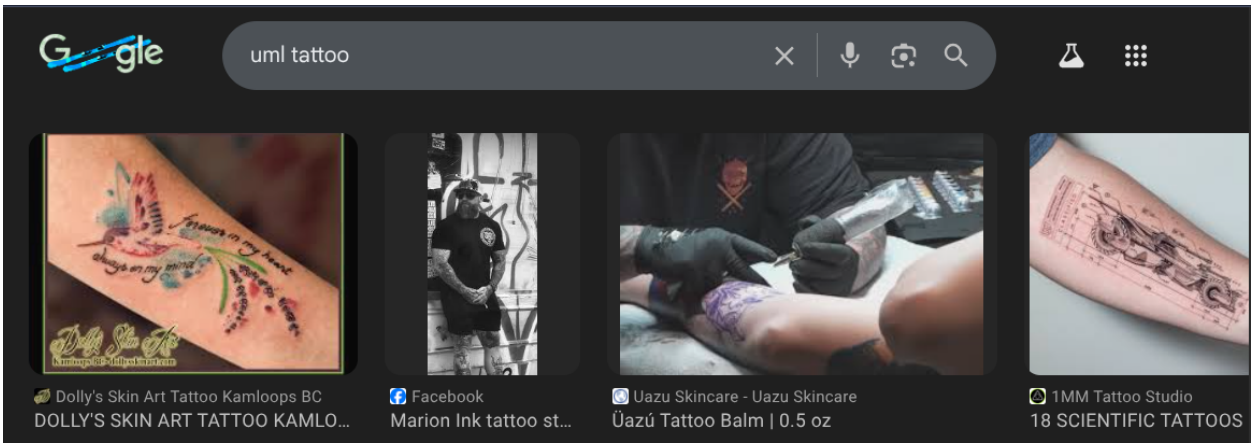

Figure 4: A depressing lack of UML tattoo results in Google Image search. Note that we only included a small subset of results as an impressive number of results are not safe for work.

relating to attributes, methods, encapsulation, and relationships (among others) for either low- or high-level classes in a system. Dependency diagrams visualize relationships and dependencies. Architecture diagrams, unsurprisingly, show how the architecture of a system is constructed. Overall, such diagrams are critical in conveying important information and meaning between project stakeholders. Given the sheer magnitude of the number of different modeling languages and diagram types, one might be interested in having a handy reference guide available to cover the specific meanings of a particular model's syntax.

3 PROPOSED APPROACH

We now describe our proposed CI/CD pipeline for ensuring that dermal references are kept up to date in line with software evolution. First, we discuss any assumptions, expected inputs, and expected outputs for our pipeline. Second, we present a limited empirical evaluation of our pipeline.

3.1 Assumptions, Inputs, and Outputs

We assume that the reference guide under evolution is publicly available and that public notifications are enabled for updates. We also assume that an experienced tattoo artist is on call 24/7 to deploy updates when ready. Last, we assume that the person who will be receiving the dermal update is fine with having a tattoo reapplied to the same spot multiple times. In terms of inputs we require a specification to be applied as a tattoo/dermal reference, one's skin, and tattoo ink/supplies. For outputs we expect to have an up-to-date representation of a reference guide applied to the user.

3.2 Proposed CI/CD Pipeline

Figure 5 presents a data flow diagram of our proposed technique. We next describe each of the steps in turn. We refer to the person receiving the dermal reference guide as the "user" and the tattoo artist on call as the "artist."

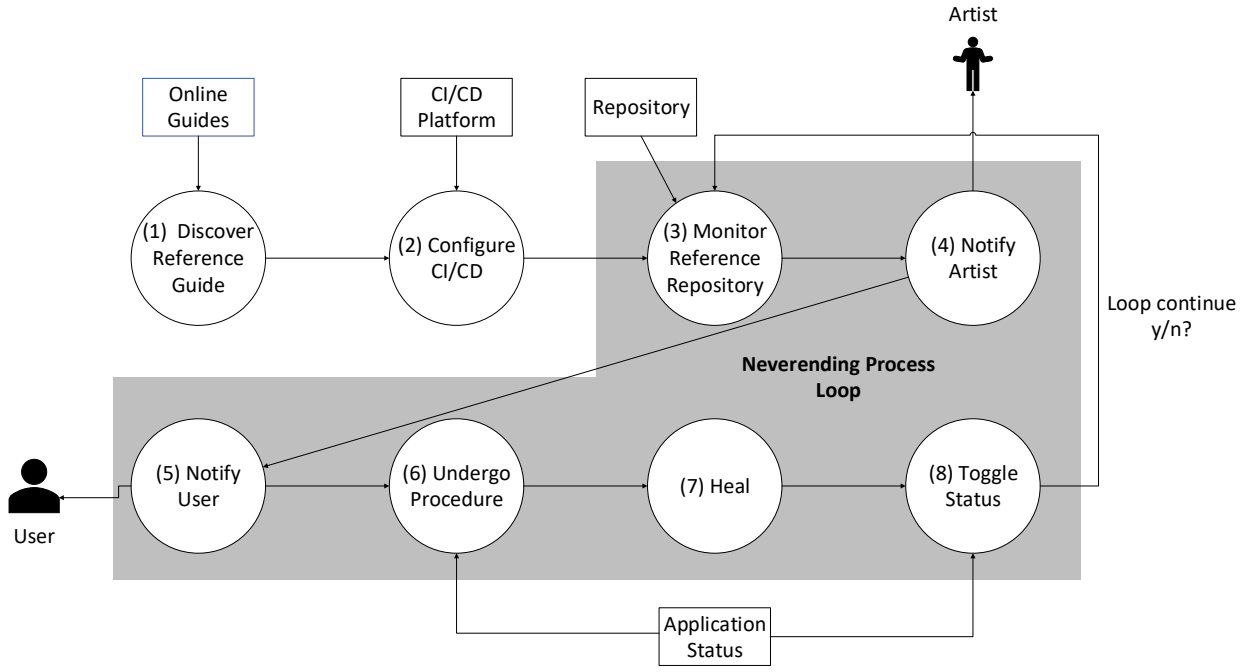

Figure 5: Data flow diagram of proposed CI/CD pipeline.

(1) Discover Reference Guide: First, the user must determine which reference guide is to be permanently etched onto their skin and then placed under revision control. The guide must at minimum (a) be displayed legibly on the user's desired part of the body and (b) provide a notification in the event that an update has been applied. For (b), the notification can be in the form of an email (e.g., to all subscribed users) or an automated notification (e.g., a GitHub Action that publishes formal version updates).
(2) Configure CI/CD: The CI/CD platform of the user's choice must then be appropriately configured. The user and artist contact notifications must be set appropriately, the system itself must be configured to monitor updates from Step (1), and the system must be configured with a notion of state (i.e., has the dermal area been previously used for a guide). Depending on the complexity of the CI/CD system there may also be the potential to auto-convert a visual representation of the specification to a stencil ready for application.
(3): Monitor Reference Repository: Following, the reference guide's repository is monitored for changes to the particular diagram. The granularity of a triggered update notification is left to the user and their respective funds and pain tolerance, however our initial suggestion is to take action only upon major updates.[1]

1 In the event that a minor update changes the reference material, we suggest using a permanent marker to extend the original guide's lifetime.

(4): Notify Artist: When a sufficient change occurs the artist is notified and the new reference guide (and any supporting materials) is transmitted to the artist. The artist is then given time to make any stylistic changes necessary for the dermal application, including color selection as informed by the toggle state (see Step (8)). A preview may then be sent to the user, depending on their preference, and an appointment can be booked through the artist's online calendar, either manually by the user or automatically (e.g., via a GitHub Action). Further, the *status* of the user's application (see Step (8)) may impact the selected foreground color and, if necessary, background color. A preview may be sent to the user as well in this step, depending on their configured preferences.
(5): Notify User: If the session has been automatically booked then user must be notified that a reference update must occur. Regardless, the user must temporarily stop using the dermal reference guide in all activities as it is officially out of date with the formal guide. Furthermore, the user must begin preparing the dermal reference area to ensure that the following procedure is optimally performed. At minimum, one must keep the reference guide area clean and well-moisturized to ensure that the dermis is receptive to the update.
(6): Undergo Dermal Update Procedure: Following appointment confirmation, the user and artist must meet to perform the procedure. Optionally, there may be a discussion about the updated reference guide in the event that linework must be emphasized for visual effect. For clarity the version number of the specification should also be clearly denoted in the reference as well. The procedure is then to be performed depending on the toggle state (see Step (8)). In the event that it is the first procedure then an ideal ink should be selected in accordance with the user's aesthetic preferences to ensure that the guide is appropriately visible. Colored ink may be added for effect or visual emphasis. If an existing dermal reference is to be updated then the procedure differs slightly. First, a "blackout" patch (c.f., Figure 6) should be applied to cover the original reference, so as to not confuse the user when referencing the guide. Second, an appropriate inverted ink color should be selected to apply on top of the patch. Ideally, the user will then undergo a waiting period to allow the blackout patch to heal (see Step (7)) before the second phase. Additional sessions may be necessary to touch-up the lighter ink work to ensure the longevity of the design.

Another option that the user may select is to get laser tattoo removal to wipe away the offending reference guide. In this instance the user and artist do not necessarily require a blackout patch and can use the style of the prior reference piece instead. Further, if the user chooses laser removal instead of covering up then Step (4) may be skipped.

One other note that, while it may be to the user's preference, the guide itself should be placed in an appropriate location that is viewable by the user without the need for a complicated setup of mirrors or potentially-inappropriate queries to coworkers for help in reading the guide itself.
(7): Heal: Following one's session a healing period must be followed to ensure that the reference guide heals appropriately and that all lines and colors applied remain crisp and legible. While there are many different approaches to one's healing process, consulting with your artist (see Step (4)) can ensure the best possible outcome.
(8): Toggle Status: The application `status` must now be toggled as well to ensure an ideal update, where we define two potential status options: *normal* and *inverted*. The system initially should be set to *normal* and the colors of the reference guide should be defined according to one's aesthetic preferences. Following application the status is toggled to *inverted*, where the next update would

be applied to cover the previous version and use an inverted color that can legibly appear on top of the coverup.

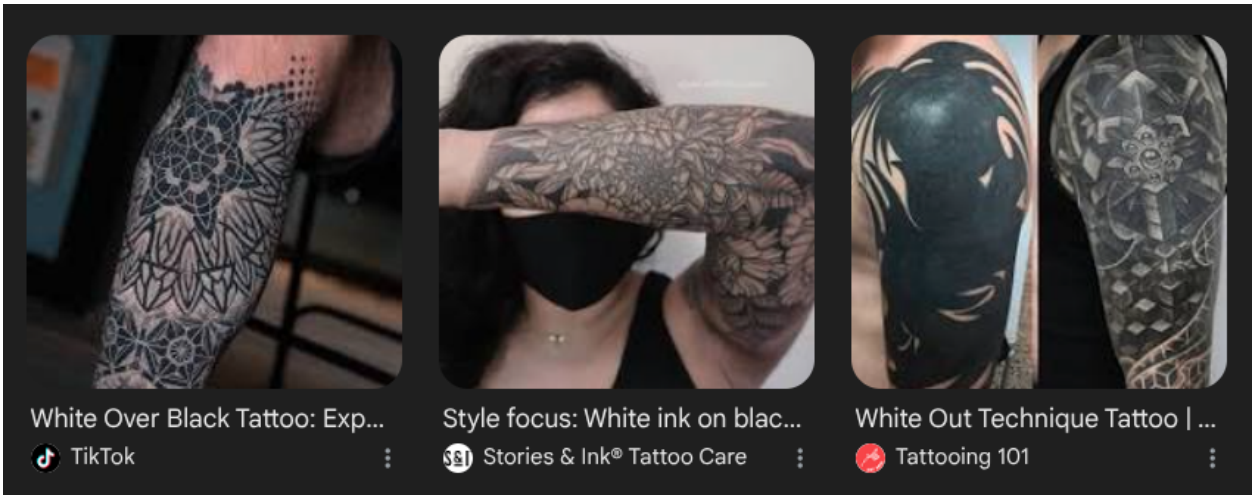

Figure 6: Google Image results showing examples of white ink over a blackout tattoo. Note that different skin tones may impact the choice of ink colors.

The process then returns to (3) and waits for changes pushed to the reference repository, where the loop continues until the user decides to either give up on the whole thing or no longer has a need for that particular guide. We further note that the user should make liberal use of moisturizer and sunscreen, as appropriate, to maximize the guide's effective lifetime.[2]

3.3 Initial Results

We now present an illustrative example of our proposed pipeline in action. First, we identified a relevant reference guide that was necessary to be imprinted on an author's skin.[3] We selected JUnit as it has recently gone through a major revision from version 5 to version 6.[4] Figure 7 presents a screenshot of the JUnit 5 dependency diagram and Figure 8 presents a screenshot of the JUnit 6 dependency diagram. As can be visually ascertained from these figures, there are significant changes from version 5 to version 6 in terms of content, boxes, relationships, and verbiage. These versions are also maintained on the JUnit GitHub repository and its major changes can be monitored over time.[5]

Following our data flow diagram presented in Figure 5, we mentally configured our CI/CD pipeline using GitHub Actions to track the JUnit repository for version changes. The artist (in this case, the author) was contacted and a (temporary) dermal reference guide was applied to the forearm in the "normal" color, as seen in Figure 9.

2 We note that this is good advice regardless.

3 Unfortunately the author in question did not decide to fully embrace the empirical evaluation and instead relied on temporary tattoo paper to demonstrate the proof of concept.

4 The author who agreed to be temporarily tattooed also does not prefer to use Java and as such vigorously washed the area once the experiment was complete.

5 See `https://github.com/junit-team/junit-framework`.

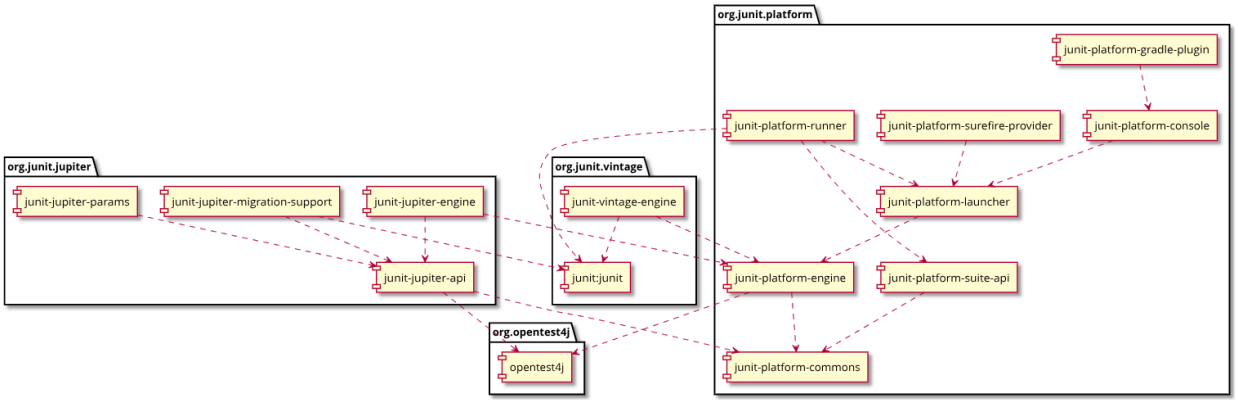

Figure 7: Dependency diagram for JUnit 5 [7].

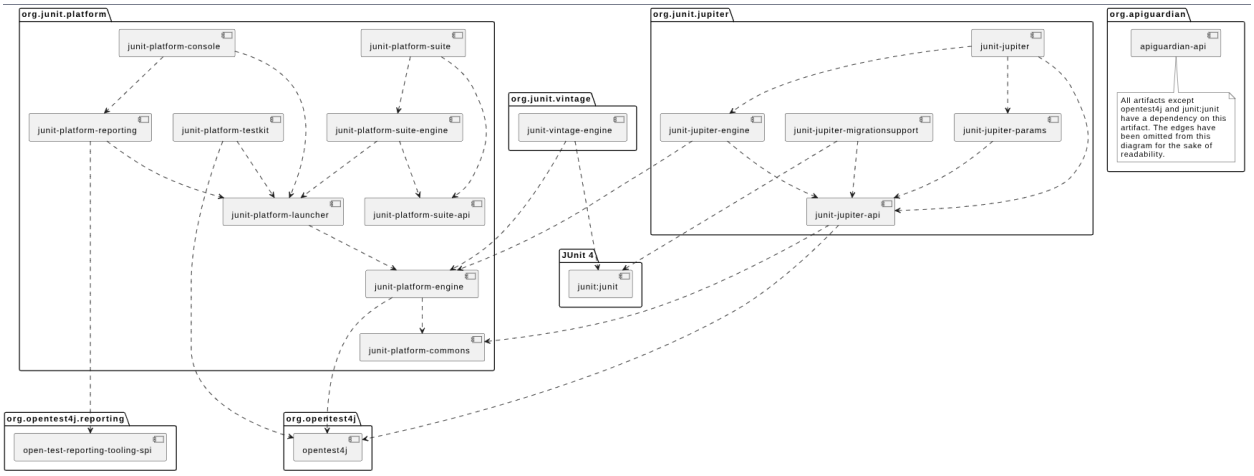

Figure 8: Dependency diagram for JUnit 6 [8].

The area was allowed to heal (or dry in this case) and then the status of our application was toggled to denote that the next version required an inverted color. Note that the guide was thoroughly referenced for a reasonable period of time before the process continued.[6]

Next, the author received a notification that a new version was available. The artist (again, the author) was notified and an appointment was scheduled for a new procedure. Given that the current status is now inverted, the updated reference guide required a bit of tweaking to ensure that it "covered up" the previous guide. This outcome is visualized in Figure 10, where we post-processed the original JUnit 6 dependency diagram [8] using Krita to provide a "blackout" background with an inverted color for the text and lines.

The procedure was performed again as seen in Figure 11 and a healing (or, drying) period was observed. Given that this was the second time through the procedure, the status does not need to be toggled as another coverup procedure would be required for a new version, of which the authors attentively await for a followup study. Alternatively, a laser removal procedure could be

6 In this case, a reasonable time was considered to be five minutes.

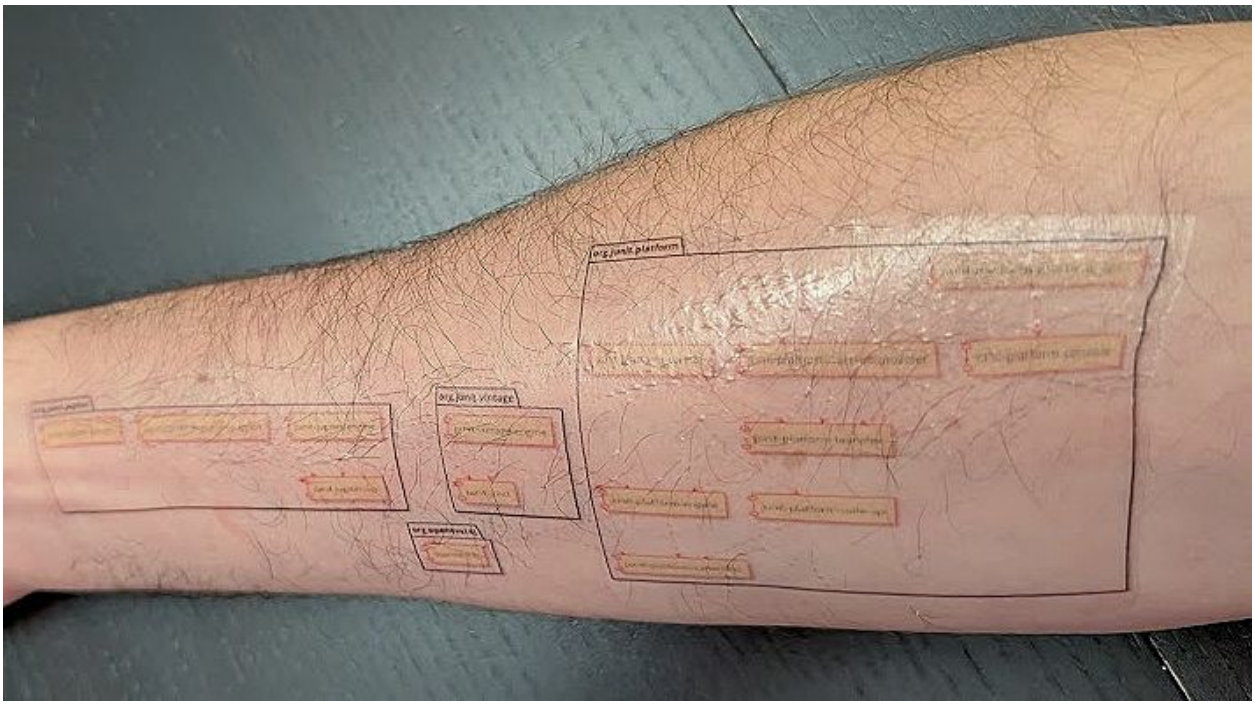

Figure 9: Result of application of JUnit5 reference guide to forearm. Note that the initial application followed improper procedures and the user desperately wished for a reason to update the applied guide.

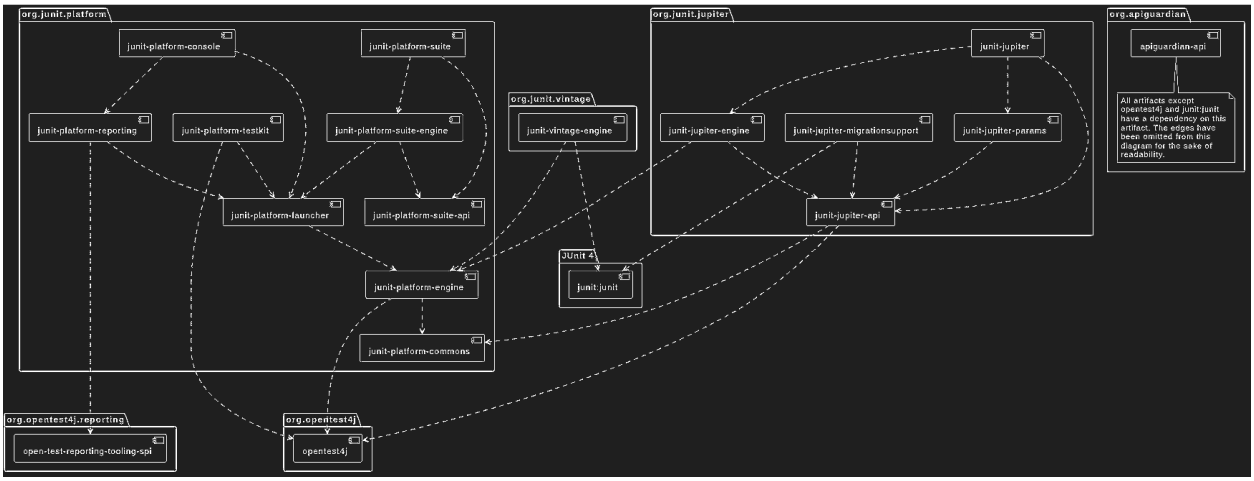

Figure 10: Dependency diagram for JUnit 6 (inverted colors) [8].

performed in the event that the author regrets their choice of a large coverup.[7] However, all study participants agreed that it was "far more impressive," though further opinions are required.

The user then gave up on JUnit and switched over to Python's `pytest` module for a different project.

7 We would like to acknowledge the fact that laser removal surgery is not strictly necessary for a temporary tattoo. Isopropyl alcohol suffices for this form of application.

Figure 11: Application of JUnit6 reference guide to forearm with coverup and inverted color palette.

4 DISCUSSION

This paper has presented a proof of concept process for ensuring that dermal reference guides are maintained as software evolves towards the inevitable heat death that faces us all. We posit that a CI/CD pipeline can be configured to monitor an external system for major changes and that both an artist and a user can accept an automated scheduling procedure. We presented a limited empirical evaluation of our process that kept an artifact of the JUnit framework up to date on one's arm.

Future paths for this research include an automated process for making the reference guide look significantly more impressive on one's body. Further, we plan to investigate if glow-in-the-dark ink is all that it's cracked up to be. Lastly, we will determine which type of cookie is best to bring to an appointment to ensure that both user and artist(s) are well-fed.

REFERENCES

[1] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *Journal of software maintenance and evolution: Research and practice*, vol. 19, no. 2, pp. 77–131, 2007.

[2] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, and N. Zazworka, "Software evolution visualization: A systematic mapping study," *Information and Software Technology*, vol. 55, no. 11, pp. 1860–1883, 2013.

[3] E. Holscher. (2026) Step 2: Building references api docs. Sphinx. [Online]. Available: https://sphinx-tutorial.readthedocs.io/step-2/

[4] A. Savage. (2022) Is it useful having a ruler-tattoo on your arm? Tested. [Online]. Available: https://www.youtube.com/watch?v=c6TopwNu7Ok

[5] J. Baichtal. (2012) Ruler tattoo for handy measuring. MAKE:. [Online]. Available: https://makezine.com/article/workshop/ruler-tattoo-for-handy-measuring/

[6] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2018.

[7] The JUnit Team. (2026) Dependency diagram – junit user guide 5.0.0-m4. JUnit. [Online]. Available: https://docs.junit.org/5.0.0-M4/user-guide/images/component-diagram.svg

[8] ——. (2026) Dependency diagram – junit user guide 6.0.3. JUnit. [Online]. Available: https://docs.junit.org/6.0.3/_images/component-diagram.svg