Mirth Connect

by Software Engineer

EHR Integration

Details

Review Date	08/28/2023
Purchase Date	Q2'20
Implementation Time	1 month
Product Still in Use	Yes
Purchase Amount	10k / year
Intent to Renew	N/A
Review Source	Elion

Product Rating

Product Overall	3.0
Use Case Fit	1.5
Ease of Use	2.0
API	4.0
Integrations	N/A
Support	N/A
Value	4.0

About the Reviewer

Implementation Team

Reviewer Organization

N/A

Reviewer Tech Stack

N/A

Other Products Considered

InterSystems

Rhapsody

Summary

- **Product Usage: The company used Mirth Connect to make transforming data between various HL7 and interoperability standards easier.**
- **Strengths: Mirth Connect's strengths include its open-source nature, availability of a supporting community, feature completeness, ease of testing due to modularity, and its ability to handle various unique connection types.**
- **Weaknesses: The product's low-code environment and its use of an older type of JavaScript interpreter make it a poor fit for experienced coders, and its logging function used excessive disk space.**
- **Overall Judgment: The reviewer found Mirth Connect to be more suited to technically proficient non-coders in IT, and not recommended for software engineers or any high-frequency use due to its cumbersome process and code environment.**

So today, we're chatting about Mirth Connect and how it was used at your previous company. Before we jump into that, could you give a brief overview of the company and your role there?

Yeah, so I worked at an electronic health record startup starting in November 2021, and I stayed there for about a year and a half. As a senior software engineer, I was part of the API integrations team. Our main objective was to create a platform EMR that others could use as a foundation for their own projects. One of the key components of our infrastructure was Mirth Connect, which served as our integration engine for cross-platform interoperability. I worked with it on a daily basis for around six months. However, I eventually proposed and took charge of a project to migrate away from it.

How long were you using Mirth Connect?

I began using it in November 2021, and for about six months, I used it on a daily basis to write code and build the API infrastructure. After that, I used it less regularly for maintenance and managing existing features. In January, our FHIR team migrated away from it, though I still had to interact with it occasionally for other use cases.

What caused your company to look into purchasing Mirth Connect?

So our company needed a FHIR API for healthcare data exchange. And Mirth was there as an open source option. One of the people involved in the decision-making process had actually used Mirth before, but in a different way, more on the IT routing side. They were aware of the FHIR plugins in Mirth and knew that it could be useful for our purposes. Plus, Mirth offers a bunch of visual tools to help create FHIR resources. The idea behind using Mirth was that it would make it easier for a team with less experience in this area to learn FHIR quickly and also ensure that the resources were structured correctly.

What problems did Mirth specifically solve for you?

Mirth provided various unique connection types that were not typically found in standard integration tools. It included connectors required for different integrations, such as SFTP or SOAP, along with the more commonly used JSON. We wanted one integration team to handle most of these connectors, and it came with built-in plugins for many of them. This made it the primary entry point and offered a visual way to learn about these connectors.

Different organizations with which we wanted to achieve interoperability had various endpoint types. These could be SOAP WSDLs, REST API endpoints, SFTP, or any other similar protocols. To ensure we could ingest and push data to each of these organizations, we had to build a standard set of connectors. This was a proactive measure because we anticipated encountering older IT organizations that tended to rely on these types of connectors. So my understanding was that it was not so much that we were currently using these connectors, but more of a preparation for encountering them in the future.

What were some of the requirements that your organization used in evaluating Mirth Connect or other FHIR connector products?

We needed a customizable infrastructure, so outsourcing to something like Redox or 1upHealth wasn't an option. We had to have control over our data model and the ability to implement any necessary logic, especially since we are a

platform-oriented company. Scalability was also a requirement. In terms of runtime performance, Mirth is built on Java and allows for easy implementation of multi-threading. Another advantage was that we could self-host, which aligned with our platform strategy. Additionally, Mirth being open source allowed us to quickly test it out. We also had prior experience working with Mirth, so it served as a stamp of approval for us.

What made you procure the license, rather than just using the software in an open source way?

Mostly because the cost was not very high and having support staff available in case any issues arise is a nice reassurance. Another factor is that the tool, despite being open source, is very much a B2B software. As a result, there is a lot of hidden configuration. I believe the main reason for getting support was to ensure that we were using the tool correctly and following the recommended practices.

Did you evaluate alternatives to Mirth Connect? How did they stack up against each other?

One competitor we considered was InterSystems. They have their own interface engine and data platform combined. Another startup we came across was Iguana by iNTERFACEWARE. They also offer licensable software with a better interface engine. There's another called Rhapsody, which is an interface engine. There was also Cloverleaf by a company called Infor.

In terms of how they differ from Mirth Connect, Cloverleaf and Rhapsody are similar, but more business-oriented. However, they are not open source, so we couldn't include them as part of our infrastructure because we couldn't customize or edit them. Moreover, the data would likely be stored elsewhere, reducing our sense of control, which is important to us.

Iguana is relatively new, so we weren't aware of it. One nice thing about Mirth is its FHIR plugins. I'm not sure if the other systems have this feature. FHIR will play a crucial role in the interoperability platform, and we need a visual method to handle it. I believe InterSystems is probably the largest competitor. I've spoken to some interface analysts I used to work with who use it. I think the primary difference is that it's a lot more expensive. So I think the difference in cost with Mirth and also the fact that it was more modular and open for our use case made it more logical for us.

Why did your company decide to move away from Mirth?

Yeah, so one of the main reasons we decided to adopt it was because of the availability of plugins that made it easier to build FHIR resources. However, most of these plugins and interface engines are designed for IT staff who are technical but not necessarily coders. That's where low-code environments come in. In our case, everyone on our team was a senior software engineer, so we had to come up with hacky workarounds to code within this no-code environment in order to use the visual plugins. Mirth, which is written in Java, uses a JavaScript interpreter in Java called Rhino, which allowed us to write JavaScript code within Java. We would export all the Mirth channels into XML and check it into Git, which made the developer experience quite long and tedious. Since we were all developers and wanted to create the most developer-friendly platform, this became a major reason for us to move away from Mirth. In our use case, we needed something more lightweight, and we had team members who were comfortable working with code instead of relying on a GUI. So, ultimately, we decided to build our own platform in-house by writing our own code.

What use cases do you use Mirth for now?

We didn't completely phase out Mirth. I was primarily focused on the FHIR API aspect, but we still kept it for other specific connections. There were some SOAP-based protocols or HL7 functionalities that were more specialized, and

we retained Mirth for those. However, since the majority of the traffic relied on the FHIR API, we migrated that part away from Mirth.

What does Mirth do across the different types of interfaces, with HL7 or other formats?

So, in Mirth, you can set up channels. Each channel consists of maybe five panes on a page, with each pane being a configuration. One of these panes will contain your input sources, which will have various predefined checkboxes. The middle pane is where you can write your code logic using JavaScript to perform transformations. The output pane allows you to specify where you want to send the transformed data. There are also other configuration options, such as sorting. An example of using Mirth for an HL7 use case would be when we expect HL7 messages from a REST endpoint. In this case, you would select the REST option in the input pane, write your JavaScript code, and then define the output destination, formatting, or any validation checks you need.

Are these resource conversions across systems, or are you also converting them across types?

You can do both. To perform an inner transform, you would specify another Mirth channel for the output. This means that the input would come through one channel, and the output would go to another channel, which would then perform additional actions. This is how we designed the architecture.

Regarding our FHIR API implementation, it was a bit cumbersome because you had to click through multiple panes to get where you wanted to go. However, it had a useful feature where you could quickly export the data into XML. They also provided APIs for this purpose. To make things easier, we wrote a script that allowed us to do a bulk export, so we could keep our GitHub repository in sync. Although the process wasn't the most user-friendly, we still had these options available to us.

Did Mirth do the actual transformation, or did they just output code that you had to run in order to perform the transformation?

Basically, Mirth just outputs code you can use, but what sets it apart is the scaffolding it provides. For instance, the primary FHIR plugin module utilized a dropdown list. This allowed for a visual method of constructing a FHIR resource. The generated JavaScript code was also visible. Even if you had no knowledge of FHIR, you could easily add the name field, which needed to be in an array. Mirth would handle this automatically. If you wished to add a new name and extract data from various sources, Mirth made it straightforward. While it was a low-code solution, it still involved working with code.

How would you characterize the relative strengths and weaknesses of Mirth Connect?

Yeah, the relative strength of Mirth is that it's open source. It has a lot of features, and there's a whole community around it. They even have a Mirth Slack where people are pretty engaged, so you can easily reach out for help. Plus, being open source allows you to try it out quickly. I'd say it's quite feature complete, and they've done a good job considering all the different aspects.

However, there are a few weaknesses. One is that it's a bit heavy and can be a bit dull to work with. The configuration involves a lot of checklists, and you have to click through multiple steps to get to what you need. Also, it logs everything, which is great for tracking, but it ends up causing the databases to become really big. You end up with a lot of sharded data, which can be a bit overwhelming.

One reason we moved away from Mirth was because of our containers setup. We were using a third-party service called Aptible to deploy containers, and each container had a limited amount of disk space. Mirth would often exceed the allocated disk space, mainly because it logs every transaction. When we added more channels and extra steps in Mirth, which may be a good software practice in general, in Mirth it adds numerous extra steps, so you're re-sharding the same data multiple times.

Could you lower the logging level in the configuration?

Yes, we lowered the output to the GUI, but there is a certain amount that needs to be stored for the transforms. There might have been a way to achieve lower logging, but it wasn't clear to us at the time. Since we had platinum support, I assume we explored that option.

Were there any stability issues with any of the transforms code?

No, but it was difficult to QA and debug. Once we got it running, it became very stable. However, the use of Rhino, which used JavaScript before ES6, and the incorporation of Java concepts created a challenge. Some of our developers were more familiar with Java, so they leaned toward those concepts. This resulted in a peculiar situation where reproducing the exact code was only possible if Mirth was running on the same Rhino instance, which is very specific. Consequently, some developers were tempted to ship code without thorough testing, making it harder to integrate into our CI/CD process and ensure proper testing.

How would you characterize the documentation and overall developer experience?

I think the documentation is acceptable. One positive aspect is the presence of an open community, which is great. However, the developer experience seems quite outdated and reminiscent of older enterprise systems. Additionally, the forums aren't very active, so it feels like a fairly niche field. I also had a less-than-stellar experience with some of the built-in modules. For instance, there was a module for HL7 V2 that I had to use, but it wasn't particularly useful because our new HL7 V2 implementation was a very specialized requirement. When I went through the development documentation for that module, it felt average. I didn't enjoy working with the transforms and felt that the documentation was more suited for IT staff rather than developers.

Do you have any tactical advice for other organizations that want to use Mirth Connect in their development process?

Yeah, use it minimally as a code environment. The useful thing is, you can do a lot of spot transforms and localize your interoperability code. For example, if you only need to modify the payload slightly for this specific connection, then it makes sense to keep that logic in one place and Mirth makes it easy to keep it close to the integration configuration. Keep the channels very specific. With Mirth, we were trying to make it our general FHIR platform. So it would take data from our model and transform it into FHIR. And of course, if the data model changes, we would have to update the FHIR API. It was a cumbersome process, since it was versioned, and we had to check it in to Git. Mirth, on the other hand, is excellent for customers who rarely make changes. We simply set it up once, they send us an XML blob, and we just have to modify those three fields before passing it along. So treat it as a minimal pass-through layer and carefully consider the logic you keep in the channel versus the infrastructure logic.

Was Mirth Connect the right decision for you?

Mirth Connect wasn't the right product for us, but it did help our team understand interoperability concepts. The problem was that we invested too much into it, and still rely on it for specific situations. Mirth Connect is really effective for some niche use cases. However, we made the mistake of trying to handle all our highly custom and frequently changing FHIR tasks as opposed to simpler FHIR tasks. As a result, we ended up getting trapped with a lot of complicated logic that was harder for software engineers to configure.

What areas for growth would you suggest for the open source developers at Mirth?

There might already be a way to specify channel configurations as code, but I immediately thought of using JSON instead of XML. It would be interesting to explore if XML is really necessary for everything, and if it could be made easier to read. Perhaps YAML can be used instead, as it is easier to parse through. Additionally, having shared repositories for common mappings would be a great idea. This is something that I believe should be implemented in general, although it may be challenging. For instance, if one person integrates with an organization for vaccine messages and writes code with no PHI, it would be ideal if they only had to do it once.

Do you have any advice for buyers who are thinking through their interoperability transformation strategy right now?

My understanding is that Mirth is designed for IT staff who possess technical skills but may not be trained developers. It can help them slow down and provide more guardrails. So, if you don't have a software engineering background, Mirth could be beneficial for you.

Try before you buy, because you can with Mirth. Additionally, think about the amount of code you would need to write in the platform. The runtime is quite specialized, making it difficult to test. However, in many cases, you won't need to write a large amount of code. If you do find yourself needing to write a lot, or if you're considering it, think about whether there are other places you can move your code to ensure your engineers are comfortable, or be very intentional about it and test it.