Razorleaf

THREE APPROACHES TO INTEGRATION THAT DELIVER GREATER PLM VALUE

INTRODUCTION

Most would agree that integrating two or more business systems together is a wise business decision. A solid integration will eliminate duplicate data entry, reduce errors, save time, lead to more accurate data and therefore improve productivity and the bottom line. Razorleaf recently sponsored an article on Engineering.com that talks about the reasons, challenges and solutions related to PLM system integrations (you can find that article here).

The value and reasons to integrate your PLM systems are established. However, this begs the question "How?" How do I integrate two (or more) disconnected systems? Are there multiple options? Are some options better than others? Are there commercial tools available in the market to help with this or is everything custom programming?

In this white paper, we will answer these questions related to the hows of integrating PLM with other systems to deliver greater value with your PLM system.

DETERMINING YOUR NEEDS WITH USE CASES

Long before you can choose what technology or approach you will use, you first need to determine what problem you are trying to solve. The answer might sound obvious, "I want to integrate my PLM system with my ERP (or CRM) system". However, the word "integrate" has broad meaning and is not detailed enough to establish your specific needs and therefore does not help to determine the best approach. Once you better define what "integrate" means to you, often, the best approach becomes obvious.

To illustrate, imagine you needed to move some packages from point A to point B. There are several options to do this. Throw the items in a backpack and get on your bike. Maybe the trunk of a car or your pickup truck will work? Possibly, a 53-foot tractor trailer is the answer. Everyone would agree that the difference between using a backpack and a bicycle versus a semi-truck are about as different as you could get. Will they both accomplish the task? Sure. Do they both have a similar cost structure and pros and cons? Not even close. So how do you determine which option is the best option? This is where your needs must be further defined. We call these use cases.

If the packages you need to move are four decks of playing cards and you need to move them two miles across a busy town the best approach is obvious. However, if the packages are the contents of a 2,000-square foot house and you need to move from Boston to Dallas a whole different method is needed. The point? Spend as much time as possible determining your integration use cases. Once you do, you will likely find that the best approach becomes obvious.

Here are some questions that will help to determine what your integration use cases might look like:

Is the integration single directional (i.e. PLM sends data to ERP) or is it bi-directional (i.e. PLM sends data to ERP and ERP also sends data to PLM)?

What is the timing of the integration?

- Real-time: the moment a PLM event happens the data must be sent to ERP without delay.
- Near real-time: a small delay, in the neighborhood of a few seconds to a minute, is acceptable, however the data must be transferred fairly quickly.
- Batch: the PLM data can be loaded into the ERP system once or twice a day.

What data needs to be integrated?
Just meta data or possibly documents and files?

Are there other parts of the business that have similar integration needs?

How do we do it today and what are the short comings of the current manual approach?

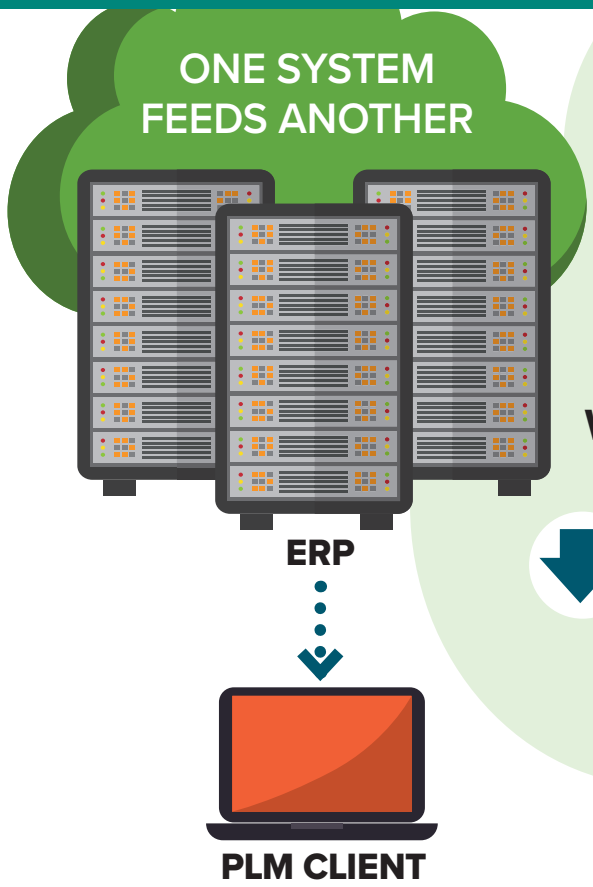
This list is not even close to an exhaustive list. It is just a few examples to get you thinking about what needs to be determined. To build your full set of use cases you need to study the interaction today — how do users do this manually today? What buttons do they click, what systems do they connect to, what data do they type, when do they do enter data, how often, what tends to go right, what tends to go wrong?

OVERVIEW OF OPTIONS

There are typically three common integration approaches. There are other approaches yet the three below represent the most common choices that we have seen over the years. For each approach, we will discuss 1.) technically, what is it and how does it work and 2.) what are a few pros and cons of the given approach.

Razorleaf

POINT TO POINT INTEGRATION



PROS

Makes real-time commands simpler

No need to install other programs

Writing of code is easy to accomplish

CONS

Issues when doing multiple tasks

Let's start by talking about the most common integration approach and that is a point-to-point solution. You classify a point-to-point approach as a solution where one system feeds another and the source system talks directly to the destination system – there is no hand-off, no middle-man, nothing in between the two systems. For example, imagine the scenario where you want to have your PLM system create a new material/item record in your ERP system when an item in the PLM system releases. You want to feed meta data from the PLM system, such as item number, description, make/buy code and unit of measure to your ERP system. You want this integration to happen whenever an item releases in the PLM system.

With a point-to-point approach, in the vast majority if not all cases, you would do custom programming. The custom programming would include some sort of event listener, something that fires on PLM release, and that would then gather the necessary PLM meta data, connect to the ERP system, log in, create the necessary ERP items and possibly report back to the PLM system with some sort of success or failure message. Generally, these connections will utilize a system's API or Web services.

PROS OF A POINT-TO-POINT INTEGRATION

Point-to-point integrations have some very clear advantages. A company may have subject matter experts and programmers on staff so writing the necessary code may be fairly easy to accomplish. Also, there is not much outside of the actual integration that needs to be accounted for. Since system A is talking directly to system B usually by means of an API or Web services the programmer does not have to account for any transport protocols (like message queues or FTP) or lower-level functions; they simply write the code to move data and the underlying infrastructure is already in place.

A second advantage of this integration approach is that it can make real-time communication much simpler. In a point-to-point solution, since the code that fires the integration is fired from inside of the application (i.e. on a PLM release) real-time is nearly built-in. The release happens, your code fires, gathers data and updates ERP — all without having to write something into a file or queue, all without having to fire an external service or application and wait for it to process.

Finally, deploying a point-to-point integration is usually much easier than other types of integrations. Most point-to-point integrations utilize some sort of scripting or add-in mechanism in the underlying program. There is not usually the need to install other programs or infrastructure components. In fact, many systems have built in deployment tools so that a customization or add in is automatically deployed

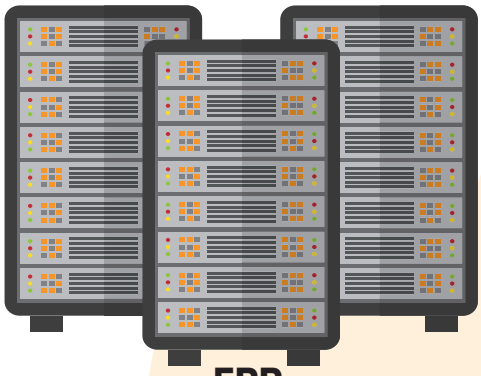
CONS OF A POINT-TO-POINT INTEGRATION

Point-to-point integrations accelerate when there is a single task that needs to be done. Point-to-point integrations start to have issues when there are multiple tasks that need done. In the example above, where an ERP item is created upon PLM release, the point-to-point integration works just fine. However, imagine the scenario changed slightly. In addition to updating the ERP system you too want to create a record in the CRM system so now you have multiple endpoints. With a point-to-point integration, you will, basically, write another integration – a second point-to-point integration.

Much of the logic and code you wrote the first time through will be duplicated. Of course, with good planning and programming you can maximize code reuse and have common methods and libraries however the fact remains that when it is all said and done you basically have two separate, disconnected integrations. Two separate, disconnected integrations that are doing nearly the same thing.

To expand on this scenario a little more, imagine that in addition to the PLM-ERP integration you also want to have a Sales Force Automation-ERP integration. Your SFA tool also needs to create items in ERP. When you wrote the PLM-ERP integration you already created logic and tools to create a record in ERP however, since this point-to-point integration is specific for your PLM system the SFA programming team will probably not be able to call your tools and leverage existing logic and work.

Razorleaf



ERP

PROS

Easier on PLM

Faster connection speeds

Makes it easier to implement communication

MIDDLEWARE BASED INTEGRATION

PLM CLIENT

CONS

Added costs associated with implementing

Can be cumbersome at times

Wikipedia describes middleware as “software glue”. It is defined as “middleware makes it easier for software developers to implement communication and input/output”. When an integration is written using middleware the programmer of system A does not need to know how to communicate with system B. Instead, he simply communicates with the middleware. The middleware already has (either out of the box or through custom programming) the tools to connect to system B. The middleware “glues” the two systems together even though the systems never communicate directly with each other.

In our PLM-ERP integration scenario we said, for a point-to-point integration, the PLM programmer would gather PLM data, establish a connection to ERP and then create the ERP item. The middleware-based approach is much different. The PLM programmer simply gathers PLM data and sends it to the middleware; at that point the PLM programmer’s job is done. It is now up to the middleware to establish the connection to the ERP system, log in, create the item, etc.

PROS OF A MIDDLEWARE-BASED INTEGRATION

A middleware-based integration allows the two systems to remain decoupled while still giving them the ability to communicate. In our PLM-ERP integration example, in a point-to-point world what happens if the ERP system is down or there is an error logging in? Since the PLM system connects directly to the ERP system either that situation will have to be accounted for and programmed around or the integration will fail with an ugly error message. This same connection via middleware would have very little consequence on the PLM system. Since the PLM system is simply dropping a message in a queue that says something like "hey middleware, PLM item 12345 just released", so long as the middleware is on-line and functioning the PLM system has no more work to do. Everything else is up to the middleware. In fact, most middleware systems will have a mechanism to retry failed records so if one side of the integration is down, without any effort on the programmer's part, the integration will continue once the down system is brought back on line.

Another clear benefit of a middleware-based integration is where the connection is not a 1:1 connection but a 1:2 or 1:many connection. With the point-to-point integration we talked about how, if you need to connect the PLM system to both ERP and CRM you will basically write two separate integrations. Not so with middleware. Remember, the source system does not talk to the destination system. It does not care if there is one destination system or 1000.

Lastly middleware-based integrations, due to their decoupled nature, tend to be great fits for integrations that have one or more long-running processes. Imagine the case where, in order to create an item in ERP a bunch of background ERP processes need to happen. It takes 30-60 seconds, or longer, to create the ERP item. In a point-to-point situation, the PLM system is paused, waiting, while the ERP system does its thing. With middleware, since the systems are not directly connected to each other, and since the only job of the PLM system is to put a message in a queue, the amount of time it takes the ERP system to operate is irrelevant to the PLM system.

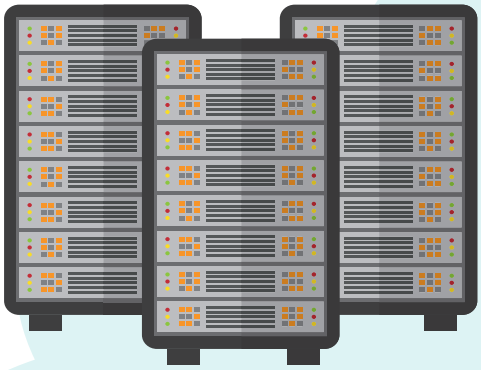
CONS OF A MIDDLEWARE-BASED INTEGRATION

An immediately obvious downside to a middleware-based approach is the fact that you need to purchase, implement and administer the middleware software. There are several commercially available middleware tools out there, even some free ones. Each one of these needs to be installed, configured, monitored, etc. Even if the license cost is free the day-to-day care and feeding of the tool is not. And, like most other software tools, the more robust and feature-filled the middleware tool, the more complex and harder to learn and administer it will be. That being said, many companies have an existing middleware infrastructure. Integration of systems is not new and certainly not unique to the product engineering/manufacturing/PLM world.

Middleware-based integrations can be cumbersome when real-time communication is necessary. Remember, with a middleware-based approach system A is putting a message in a queue and then, at some point in the future, the middleware system is picking up that message and processing (i.e. connecting to ERP and creating the item). There is an inherent delay in this approach. In many cases the delay is seconds. Yet, when real-time is necessary, a few second delay may not be acceptable.

Razorleaf

FEDERATED INTEGRATION



DATA IN A SINGLE LOCATION

PROS

Out-performs all other integrations when it comes to real-time

Zero risk of mismatched data

Lower costs because of less complex integration

CONS

Not all data lends itself to federated integration

Modifying the UI of your PLM system can be daunting

DISPLAYS IN ANOTHER LOCATION

In the IT world when you federate you separate. In the two integration scenarios above we discussed copying data from one system to another. When the integration is complete you end up with the same data in multiple systems. A federated integration is very different. Instead of moving or copying data you leave the data in its original location, a single location, and simply display the data in the other system.

To illustrate the use of a federated integration solution we need to modify our PLM-ERP scenario a bit. Federating data related to item numbers would probably not work as both systems need the item record to live in the respective system. Imagine though that the scenario was still a PLM-ERP integration, yet cost was data to be integrated. After an item is created in ERP the purchasing department will assign the cost. PLM users need to know the cost when they view the PLM item in the PLM system.

With a federated solution, instead of copying any data, the federated integration provides the means for the PLM system to retrieve the ERP data on demand. This may be done by means of a window into the ERP system from the PLM view or a button that displays a message with the current cost or some

other way. The point is, while the data is visible in the PLM system it doesn't really exist there.

One important factor to keep in mind about a federated integration, there is no magic way to get the data that needs to be displayed from the source system. There is still programming involved and underlying infrastructure. A federated integration is interesting as once you make the decision to integrate in a federated manner you next need to decide how. The answer may be a point-to-point federated solution or it may be a middleware-based federated solution. Many of the pros and cons previously discussed still apply.

PROS OF A FEDERATED INTEGRATION

A federated integration outperforms all other integrations when it comes to real-time. Since no data is actually moved the integration is always real-time — it is not possible to get out of sync. This is a great fit when the data is subject to frequent changes; things like price or quantity on hand. If you were to copy price or quantity on hand data from ERP to PLM you might end up with an integration that fires hundreds of time a day as these values change.

With federation, there is zero risk of mismatched data. Even in the most robust integration, if there is a hiccup, since the data is being copied, there is a possibility that the data in system A differs from system B. Depending on what that data is this could be a very expensive inconsistency. In federation that will never happen. A failure in a federated scenario might mean that the PLM user cannot access the cost; he clicks the button that is supposed to display the current cost and gets an error due to the ERP system being down or some other problem. While an error is not good, an error is better than getting the wrong cost (and not knowing it is wrong).

The cost of the integration may be less. Since no data is being moved often the integration is less complex and less expensive to write.

CONS OF A FEDERATED INTEGRATION

A federated integration assumes that the system that will display the data (the PLM system in our scenario) can display data from another system. Some systems have the ability to embed windows or controls in them where that window or control is connected to another system. In this case, the federated integration is a workable solution. However, if that is not the case, modifying the UI of your PLM system might be a daunting task.

As we previously discussed, not all data lends itself to integrating in a federated manner. In our cost example, imagine that the PLM system had built in methods that would calculate the cost of a product based on the quantity and cost of each item in the bill of material. Our items don't have a cost. They display a cost but that display is just a window into a separate system. If we try to run the costing routines on our PLM BOM it will fail as there is no cost — federated does not work here.

CONCLUSION

Options for PLM integrations are many and the pros and cons of each option differ greatly. However, this does not mean that choosing the appropriate integration infrastructure needs to be complicated. If you know your use cases and know them thoroughly, in many cases, the correct approach is obvious.

If you know you are trying to move the house contents of a family of four halfway across the country, you immediately know what transportation option is the best choice. We find that if you know for sure what your integration needs to accomplish the correct approach will likely be an obvious choice.

ABOUT RAZORLEAF

Founded in 2000, Razorleaf is dedicated to helping clients bridge the gap between PLM technologies and business problems to deliver greater value from their technology investments.

For more information about Razorleaf, call 330-676-0022.