Product Development Outsourcing

metacube
Trusted PDO

How can you ensure your team has both the skills and experience to deliver solutions

for your clients whilst minimising the risks and managing the cost overhead needed to

ensure your client gets the best results.

Introducing PDO

Product Development Outsourcing (PDO) is the outsourcing of the technical design, development, QA, delivery, support, maintenance and even the client implementation of a Software Product from a Product Company to a Software Development vendor, or the outsourcing of different elements of this process. We will cover this in more detail later in this document.

It is important to understand the relationship between Product Development and Application Development.

The fundamental difference is the mindset, while there are some other differences which we will discuss below.

Product Development

Product Development is the building of the product that a company sells to its clients. The roadmap of features and functions of the product, its user experience and its competitive edge are all of high importance.

Product Development is focussed on building products that customers want to buy and will keep on buying.

Application Development

Application Development is the building of software solutions to solve business process information issues for companies or to provide their clients with a shopping experience that facilitates and drives sales of their products and services.

Application Development does not produce a product that will be for sale.

What are the pot holes in the road to a good PDO experience?

1 Protect against governor limits

- Without this, the product wont pass SF stringent checks
- Enable to win more customers with greatest needs

Lets consider you have built an app that performs some kind of action on Opportunity records and for each Opportunity Product line associated with the Opportunity.

Using this app internally within an organisation you have far more control and are able to predict with fair accuracy that your app will not break governor limits. You could simply run a report or a short apex script to find the maximum number of Opportunity Product lines associated with an Opportunity. So long as your app can work this number of lines + some margin to accommodate growth for your company, you will most likely protect against governor limits.

However building an app for an ISV is quite different. You have to anticipate the worst case scenario of Opportunity Product lines for a future customer which currently you don't have.

So, do you set your limit to 20 lines, 50, 100 or higher still. What is worst the likelihood a company that has very big Opportunities is likely to be one of your largest customers and if they require 100 lines, whereas your limit is 50 you might lose this customer.

Whatever approach you take working within a multi tenant cloud environment there will always be a very maximum limit; however you can design in a way in order to accommodate most if not all realistic large Opportunities.

For example, 1 option to solving our problem would be to intelligently decide to process the Opportunity in real time or to use an asynchronous process to process the very large Opportunities. At run time you could first check if your Opportunity has less than 50 lines, if it does process the Opportunity in real time, if it is greater than 50 you could divert the processing to an asynchronous batch process.

As you can tell the amount of extra thought, design, development and testing to build an ISV app is far greater.

2 Quality code development

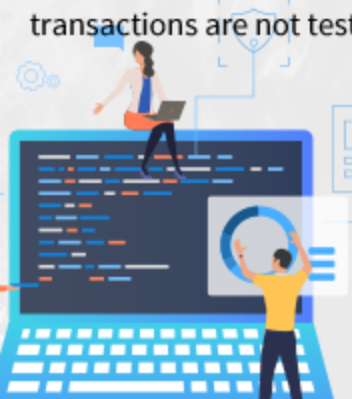
- Without this the product wont pass Salesforce stringent checks
- Baseline product is easier to expand

In our Opportunity example earlier we decided to build an additional batch process to handle large Opportunities. This is great because now you will be able to target those big clients. However, you realise that the batch process is failing when working with large Opportunities. You identify there are several problem.

First of all, the developer has added a SOQL within a for loop in each execute of the batch and sometimes it is failing.

Second issue is that a SOQL that filters specific Opportunities to process is passing too many records to the batch which is slowing down the processing and processing records that it shouldn't.

Moreover there are other smaller issues such as no error handling using try catch blocks; the code needs to be structured better because the developer has added over 1,000 lines of code into the batch class; the database transactions are not tested to ensure the running user has correct permissions to update records; a dynamic SOQL is not being checked for SOQL injection security issues; and there is no check to ensure the batch process is not being called by a previous future or batch process before the batch is launched which would cause an exception.



All of these issues are code quality improvements. Although these issues would be poor development and sometimes cause failures for an app built to work internally in a company, for an ISV app Salesforce would simply prevent you from launching your app on their marketplace.

3 Security

- **Without this the product wont pass SF stringent checks**

We have now developed a very good robust ISV app that can cater for almost any client of any size. But if we neglect the importance of security we will probably not pass Salesforce security checks, and if we do pass but we still have security issues and if one of our customers is attacked through a security hole you will likely lose that valuable customer, your reputation seriously damaged, lose other customers as a result of word-of-mouth. We did highlight 1 security issue in our code improvements earlier, a dynamic SOQL was not being checked for SOQL injection which could allow an attacker to change your customers data and even delete it. There are many security pot holes that you must fill to pass the Salesforce security checks before your app can feature on the Appex change marketplace and although it is wise to protect code developed solely to work in 1 production system there is no prevention by Salesforce stopping you deploying such code to your production instance. In the example of dynamic SOQLs, ISV apps usually employ this programmatic syntax widespread compared to application development for 1 Salesforce instance and so if not developed correctly will be more prone to security attacks.

4 Efficient stream lined code

- **ISV customers demand more responsive products, especially when they are paying for them**
- **Baseline product is easier to expand**

After all the code improvements we have made our app is now working very well with no exceptions being raised by governor limit breakages. But we could make our even better by making it work more efficiently, and by doing so our customers that demand a highly responsive product get what they expect. There are many ways or designing a system to achieve this.

For example, any callouts to external systems can slow down a system but if this processing can be done asynchronously allowing the user to continue working whilst the external system does its' work; or employing the appropriate use of batching and future methods; and even cleverly designed UX which gathers data and processes it in an interactive journey.

5 Using the right technology at the right time for the right purpose

- **Baseline product is easier to expand**

Many solutions can be developed to solve the same issue and although each often has pros and cons, and often there are obvious technologies to discount; it can be surprisingly difficult to select the correct technology to use for the requirement. For example, do use a trigger, process builder, visual flow or workflow when a record becomes updated; do use a custom setting or custom metadata to hold product configuration settings; do you store large data sets in a custom object, an external object or a Big Object; do use SOAP or REST for integration. The comparisons continue, but as you can see it is harder than you might first think. The affects of using perhaps the wrong technology when building an internal application is not as great; doing so for your customer using your app can be profound. You may be affecting and constraining your customer in ways you wouldn't expect by using an inappropriate technology for the requirement.

6 Flexible architecture enabling seamless growth

Enable to win more customers with specific needs

Fundamentally building the correct architecture from inception is vital to allow customers to expand their business, introducing lookups instead of master-detail relationships or vice versa, fields of wrong data types, or junction objects at the wrong hierarchy level etc can create a very confused and rigid architecture constraining your customers.

7 Stability in multiple Salesforce configurations and editions

Attracting any type of customer

Building an application for 1 Salesforce system is an enviable position. You know the Salesforce edition type of your system, you know the limits of all the Salesforce limits, you know all the bespoke configurations that have been built on the system that you need to consider when designing the new application.

When building an ISV app you don't know any of this. After careful planning and development our app is looking really good but each customer, especially the larger ones that you want to attract, often have highly customised their system.

This adds another dimension of complexity to designing our app.

Say a customer you really want to attract has built some validation rules on the Account object, or has added some mandatory custom fields. Because your app doesn't know anything about these things when inserting a new Account an exception is thrown. This and many other such bespoke configurations you need to be aware of and build a stable system that wont be affected. Moreover your customer may have a Salesforce edition that has lower limits, they may be using other Appexchange apps that could interfere with your app or directly affect, or other paid for Salesforce features causing unwanted side-effects.

All such scenarios and many more need to be considered, the app designed to unaffected, developed and tested.

Author

Steven Fouracre
steven.fourace@metacube.com
+44 (0) 1273 952913