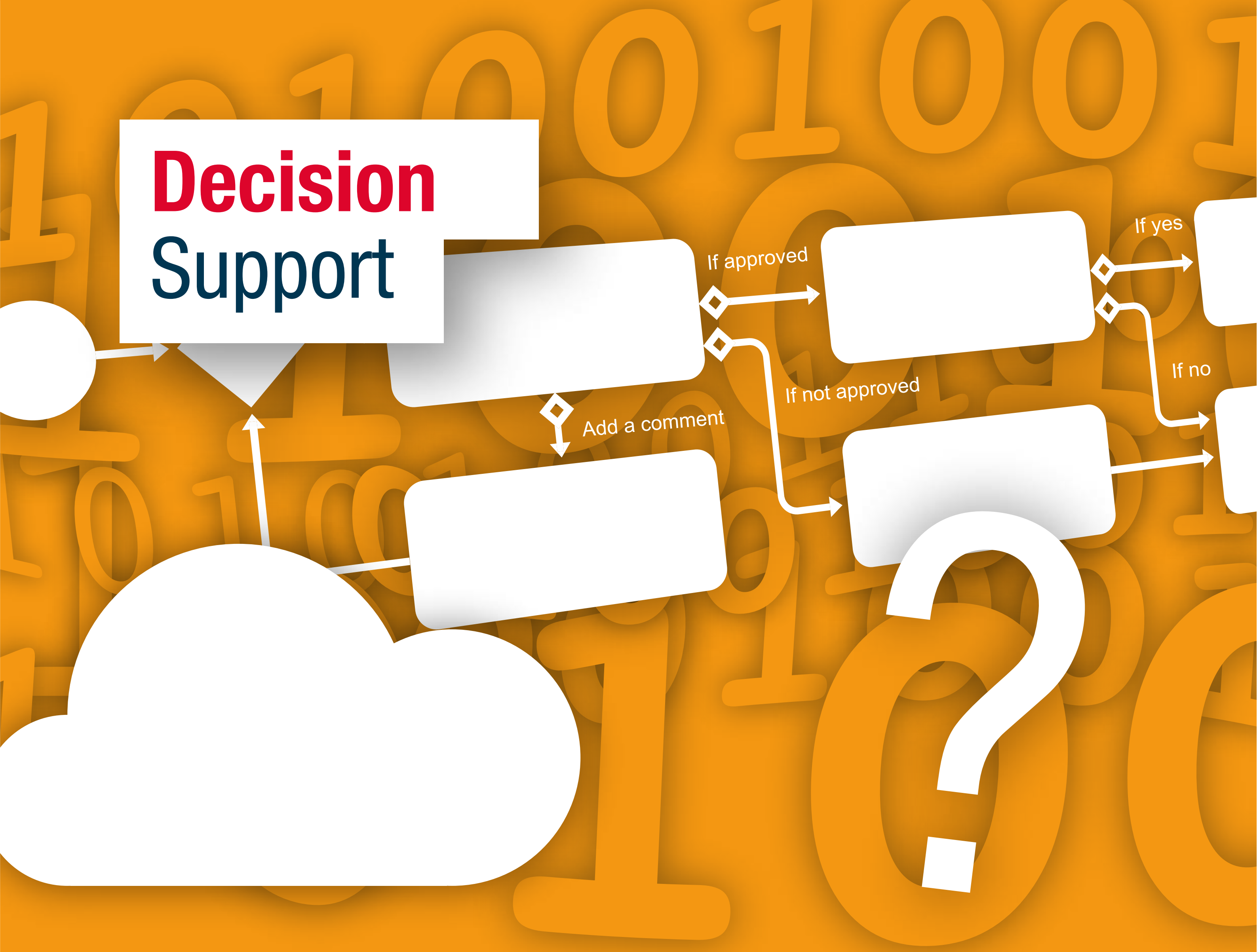
Decision Support



Custom Dev, BPM or SaaS?

Bonitasoft

Choosing among Custom Development, BPM and SaaS

What's best for your needs?

by Reint Jan Holterman, Publisher, BPM Leader

TABLE OF CONTENTS

Executive Summary

When starting a new software initiative, an organization first has to decide what development approach fits best with its objectives. Common development approaches are custom software development, BPM-based development and SaaS-based development.

Custom software development is defined as the process of creating software for some specific organization. **BPM-based development** consists of a set of techniques and tools for the continuous, iterative improvement of all the processes involved in running a business. **SaaS-based development** can be defined as a development approach where specific remote services are combined and invoked via the Web in order to create a new application.

Each development approach has its own characteristics, and its own strengths and drawbacks. An assessment framework can be useful in choosing which approach works best in which situations.

In this paper, we develop a framework based on two well-known models; one widely used in project management and the other for defining software quality. The framework combines the project management triangle of **Time – Scope – Cost** (with **Quality** in the middle) with the FURPS model for software quality. FURPS is an acronym for *Functionality*, *Usability, Reliability, Performance* and *Supportability*.

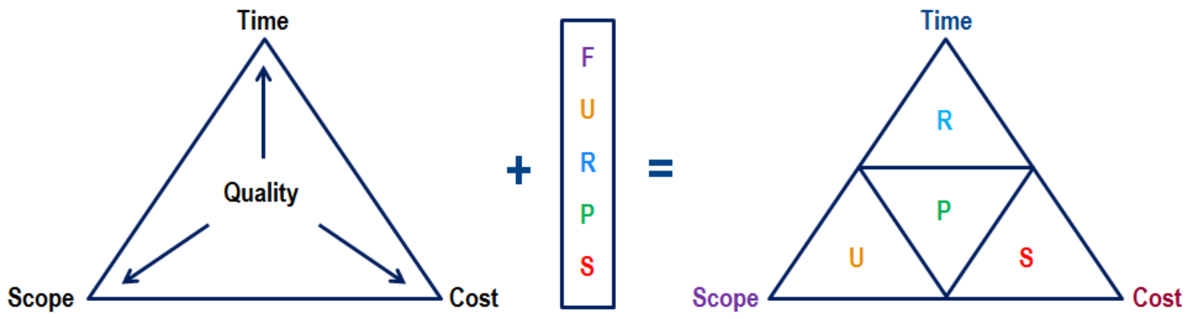


Figure 2. Assessment framework based on 2 models

The framework describes seven criteria for evaluating which development approach may fit best with your initiative. These seven criteria are:

- Time (to-market)
- Functional scope
- Cost
- Usability (Agility)
- Reliability
- Supportability
- Performance

At the end of this paper, a calculation tool is provided to support an objective decision for your development approach. The calculation sheet lists these seven criteria and illustrates a set of relative scores per criterion for each approach.

Bonitasoft

1. Putting Custom Software Development, BPM, and SaaS in Perspective

This section provides a short historical overview of the evolution in software development, with definitions for each one.

1.1. The Evolution in Software Development in Brief

From the earliest days of software creation, programmers have been thinking of ways to make their lives easier. At first, they copied-pasted lines of code, then quickly began to use instead sub-routines that could be called upon from another line of code -- for example, to do a specific task such as adding VAT to an invoiced amount or sorting an address list alphabetically. Sub-routines in turn were replaced by little executable programs that could be invoked with a set of parameters and that would return some specific value. These programs did not necessarily have to be written by the same programmer, as many software developers shared their work with others via bulletin board systems, floppy disks and (later also) via forums and e-mail.

1.2. Development workbenches

Over time, as software was used to solve more complex (business) problems, software routines also became increasingly more complex. Developers needed enhanced ways to design and administer what they were doing, what libraries and routines were used, how these needed to be called upon and so on. This led to the rise of so-called software development workbenches or IDEs that facilitated software design, specification, building, packaging, testing and deployment. These development tools took over a number of tasks that needed to be done before a software program could actually be used. Yet, once a program was deployed, no matter as local deployment or as a service, it was more or less fixed; i.e. any changes had to go through the entire cycle of coding, building, packaging, testing and deployment over and over again.

1.3. Definition: Custom Software Development

Custom software development is *the process of creating software for some specific organization or other user*.[1] The outcome is sometimes also referred to as *tailor-made software* or *bespoke software*. Custom software is different from packaged software, which is typically developed for a mass market instead of for a specific organization or user.

1.4. Composing code

The rigidity of classical software development led some to think of new concepts, where specific software routines (or services) could be composed together into a new working application, and where all underlying pieces of code remained where they are. So instead of deploying a new software

[1] http://en.wikipedia.org/wiki/Custom_software

package, they deployed a new model. This greatly improved the flexibility in software development, and soon was broadly adopted in the software industry. It also had another important side effect; modeling brought the "techy" world of software development much closer to the "world of business." Suddenly, business people could look at an abstract model of, say, a mortgage calculation, and provide feedback on this to the developer. The model served as *lingua franca* between software engineers and business engineers.

1.5. Definition: Business Process Management (BPM)[2]

The standardization authority OMG (Object Management Group) defines Business Process Management or BPM as *a set of techniques for the continuous, iterative improvement of all the processes involved in running a businesss.*[3] The focus of BPM is on how people within a business work together, with tools to support the documentation, analysis and automated execution of business processes. BPM provides us with a systematic approach to creating more effective and more efficient processes within an organization, with software tools that allow the organization to flexibly adapt to changes in the business environment.

1.6. Software deployed as services

About at the same time, as technology progressed, the ways to collaborate on software coding progressed, too. Developers, being early adopters of the World Wide Web, quickly found new ways to share software executables with each other. So for example, instead of sending files to each other, they opened up a web site where other programmers could on-the-fly invoke a specific service. This saved programmers from the efforts to incorporate pieces of code into their own deliverables. Moreover, it solved the problem of staying in-synch with the latest developments – no longer it was needed to modify distributables (i.e. packages of software code bundled together as one single deployable package) whenever one of the included components had changed. As long as the service still accepted the same set of parameters and returned a value that was expected by the requesting software program, it worked, no matter what changes were made under the hood, on either side.

1.7. Definition of SaaS-based development

Software-as-a-Service or SaaS is - first of all - a software delivery model. It describes the distribution of software applications, in which applications are being hosted by a service provider and are being shared over the Internet.[4] In most cases, users access the software application via their web browser. In its purest form, SaaS software delivery can be compared to the electricity distribution and consumption at home, where you plugin an appliance, instantly get the service requested (i.e. an electric current) and only pay for the kilowatts used.

[2] There is sometimes discussion about what the acronym 'BPM' stands for: business process modeling, business process monitoring or business process management. In this paper, we stick to the latter.

[3] http://www.omg.org/bpm/

[4] http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service

The concept of SaaS can also be used in software development. Saas-based development is *a development approach where specific remote services are combined and invoked via the Web in order to create a new application*. Sometimes this results in so-called *mashups*, lightweight applications that combine data, presentation and functionality from multiple services, creating a compound service.[5] This saves developers from writing their own routines for functionality that was already developed by another developer. In specific cases, a SaaS-development platform may be used to create new software, as e.g. the one made available by SalesForce[6].

Now that we have put the concepts of custom development, modeling processes and SaaS-based development in relation with each other, we will describe the characteristics and application of each development concept in more detail.

2. Custom Software Development

Sometimes, the temptation for an organization to build its own custom software package is strong and, in specific situations, the business case to do so can also be justified.

Probably the most-heard reason to invest in custom development projects is that "What we need is so specific, you can't buy it anywhere" quickly followed by "it is so crucial that we need to own the IP." Although these two reasons could also be filled in with a BPM approach, most IT managers first turn to custom coding as it is far more familiar to them.

2.1. Characteristics of custom software development

Custom software:

- Provides most flexibility in creating what you want, both in terms of functional requirements and in look-and-feel of the (end) user application.
- Requires software development skills that are widely available in the market, so starting a new project is relatively straight-forward.
- Provides full ownership of IP.
- Offers a poor time-to-market. Custom coding requires a lot of effort, no matter what methodology is used (classic waterfall, Agile/Scrum, Extreme Programming). Specifications have to be defined and made clear to everyone, the code has to be written and tested afterwards, pieces of code have to be integrated into one "build," deployment checks have to be carried out, and user acceptance tests have to be set up.
- Is generally more costly than BPM and SaaS-based development. Given that custom development is time-consuming, the costs are also relatively high.

[5] http://en.wikipedia.org/wiki/Software_as_a_service

[6] See for more information: http://www.salesforce.com/platform/overview/

- In addition to the point above, custom coding projects offer notoriously poor predictability in terms of time and costs involved.
- Maintenance is time-consuming and costly. Finishing a software development project does not mean that no more time or money needs to be spent; maintenance is an on-going evil that easily takes up around 20-25% annually of the initial investments made. In fact, this number can be as high as 75% if future modifications and improvements are accounted for as well.[7]

2.2. Example: Connecting hardware to software

A business need that might best benefit from a custom-developed solution is often based on requirements very specific to an individual company.

For example, some companies use an employee registration system for safety as well as for time registration purposes. These systems keep track of at what time each employee entered the building on a specific day, where they are inside that building (so that in the case of an emergency, all employees can quickly be evacuated) and through which door and at what time they checked out again.

More often than not, connecting this employee registration system with back-office systems like an ERP system or Building Safety System requires developing a piece of custom software. Various companies may be using different systems or want to keep track of different sets of parameters. This then leads to a tailor-made solution for connecting the registration system with other systems.

3. Business Process Management

The benefits of using BPM primarily lie in two things: in having more business agility and in bridging the gap between business and IT people. A BPM-based development approach helps you take into account future changes in the business environment, such as new tax regulations or alternative ways to buy your products (e.g. through online self-service). BPM fosters creating an adaptable, future-proof solution that you can easily adjust as soon as external changes occur.

Bridging the gap between business and IT is another advantage. This is especially important in environments where business demands fluctuate quickly or where it is extremely difficult to clearly describe the exact business requirements. Industries where this is the case are e.g. Banking & Insurance, and the Utilities sector.

[7] http://www.clarityincode.com/software-maintenance/

3.1. Characteristics of BPM-based Development

BPM-based development:

- Offers very a high level of business agility.
- Uses *composing* rather than *coding* to give the business analyst more control over what is being developed.
- Provides good transparency in how processes are actually taking place, which may lead to improved communications between business and IT people.
- Can be delivered relatively fast, as time-consuming functional and technical design phases can be reduced to one or more process modeling workshops in which business and IT people together take part.
- Has moderate maintenance costs, thanks to using process models that can be changed relatively quickly and easily.
- Provides full ownership of IP.

3.2. Example: Employee on-boarding

An example of a business need that would best benefit from a process-based application is *employee on-boarding*.

Employee on-boarding involves all activities needed to introduce your company to newly hired employees by explaining about culture, values, customers, products, procedures and benefit programs. The goal is to quickly ramp-up these new employees, in order to get them as informed and productive as quickly as possible.[8]

Typically, on-boarding is seen as a process, in which a number of steps need to be completed in a more or less fixed order by different people. First, employees get a welcome speech by a member of the board. Marketing explains about market drivers, typical use cases, unique selling points and benefits gained. Someone from HR explains more about how things work internally, about office locations, who to contact for what, pension schemes, health care insurance and possibly also about employee stock option plans. The new employees are asked to complete maybe a dozen or so forms, possibly hand in a copy of their ID and collect an employee number, a laptop and a company smart phone. Also, they can be automatically enrolled into a product overview training within the next two weeks.

Employee on-boarding can very well be managed (and modeled) as a business process. Output of one activity can be used as the input for the next. Different activities are performed by different actors (or roles) in the company. Specific information such as ID and laptop registration numbers need to be stored in various IT systems for HR, IT and Finance.

[8] http://www.bpmleader.com/2013/04/04/on-boarding-process-for-new-employees/

4. Software as a Service

Using SaaS in software development has two sides. On the one hand, it makes focusing on your own company's core strengths much easier, as you can quickly create a best-of-breed solution incorporating various software services. This saves time and ensures that your own developers are not distracted by creating functional services that are already available out there. On the other hand, using external (on-demand) services makes your solution very brittle, as you do not fully own nor control the services that are being invoked.

4.1. Characteristics of SaaS-based Development

SaaS-based development:

- Allows maximum focus on one's own core strengths. There is no need to "reinvent the wheel" as various (external) services can be invoked instead.
- Supports a best-of-breed approach, leading to more advanced and state-of-the-art solutions.
- Has relatively low maintenance costs, as various services are maintained by their respective owners.
- Has costs (of invoking external services) that are well predictable.
- Offers a highly scalable solution, as different services are executed by different servers.
- Risks creating a brittle solution, with limited control over when services are updated and when new features are being added.
- Makes you dependent on 3rd parties (i.e. poor ownership of IP). This covers to some extent the point above on 'brittleness', but goes further than that. The more crucial a piece of functionality that is used as-a-Service in your own solution, the more you are dependent on the provider of this service. Competitors know this, and can acquire the service (provider) just to thwart your plans.

4.2. Example: Data cleansing

An example of a business need that would best benefit from a SaaS solution is *data cleansing*.

Data cleansing is a service that provides you with higher-quality records about e.g. your customers or your employees. It involves, among other things, replacing inconsistent zip code / street name patterns with correct ones. This data cleansing is something you can do in bulk, for all your customer records in your CRM system for example. But it is also something you can very well do one-off; so as soon as a new customer record is to be inserted, first cleanse the data in this record and enrich it with additional details if possible. Typically, this is done through an online on-demand service where you can upload one record, instantly get a cleansed record back, in return for paying a few cents per cleansed record.

Using a SaaS-based development approach, a developer could extend the HR system to automatically cleanse the data for each new employee before this employee is registered. She then simply lets the HR system invoke the cleansing service, passing the details of the new employee immediately before inserting them, and then update the details with the response given by the data cleansing service.

5. Seven Criteria for Selecting your Development Approach

As we have seen in the previous sections, each development approach – custom development, BPM and SaaS – has its own strengths and drawbacks. In the remainder of this paper, we will provide a framework for choosing which approach works best in which situations. This framework is based on two well-known models; the one used in project management and the other for defining software quality. As we are looking for the best development approach for new software projects, it may be appropriate to combine best practices from project management with those from software quality.

Recall that the assessment framework looks like this:

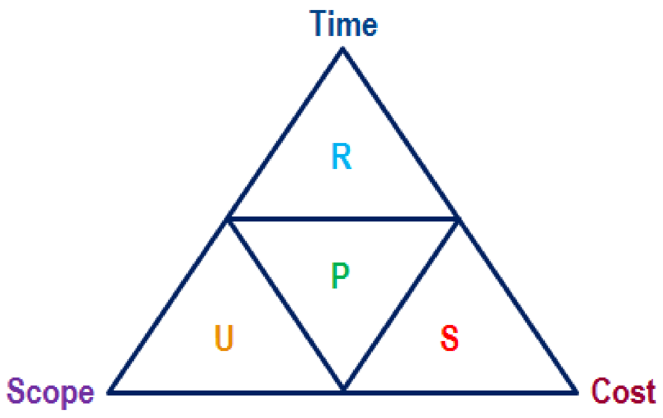


Figure 3. Framework for defining selection criteria

It combines the project management triangle[9] of Time – Scope – Cost (with Quality put in the center of the triangle) with the FURPS model as was developed by Hewlett-Packard in 1987[10]. Recall that FURPS is an acronym for *Functionality, Usability, Reliability, Performance* and *Supportability.*

We have substituted *F(unctionality)* with *Scope*, as these generally describe the same, i.e. what a project should deliver in terms of requirements. The *Quality* in the center of the triangle is replaced – or rather broken down into more detailed descriptions – by the "URPS" part of FURPS.

On purpose, we have put *R(eliability)* close to *Time, U(sability)* close to the functional *Scope*, and *S(upportability)* next to *Cost*. The P of *Performance* is put in the middle, as it equally relates to Time, Scope and Cost.

[9] http://en.wikipedia.org/wiki/Project_management_triangle

[10] Grady, Robert; Caswell, Deborah (1987). *Software Metrics: Establishing a Company-wide Program*. Prentice Hall.

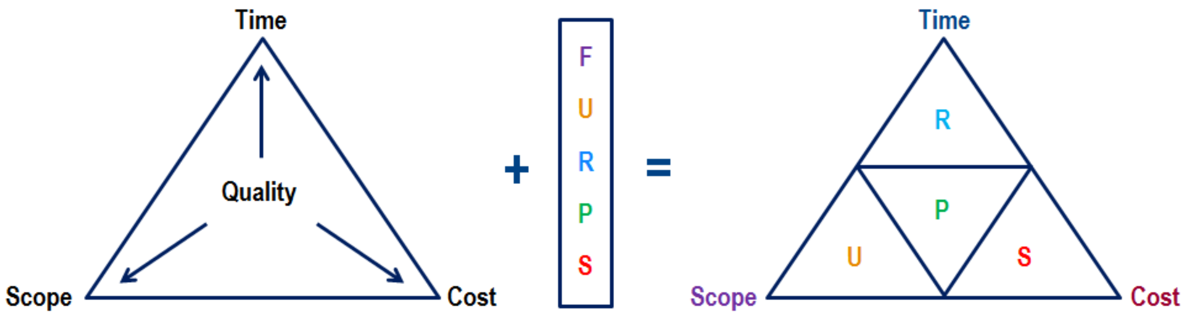


Figure 4.

In the remainder of this section, we will describe the seven characteristics of our assessment framework, and how they relate to custom development, BPM-based development and SaaS-based development.

5.1. Time (to-market)

More often than not, software development projects are under pressure to be delivered before a specific deadline. For marketers to get maximum exposure and good momentum, a new release is often linked to a specific occasion or event. Competitive pressure, legal obligations or having first-mover advantage can be other reasons.

Whatever the reason, time-to-market is an important factor in deciding what the approach will be. If plotted on a relative scale, you will find that custom development takes the most time, on average. As explained in the previous chapters, custom development requires writing functional specifications first, then coding, building and packaging the code, followed by testing and deploying it onto a server. This is a time-consuming series of activities, which means that this approach can be chosen if your time-to-market is long enough to realize your project. Typically, SaaS and BPM approaches will yield a quicker result, so are better suited for situations where projects need to be realized in a (very) short time-to-market.

If put on a relative scale[11], you will get something as shown below:

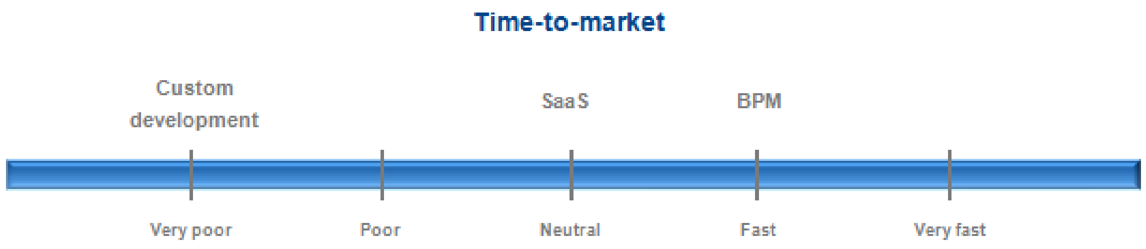


Figure 5. Relative comparisons for time-to-market

[11] This relative scale – and the other 6 scales in this chapter – are based on the author's own judgment and past experiences. It may be that in (your) specific situation(s) different rankings apply.

5.2. Scope

Software projects are always started to help fill in one or more business objectives. Objectives such as delivering better service to customers or reducing production costs can be realized with a self-service mobile app or a logistics application that reduces inventory stocks. No matter what the underlying business objectives are, the software requirements always should serve as a “translation” for what the software needs to do and what the business wants to achieve. There should be a clear alignment with the business objectives.

Unfortunately, a point that comes back in almost every software development project is (lack of) clarity of the requirements. The translation from business objectives into a set of functional specifications is a cumbersome task. More often than not, these requirements are formulated in a way that is not SMART – not *Specific, Measurable, Acceptable, Realistic* and *Time-bound*. The business analyst indicates that he wants a solution that is very intuitive to use – but how do you define what is “intuitive” in an objective way? Or the analyst says she wants the software product to be translatable – but into which languages? Does this incorporate multi-byte languages such as Chinese or Japanese as well? And does the solution need to support both languages that are written left-to-right as well as right-to-left, such as Hebrew and Arabic?

The more uncertainty and the more ambiguity there is in defining what exactly needs to be built, the more flexible an approach is required. Proper alignment with business objectives requires a flexible approach that leaves room for modifications while the development project is underway. Both BPM and SaaS-based development leave more room for modifications compared to a custom development project. Moreover, allowing the business users to specify requirements in the way they understand best – visually, as a series of activities that need to be carried out in a certain order, and everything formulated in business terminology – significantly contributes to getting better-quality requirements. BPM does this best, and is the least-technical approach of the three. That leads us to the following relative scale:

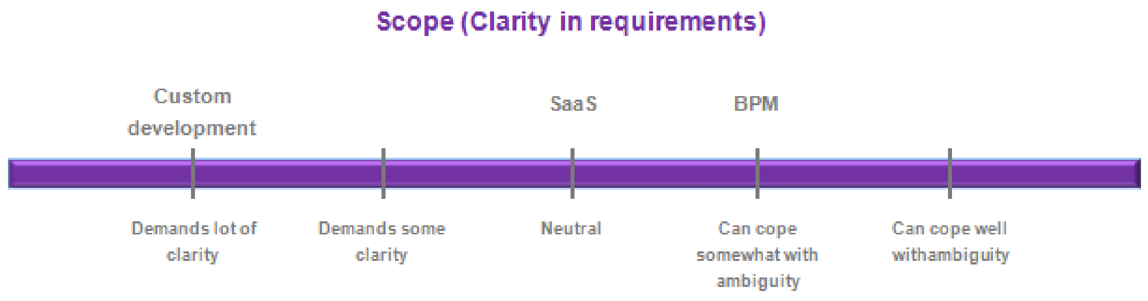

Scope (Clarity in requirements)
Custom development
SaaS
BPM
Demands lot of clarity
Demands some clarity
Neutral
Can cope somewhat with ambiguity
Can cope well with ambiguity

Figure 6. Relative comparisons for Scope

5.3. Cost

The costs of software development comprise more than just the number of hours spent developing times the average hourly rate. It consists of costs for workshops to get the requirements clear and prioritized, costs to get the necessary infrastructure (software and hardware) in place, costs of testing the solution, costs of documenting it properly, costs to maintain the solution once it has gone into production, costs to acquire or use specific 3rd party components, project management costs, legal costs and so on and so forth.

Although there are many different factors to take into consideration regarding the total cost of ownership (TCO) of a software project, the main cost drivers are usually the hours spent on creating the initial code, and the software maintenance. The latter, as we have seen, may occupy between 25% and 75% of the total project costs. This leads to the conclusion that the more readily available software components (services) are used, the lower the costs will be, as it saves on both development and maintenance time. On the other side of the spectrum, custom development will typically require most efforts to get all requirements documented, implemented, tested and rolled out. This leads us to the following relative scale:

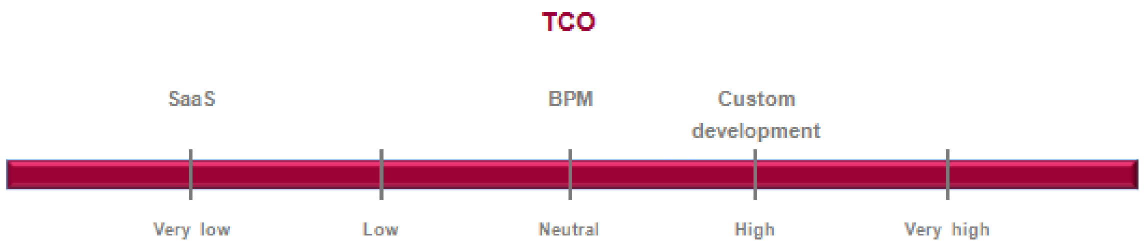


Figure 7. Relative comparison for Total Cost of Ownership

5.4. Usability

Software usability is often described as the look-and-feel of an application. However, ensuring it is usable in the sense that *it can be used* to complete a specific task is often much more important than how a software application looks. On the one hand, this is related to the aforementioned scope – whether the functional requirements align with the business objectives. On the other hand, it also means to what extent a software solution can rejuvenate itself to keep up with ever-changing business objectives. Usability then, to our opinion, is about how well an application suits with the original business goals for which it was developed.

To clarify this: If your software solution is dependent on quickly-changing market conditions, you'd better choose an agile approach. Suppose your solution incorporates tax regulations, import/export restrictions, legal conditions and the like. You can safely assume that these will change over time. The same goes for entirely new product-market combinations that are virtually untested and placed in the market without knowing if they are viable or not. These also require a development approach that is agile. Agile in this context means "being adaptable to future changes and requirements," in other words, the ability to modify your software solution after it has already been brought to market, to ensure it stays usable and still can serve its purpose.

Of course, agility can be built in into your solution using each of the three development approaches. However, you will find approaches where (business) people can model or configure the application afterwards to be more responsive to external changes – and hence be more agile – than solutions that do not offer this feature. BPM-based solutions in general show more agility towards externally imposed changes compared to custom developed (hard-coded) solutions[12]. The SaaS approach where different components can be incorporated or switched off again most of the times falls somewhere in-between, as the below relative scale shows:

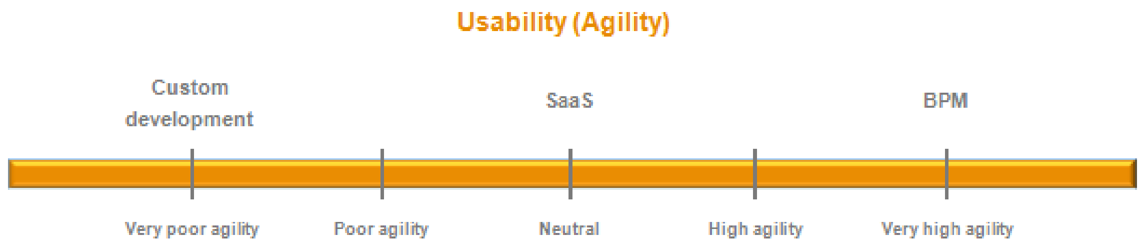


Figure 8. Relative comparison for Usability

5.5. Reliability

Reliability means that a software solution does what it was built for, and at the time you want it to be available to do so. It also implies that the outcomes of what it does are predictable.

Software reliability has a lot to do with how well it was architected, developed and tested. Reliability correlates positively with the skills and experience of the people who created the software, and negatively with the complexity of the solution.

So then, how does this relate to approaches such as custom development, BPM and SaaS? In general terms, one could say that using BPM, a lot of the lower-level groundwork and complexities have been abstracted from the modeler. So the modeler/composer no longer needs to know, for example, how to avoid null pointers in his code, as this has already been taken care of by the modeling environment. The same goes for SaaS projects, where the developer of a specific SaaS module has solved this type of coding challenges already for you. Yet, in SaaS you still need to make sure that different modules all work well together ensuring that the interfaces or contracts are well-defined and properly implemented. It is much less a visual drag-drop experience than BPM environments offer. SaaS-based development requires a more tech-savvy person than composing functional components together.[13]

Besides technical complexity, the infrastructure on which the application runs is important. Especially for SaaS-based solutions, if the remote services are not available due to network outage or because the interface changed, this has an immediate effect on the (perceived) reliability of your software

[12] Several Agile development methods have been introduced to solve this irresponsiveness in custom development projects, such as Scrum, XP and DSDM. Yet, these are mainly meant to make the scope of a project more controllable and predictable – the essence of hand-coding applications in itself remains rather in-agile

[13] It is recognized by the author that this may be an over-simplification of the real-life situation. However, by and large process modeling requires less technical skills than software coding, while integrating various SaaS components requires yet other skills that are more closely related to custom development than to BPM.

application. Solutions running entirely on self-controlled hardware, as can be the case for both BPM and custom created solutions, are much less brittle.

If we would combine both brittleness and technical complexity, we would see that SaaS-based development scores lower than custom development, which in turn would do worse than BPM-based development when it comes to reliability.

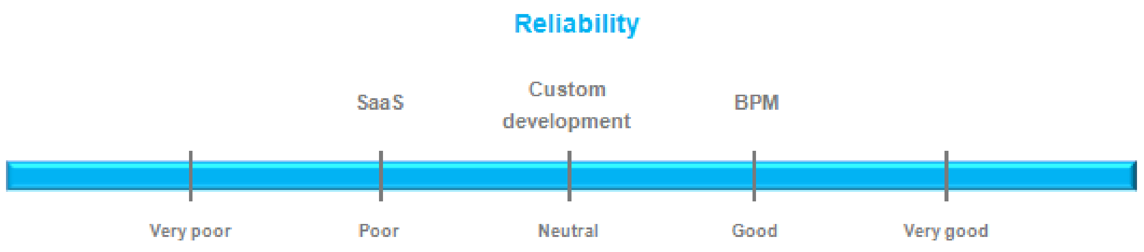


Figure 9. Relative comparison for Reliability

5.6. Supportability

Supportability of software relates to what extent the software can be configured and maintained. By nature, BPM-based solutions are highly configurable. Each step or activity in the process can be rearranged on the fly by the business analyst; and also the inner workings of each activity itself can be configured or changed. To a lesser extent, this also goes for Saas-based solutions, where different services can be connected to each other. However, one does not control the (internal) behavior of these components. Custom developed solutions, in turn, are least configurable, and their configurability is largely dependent on the way they have been architected and on the availability of skilled developers to make the modifications.

For maintainability, roughly speaking the same applies. In addition, it is good to realize that there is a major difference in maintaining models/activities (BPM) and maintaining external services (SaaS). For the latter, one is largely dependent on the priorities set by the external party (developer) who owns a specific service. This is much less the case when development – custom or using BPM - is done in-house. For these two in-house flavors, custom development will, generally speaking, consume the most efforts – i.e. coding, testing and deployment activities - to bring out a new version. This leads us to the following relative scale for supportability:

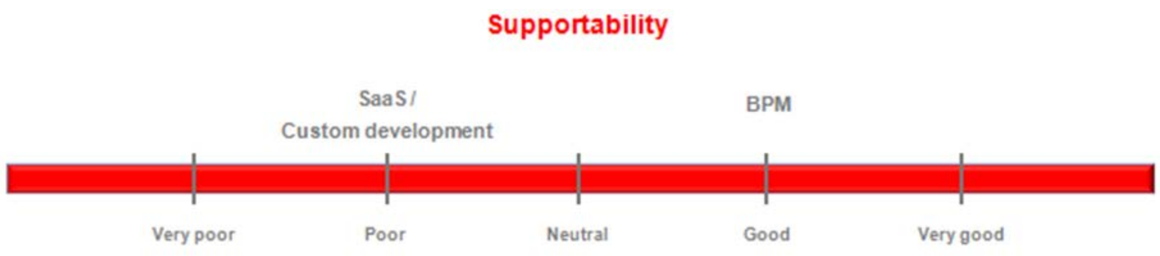


Figure 10. Relative comparison for Supportability

5.7. Performance

In the FURPS model, performance links to terms like "Speed, Efficiency, Resource Consumption (power, ram, cache, etc), Throughput, Capacity [and] Scalability"[14]. Although much depends on how a piece of software is constructed, regardless of the development approach chosen, some general statements can be made here.

First of all, there will be a difference in running an application in-house versus in the cloud. By nature, SaaS-based applications will run in the cloud, at least partly. Custom created and BPM-based solutions can be hosted externally, but could also be run on a server on-premise.

The performance (speed) of an application run in the cloud will always be negatively impacted by network latency as opposed to applications running in-house. Scalability, on the contrary, can be much better, as services can be deployed or "scaled out" across a range of servers somewhere in the cloud. For applications deployed in-house, sizing of servers, databases and the proper allocation of CPU time requires the skills of an experienced IT administrator.

Secondly, there will be a difference in performance for applications built directly onto "the bare metal" versus applications that are abstracted from underlying OS and hardware through a management interface. In other words, custom applications could be programmed using specific OS routines and hardware features that BPM-based applications could not use, as the BPM software does not cater for that. The restriction, however, of using specific OS routines is that it makes the software much less portable. It cannot be moved or upgraded easily to another OS or to another type of hardware. This approach may make the solution faster – it certainly also makes it more brittle.

Looking at these two factors, one could create the following relative scale for performance:

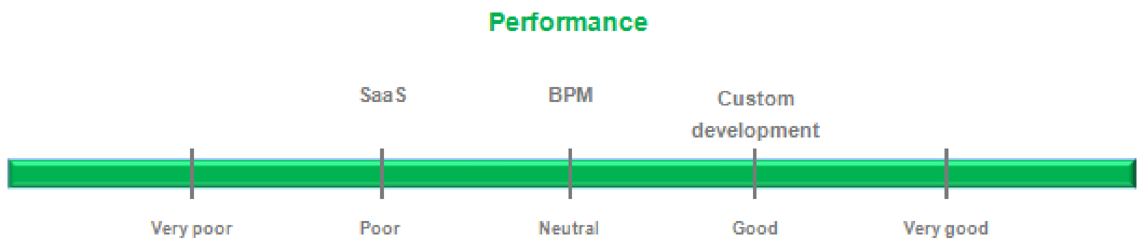


Figure 11. Relative comparison for Performance

5.8. Other factors

There are other factors one could take into account which were excluded from our model. These factors are left out for two reasons. Some represent a more detailed description of one of the seven criteria already described. Or, they may have less to do with the development project and with the quality of the product-to-be-delivered, and relate more with the organizational setting in which the project is embedded.

[14] http://en.wikipedia.org/wiki/FURPS

Nonetheless, we would like to mention some of them, as they can be useful in your discussions whether to opt for an approach based on custom development, BPM or SaaS.

- The need to have (full) **IP ownership**. This may be especially important in situations where you have to create software for military purposes or for high-tech production processes.
- The need to have (full) **data ownership**. This limits the use of SaaS in some scenarios where sensitive or personal data has to stay on-premise or safely behind corporate firewalls. These might include healthcare-related solutions and solutions for the financial industry.
- The access to and/or **involvement needed of the IT department**. For both SaaS and custom development, the dependency on the IT department is higher than for BPM (provided a BPM environment is already in place).
- The need to **measure outputs** of - and then optimize - business processes.
- The access to and/or **involvement of strategic business partners** who may have their own set of demands and checks that influence your own company’s decision.
- The access to **or availability of a BPM solution** with the organization.
- The **corporate culture**, which relates to *NIHS* – the Not Invented Here Syndrome, may hinder a SaaS-based approach.
- The level of **support from senior management** for whatever the approach you choose.

If you want to include one or more of these factors in your decision process, it may be useful to (first) create similar relative scales as we have done. These scales can be developed in a small group where people with knowledge about custom development, BPM-based development and/or SaaS-based development take place to discuss the relative rankings.

6. When Custom Development, BPM and SaaS Reinforce one another

In the sections above, we have made a distinction between custom development, BPM-based development and SaaS-based development. The assessment framework with the seven criteria can help make a more balanced decision about what approach would fit best with the objectives of your software project.

Yet, in reality you will often come across combined approaches, with one approach “in the lead” where other approaches help fill in some of the caveats. It is not uncommon to see a model-based approach, where some activities represent the invocation of external services, or where specific activities have been custom-developed and then wrapped as a building block for the process model.

Alternatively, it could be that an externally invoked service in reality consists of a larger application that was (partly) modeled as a process. However, as the inner workings are abstracted from the invoking party, this is not visible (in fact, it is also not relevant). A nice example in this respect is making a payment while ordering a product online. Payment of a product is just one step in the ordering process, and a step that is typically outsourced to a specialized online payment transaction provider. Invoking this external payment service may trigger a series of (micro) processes that a) validate your credit card, b) check if the amount to be paid fits within your card limit, c) deduct the amount from your limit, and d) give a response back to the main (ordering) process that the payment succeeded. All of these steps have to happen in a fraction of a second, and have to be completed as an autonomous transaction.

Nonetheless, when starting a new development project, it is always recommended to decide what will be your main development approach. In the next chapter, a calculation tool is presented that will help you make this decision in an objective way.

6.1. Evaluate Your Options: Run the numbers!

Choosing can be quite difficult, especially if there are many different factors impacting your decision. Some factors may be relatively straight-forward, while others may be more "soft" and multi-interpretable. This also applies to the factors related to choosing which approach may be the best match for your software development project.

A best practice often applied in these circumstances is to:

- define a list of criteria,
- assign a weight factor to each criterion, and then
- rank or score points for each one of them.

This makes the decision process a lot more objective, as people can have discussions over what factors are most important – and should therefore have the highest weight factors.

To facilitate this best practice, we have created a calculation tool. This tool lists all factors mentioned in the previous chapter. The *value per approach* columns are derived from the relative scales of the previous chapter – where the option most to the left is given a "1," the neutral value a "3," the one most to the right a "5," and so on.

To further illustrate the calculation tool, we have (only) filled in the weight factors for the *Employee on-boarding* example from Section 4. Multiplying these fictitious weight factors by the *value per approach* leads to a total value (per approach) for each criterion. We then add these values up, leading to a total per approach. In this example, the custom development approach gets 28 points, a BPM-based approach 76 points, and a SaaS-based approach 50 points. BPM gets the highest score, which indicates a recommendation for a BPM-based development approach.

Please note that in the below calculation sheet, we have (only) applied the seven main criteria that deal with the project variables time, cost and scope, complemented with the four criteria related to software quality. In some situations, you may want to add additional factors to your sheet. This is perfectly okay, as long as it helps you reach a more objective decision that is shared by all project stakeholders.

As a rule-of-thumb, it is recommended to keep the weight factors on a 5-point scale. This gives you enough flexibility to differentiate between criteria, yet also provides a 'neutral' weight for criteria that are somewhere in the middle. Weight factors typically are assessed by a group of stakeholders, e.g. during a workshop. Such a workshop helps to build a common ground for all stakeholders what the project is about, and leads to (more) buy-in from the stakeholders. A joint weight factor assessment also helps to make objectives as well as risks clear to all stakeholders.

Criterion	**Weight factor (Importance)**	**Value per approach[15] (Values are based on the relative scales of the previous chapter)**			**Totals per criterion**		
	"Employee On-Boarding"	**Custom**	**BPM**	**SaaS**	**Custom**	**BPM**	**SaaS**
Time-to-market	**3**	**1**	**4**	**3**	**3**	**12**	**9**
Functional scope	**4**	**1**	**4**	**3**	**4**	**16**	**12**
Costs involved[16]	**-3**	**-4**	**-3**	**-1**	**-12**	**-9**	**-3**
Usability/Agility	**4**	**1**	**5**	**3**	**4**	**20**	**12**
Reliability	**3**	**3**	**4**	**2**	**9**	**12**	**6**
Supportability	**4**	**2**	**4**	**2**	**8**	**16**	**8**
Performance	**3**	**4**	**3**	**2**	**12**	**9**	**6**
TOTAL	n/a	n/a			**28**	**76**	**50**

Table 1: Process Optimization Matrix

[15] These values correspond with the relative scales in the previous chapter. As indicated, the values are based on the author's own judgment and past experiences. It may be that in (your) specific situation(s) different rankings apply.

[16] The weight factor for 'Costs involved' is a negative number, as cost is a criterion that you typically intend to minimize.

About Bonitasoft

Bonitasoft is the leading provider of open source business process management (BPM) software. Created in 2009 by the founders of the original Bonita project, Bonitasoft is democratizing the use of BPM in companies of all sizes with an intuitive and powerful solution at an optimum cost.

Bonita BPM sets a new standard for BPM. It combines three solutions in one: an innovative process design studio that includes a rich set of connectors to integrate process applications to nearly any IT system; a fast, scalable service-based BPM engine; and a breakthrough, mobile end-user interface that allows people to fully manage both routine and unexpected process activities.

For more information: www.bonitasoft.com.

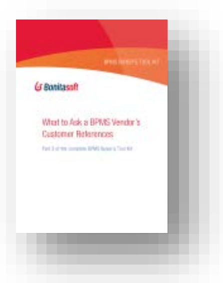
About BPM Leader

BPM Leader (www.bpmleader.com) is the largest independent community for business process management professionals worldwide. *BPM Leader* is the expert network and community site where you can find the latest insights, ideas and best practices on Business Process Management, Workflow Automation, Case Management, Lean Six Sigma, Change Management and related domains. A strictly independent knowledge sharing platform, *BPM Leader* brings together over 13,000 BPM professionals, bloggers, industry experts, users, vendors, consultants and analysts from all over the world.

Further reading

BPM Buyer's Kit
Sample RFP

BPM Buyer's Kit
What to Ask During a BPMS Demo



BPM Buyer's Kit
What to Ask Customer References

More information available at www.bonitasoft.com and at www.bpmleader.com.

HEADQUARTERS

GRENOBLE, FRANCE

32, rue Gustave Eiffel
38000 Grenoble

EMEA, ASIA & LATIN AMERICA

PARIS, FRANCE

73-77, rue de Sèvres
92100 Boulogne-Billancourt

NORTH AMERICA

SAN FRANCISCO, USA

51 Federal St. Suite 305
San Francisco, CA 94107

Bonitasoft

www.bonitasoft.com