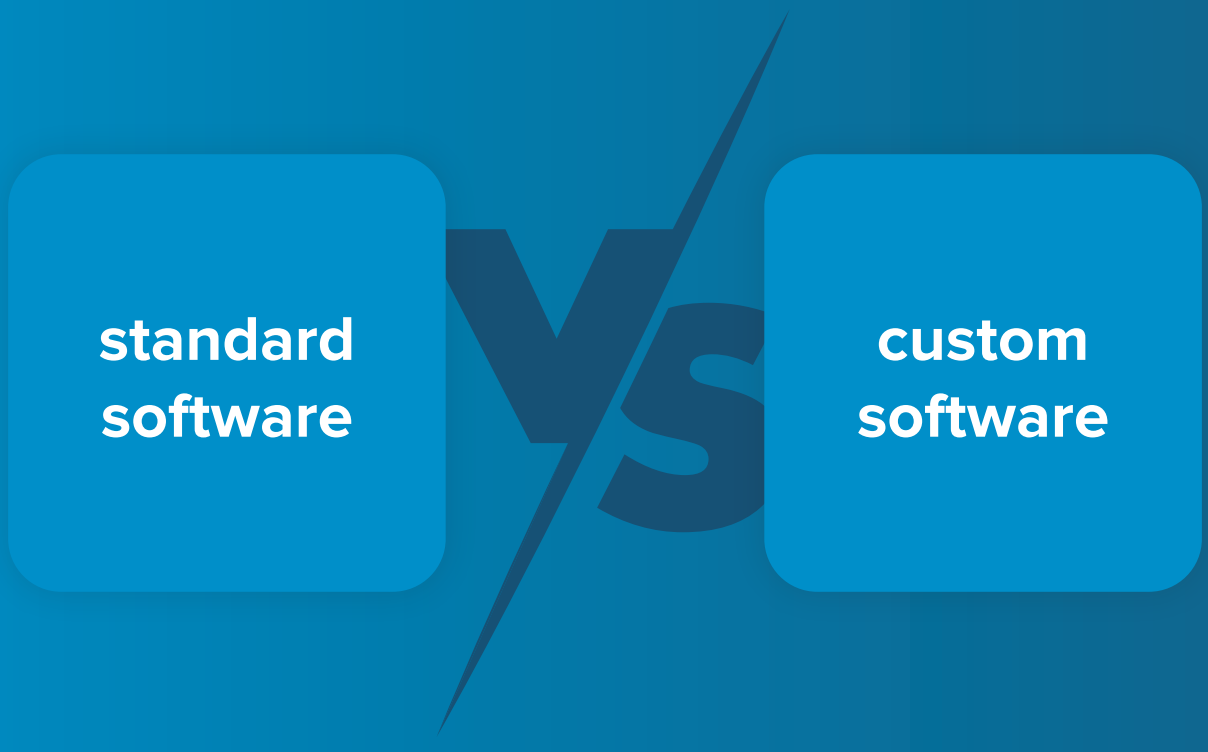
Coteng

software that pleasantly surprises



How to pick the most suitable software solution

coteng.com

Content

Introduction 2

Focus on the business process first 2
Appoint a decision maker with global knowledge of the company 2

Things to consider when evaluating standard software 3

What standard software solutions exist? 3
Check the requirements of all stakeholders/users 3
Pick a user-friendly solution 3
Adapt your business process to the software or vice versa 3
Check out the license model 4
Overkill of functions 4
Future costs 4
Future changes 5
Data access via interfaces (API's) 5
Trial and testing 5

How to evaluate custom software for your project 6

Start small and adjust gradually 6
Flexible architecture allows to innovate on the go 6
Check references of your developing partner 6
Own the source code 7
Design for all stakeholders 7
Long term recurring expenses 7

Conclusion 8

Introduction

In this paper we explain the differences between standard and custom-made software. We give you the pros and cons of both types based on our experience in this matter.

Our goal is to guide you to the best suitable solution for your business case.

Focus on the business process first

Before moving on to the items that can help you decide what software to pick for your particular project, you need to understand that software is only an enabler. Previous to implementing software, you should have a clear view on the **business processes**.

In a production department, processes are mostly well established. Otherwise, workers would not know what to do and this could result in variation of product.

In administrative departments though, processes can be less defined and work may be done by various people in different ways. For example, one sales person uses a spreadsheet to track prospects, another one uses no list at all. Companies that are very process driven perform better in automation because they are used to change and optimize processes.

It's wrong to expect that software will turn companies with poor process control into good process control. An expensive tennis racket never changed a bad tennis player into a good one either! Both standard and custom software projects will have a high chance of succeeding in an environment with good focus on the process.

Appoint a decision maker with global knowledge of the company

Another important factor is the availability of (a) **good decision maker(s)** in your organization. This person should not necessarily be technical. It is rather somebody who is process oriented and has a thorough understanding of the organization and its procedures.

These two extreme situations you'd better avoid:

1. (high level) managers making decisions without understanding the impact at lower level
2. low level workers giving too much attention to details that will not improve the success of the project

It's better not to outsource such a decision role to an external consultant. The ideal profile has decision power and must be able to understand and value all stakeholders. A good start is to let all stakeholders define what could improve their productivity. Next, you define with them where the quick wins are in terms of ROI or adaptation rates.

Things to consider when evaluating standard software

- What standard software solutions exist?
- Check the requirements of all stakeholders/users
- Pick a user friendly solution
- Adapt your business process to the software or vice versa
- Check out the license model
- Overkill of functions
- Future costs
- Future changes
- Data access via interfaces (API's)
- Trial and testing

What standard software solutions exist?

When there is a need to automate a process or task, always check first what is **available as standard software**. In many cases there may even be open source (= free) software available. Take some time to read a few reviews about existing packages. It will give you an idea what the differences are between different packages. At this stage ignore technical items, but focus on functionalities and user experience. Once you have a short list of potential candidates, evaluate them using the following guidelines.

Check the requirements of all stakeholders/ users

The first question you have to ask yourself is: does this package **support all my requirements**? The more stakeholders the more different demands they might have. One of the most common mistakes is to only focus on the requirements of the management. For example: marketing or management wants the sales team to use a CRM. They need reports and data.

If you want your project to become a success, make sure you also fulfill the demands of **the frequent users**. They will be the ones using the system day in day out. Therefore, the system should be a measurable help for them, not an additional burden. It needs to offer functionality that significantly improves their workflow.

Pick a user-friendly solution

However, functionality as such won't be enough. It should be implemented in a **user-friendly** way. For instance, when the same task has to be repeated fifty times a day, it will make a big difference if this can be done in two clicks on one screen instead of in four clicks on two screens. The system should support the business process the user is requested to follow. If not, the system will be perceived as a burden and adoption will be low. The best way to judge this is to let several frequent users play with the system (see also item on test version).

Adapt your business process to the software or vice versa

These days most standard software packages have settings that allow you to **adapt the system to your specific needs**. Vendors will claim their system can be configured to meet all requirements. It is important to understand whether the system meets your requirements with or without the help of external consultants.

If no external consultants are needed, examine if the system will also support future changes in your business process. If external consultants are necessary, you may consider **adapting your business process to the system**.

This can perfectly make sense but make sure to evaluate the **impact on efficiency** with frequent users. Efficiency might improve, but examples are also known where it reduces (even by 20%!). Remember the remark on two vs four clicks for the same task.

When the same task has to be repeated fifty times a day, it will make a big difference if this can be done in two clicks on one screen instead of in four clicks on two screens.

Unfortunately, the devil is in the details! Problems are often detected when it is too late. Most companies have similar processes, but when compared in detail, many turn out to be unique. This differentiates you from competition, and you certainly don't want to turn it into standard.

If you do decide to adapt your business process to the new system, make sure you foresee good **change management**. But be cautious, in many cases big software companies use change management to cover up poor user experience for their product. Good software sells itself. Have people ever needed change management to use Gmail, Facebook or Doodle?

Another option is to have **external consultants adapt the system to your needs** using the configuration options. In this case, it is important to estimate in advance that cost correctly and compare it to the cost of having custom software developed for you. Certainly, consultants can achieve a lot via configuration. In certain cases however, they may push the possibilities of the system to its limits in order to get all requirements fully covered. Often this goes at the expense of the user-friendliness as you try to do something the system was not really designed for. The last thing you want is to end up spending a lot of money on consulting (in addition to the license cost) for a system that sort of meets your requirements but is sub-optimal.

When adapting a standard software package to your requirements, make sure they are future proof. You do not want to re-do all adaptations because the next version of the software does no longer support them.

When adapting a standard software package to your requirements, make sure they are **future proof.** You do not want to re-do all adaptations because the next version of the software does no longer support them. Also make sure that adaptations do not negatively impact other modules. Mind that some modifications may not be supported by the API (interface) used to communicate with other systems you may want to connect to in the future (see also section on API's).

Check out the license model

Spend some time to review in-depth the **license model**. Most modern standard software comes as SAAS (Software As A Service) in the cloud. It is common that you pay per user per month or per year. This is very convenient for budget controllers and IT departments. The cost - at least for the current year – is very predictable, no worries about maintenance or updates.

In some companies however, the **high cost per user per year** may lead to a decrease in the number of users. Anyhow, this is always a bad idea. It results in inefficient situations and more human error. Take the extreme example where in a sales office all sales people should have access to the ERP system. In order to reduce license costs they appoint an admin person who collects all data from the sales people via spreadsheets and then puts these data into the system. This is not how automation should work! To keep a logical flow, data should be captured at the very moment it is generated. This way you can keep delays, human error and inefficiency as low as possible. In order to have a healthy process, systems need to be available to all people that can contribute or benefit from it.

If your software package will have a **large number of users**, it is meaningful to compare the total license cost over 5 years to the cost of custom development (including maintenance). This will certainly be a good exercise if you are not 100% confident all your requirements will be met by the system without extra configuration/adaptation effort (see the section above on requirements).

Overkill of functions

Consider that in most cases **not all functionality is required** from day one. In most digital transformation processes people will only use a fraction of the functionalities of a standard package, even at the entry-level version. Moreover, all these features may make the application less user-friendly for those who only require some basic functionality. Let's say you give users a screen with 20 fields on it, but they only use 5 of them.

Future costs

Although the SAAS model is very attractive, it also has some disadvantages. You have no control on cost in future years. If the supplier decides to increase the price, even with no extra functionality, there is little you can do. It is very common for SAAS companies to start with attractive pricing and increase their prices once they have locked in enough customers. Therefore, it is worth investigating upfront how much effort would be required if you decide later on to migrate to another system. Some suppliers allow easy data extraction, while others make it very hard to capture all your data and leave.

The more complicated it seems, the more people will be reluctant to use it. In some packages this is covered by giving different user interfaces to choose from, but it certainly is not a standard item. In addition, you also pay for what you do not use.

Future changes

Another question to ask yourself is what the impact could be on your organization of **future changes** to the system. In a SAAS setup you are not in control of how the system will evolve in time. It is known that SAAS systems continue to develop, driven by market requests or by the need to justify price increases. As a user you have to undergo these changes. Sometimes applications become overloaded with features that are rarely used. What if some of these changes have an unwanted impact on your organization or efficiency?

Innovation and continuous process optimization are key for successful businesses. Over time, it is not unlikely that one of your business teams comes up with **new requirements** for the system you use. These may already be supported by your standard system or not. As a user of a SAAS model (or any standard system by extension) you have no control on the evolution of the application. In the best scenario you can make suggestions to the supplier.

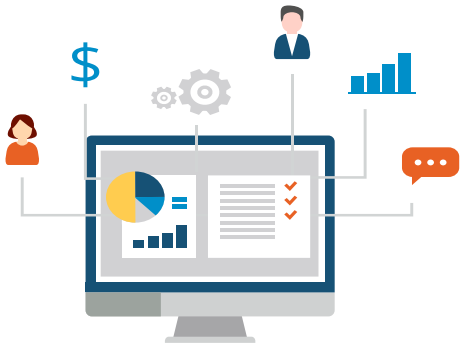
Innovation and continuous process optimization are key for successful businesses. Over time, it is not unlikely that one of your business teams comes up with new requirements for the system you use.

Data access via interfaces (API's)

In today's business environment software packages need to be able to **exchange data**. When considering a standard software package, make sure it has **well documented API's**.

Check in advance what data need to be available and how. Some applications only make a subset of data accessible via an API. So if you need certain data to be linked to other applications, make sure they are part of the API. In that context it is also useful to have some basic understanding of the structure of the data in order to judge if the outcome is compatible with your existing applications you want to connect to. Focus on how your business is organizing data in the existing systems to evaluate.

There is another reason why API's are important. They allow you to pick the best application for each task and connect them. In the past, standard application developers wanted to offer the 'full solution' as an integrated system. However, such an application is never 'the best' in each individual domain resulting in sub-optimal performance in most domains. Today, high performance API's and good master data management allow organizations to combine the best solutions for all domains.



Trial and testing

It is clear that there are a lot of things to consider. The best way to evaluate most of these items is to **test the application**. Most vendors of standard software applications have free versions available for evaluation. If you cannot test the application on your own without interference of consultants, alarms should be going off. If applications need extensive configuration or adaptation by specialists to cover your needs, it will be difficult to evaluate upfront. Unless the configurations can be done by one of your testers.

It is important to have the system evaluated by a group of people that represent all the different type of users. Achieving 100% satisfaction by all stakeholders will be a real challenge. Therefore it is important to assign a project leader or team capable of understanding which items are critical for success. This difficult role should not be given to an external consultant. Too often, decisions are taken top-down without understanding the impact. This will result in software projects that never deliver their return on investment.

How to evaluate custom software for your project

- Start small and adjust gradually
- Flexible architecture allows to innovate on the go
- Check references of your developing partner
- Own the source code
- Design for all stakeholders
- Long term recurring expenses

Start small and adjust gradually

When implementing custom software, avoid a big bang scenario. One, where you put all effort in writing long detailed specifications upfront, have them developed and finally tested at the end. Instead, **start small and fast**, expand as required. Determine what parts will provide short-term the biggest gain.

Go for a Minimal **Lovable** Product (most people talk about a Minimal Viable Product). This allows you to have quick wins in terms of efficiency and adoption. Make sure to have real users and have all different type of stakeholders work asap with a prototype. Do not wait until the end of the project to test. Real users will come up with good ideas during this type of testing. Often people will see **additional opportunities** to improve productivity once they start working with the tool. Add more functionality or modules over time, always keeping a good balance between cost and ROI/adoption rate. Keep in mind the role of the good decision maker as mentioned in our introduction.

It's very important to understand that investing in software is pointless if users do not like to work with it. Use the flexibility of custom software to provide a tool to your workers or your customers that improves their productivity or capabilities. **Good software requires less change management.**

Flexible architecture allows to innovate on the go

If you have software developed, make sure a **flexible and scalable architecture** is being used. For instance, when you are expecting substantial growth in users or data volume. Nowadays, requirements can change fast. Whenever correctly designed, changes and extensions to custom software do not have to be difficult or expensive.

This flexibility allows your company to continuously improve your performance to your customers and to be innovative. Too often, innovative ideas on the work floor are being blocked due to the inability of the system.

Be aware that poor architecture may lead to a system full of quick and dirty improvements that in the long run may become a burden. It is therefore important for the developers team to have a good understanding of the business. Only then, they will be able to understand what parts of the process can be variable in the future. This should be anticipated in the architecture.

Be aware that poor architecture may lead to a system full of quick and dirty improvements that in the long run may become a burden. It is therefore important for the developers team to have a good understanding of the business.

Check references of your developing partner

Before developing software, ask the partner for **references**. Developing custom software is a matter of people. The result will depend substantially on how well your team gets along with the vendor. Therefore, it is important to **understand the culture of the company** you will be working with. Check some references to get a clear view.

Mind that experience with a certain suppliers team of a large company may not be relevant for you if you will work with another team. Read over the talk of the sales person and try to get a thorough view on the persons you will be working with.

Avoid at this stage the technical talk. So called soft skills are as important as technical skills in order to capture your needs well and translate them into good software. Remember, you need to fulfill requirements of all stakeholders. This means for some projects the team may need to talk to the production operator as well as to the marketing manager. The goal must be to translate user input into productivity gaining tools or to enable innovation.

In order to realize the full potential of custom software, the development team must be willing to invest in understanding your business and uniqueness. Although it is always good to know what others do, be careful with people thinking they know your business better than you do. You don't want to lose your differentiation factor towards your competition, you want to be better!

Although it is always good to know what others do, be careful with people thinking they know your business better than you do. You don't want to lose your differentiation factor towards your competition, you want to be better!

It requires a strong basis of **trust** between you and the vendor to develop custom software successfully. The result will depend substantially on how well you cooperate. It will be difficult to built the optimal solution if each party is trying to maximize its own benefit all the time. The vendor who is trying to charge extra for every single change and you wanting to get more and more out of it for free. Good vendors ask a lot of questions before they make the price offer and indicate in advance what factors will have an impact on the price. They will also foresee enough room for **flexibility** during the project. Check with your reference person on the vendor's behavior in that context.

Own the source code

Request to be the owner of **the source code** and foresee a transfer clause in the contract. In such a clause you can specify what you expect the supplier to do in case you want to transfer the software to another supplier. For instance, you can agree on several days of support/ knowledge transfer at a particular price. Moving to another supplier is a major decision that should be well considered. It is at least a way out if the service wouldn't be satisfying anymore in the future.

Another big advantage of owning the source code is that you no longer have to worry about high license costs. The number of users is **flexible**. Add as many users as you want (collect data at the source) or use the software in multiple regions. Moreover, you can consider to give your customers the advantage of your application by letting them use it for free or have them pay for it.

Design for all stakeholders

As you can define all specifications from scratch, you have to be careful not to design the system to the wishes of a single, non-representative person or group. Other people might find that new way of working less optimal. Here the role of the appropriate **internal decision maker** comes up again. For instance, there can be competitive advantages in developing separate user screens or workflows in different countries. Use the flexibility of custom software at your advantage. And it shouldn't even be so expensive at all (see architecture). By appointing the right internal people to judge on the reason to develop custom software, you will invest well and avoid needless complexity. The more your company is process focused, the better the result.

Long term recurring expenses

When asking an offer for custom software development, also find out the long term recurring costs.

Take the hosting cost and maintenance/support cost into account.

For the **hosting** you can choose from several options. You can host the application on your own internal servers, on one of the commercial cloud providers or on the supplier's server. Beware that the cost of these options may vary a lot, even between the different commercial cloud providers. Most of them offer very good performance within the standard formula. The differences can come from capabilities that might be an overkill for your situation.

By definition, custom software should have a **low support/ maintenance cost**. For instance, if your application is intended for internal use by 100 to 1000 users and generates a little bit of data every day, maintenance contracts shouldn't be expensive. On the other hand, if your application will be used externally (eg by your customers) by 10k to 100k users and generates a lot of data, you may expect higher costs. In such a scenario, the cost per user should however still be very low.

Conclusion

Follow these steps to succeed in your next project

1. Appoint a capable internal **decision maker**

This person:

- understands and values the needs of ALL stakeholders
- has a thorough understanding of the company procedures and strategies
- is process-oriented

2. Start to **compare** the software options:

standard software

(+)

- Exists, Ready to use*
- Try before buy*
- Known cost*

(–)

- High license cost in case of many users
- May not be optimal for your organization
- May require consultants/cost to configure
- Not a way to differentiate or innovate
- Depend on supplier for future (cost & functionality)

*assuming no configuration/adaptation is required by externals

custom software

(+)

- Flexibility to optimize to your needs
- Only pay for what you really need
- Room for innovation
- In control for future

(–)

- Need to know what you want
- Higher initial cost (first 2-3 years)
- Can take 3 to 6 months to build
- Risk to over-customize

If a standard application exists that – according to ALL stakeholders –

does exactly what you need without unwanted overhead,

without **extra configuration or consultancy**

and the **license fee** isn't going to be a barrier for broad usage

USE IT!

In all other cases, the choice should be based on a thorough evaluation.

The decision maker will decide whether custom solutions will have a positive impact or the standard will do fine. In a world where companies are no longer only judged by the quality of product they sell, this may be important for your future.

EVALUATE