White Paper 2018-05

How to Develop Information Systems in an Effective Manner for Project-Driven Businesses

In a series of White Papers, based on our extensive experience in this area, we cover challenges of Information Systems in project-driven organisations: from the overall architecture to systems implementation, development and ongoing management. In this third White Paper we cover the specific challenges of developing bespoke Information Systems. After a first discussion on the choice between customising a commercial software and developing one's own, the key success factors of successful developments are investigated.

Make or Buy?

Some organizations prefer to develop their own software. We find that this choice becomes increasingly inappropriate for most project support-related software as specialised software in all sorts of areas becomes available from many start-ups and small companies around the world, that offer good service and ergonomics.

Recently most organizations have overcome the apprehension of having data in the cloud (including commercial data. This makes it easier to implement cloud-based solutions proposed by developers, avoids the potential issues of IT infrastructure and enables instant global deployment.

Buying a commercial software allows to share the development cost with others and benefit from improvements suggested by others, as well as benefit from general technological improvement. The drawback is of course that the software might not stick exactly to the process and terminology of the organisation, and that there might be an associated security risk associated (although this can be mitigated by using open source software).

Reasons for developing one's own software should be limited to the following cases:

- Security reasons (e.g. financial software for banks; high reliability applications),
- Confidentiality reason around a certain proprietary process,
- Specific functionalities that are too niche or too advanced for there to be commercially available solutions.

Strong communication between the future user and the developer is essential to ensure success of the project. Remote developers ("offshore") might look cheap but are generally a disaster.

Choose the adequate level of systems development

What is called systems development actually covers 3 different levels of development:

1. Level 1: full development from scratch using a software development framework
2. Level 2: development/configuration on the basis of an existing software platform that provides basic functionalities
3. Level 3: customisation of an existing software product by adding a limited number of functionalities or amending the data structure

The choice of the adequate level must be appropriate and consistent with the strategy. Evolving from level 1 to level 3, more development effort will be needed, the delay and the risk will be higher, and benefit from community development effort and updates will be lesser, and the influence on the product roadmap and development will also diminish.

The level 2 approach is often overlooked but can be an interesting compromise for a large number of fit-for-purpose solutions. It allows to have close-fitting solutions with a good hand in the way they are implemented without going into the pain of a full development from scratch. Depending on the need, a wide variety of platforms exist. For example, Microsoft SharePoint is a powerful platform, inexpensive with regard to the proposed functionalities, and that can be leveraged into useful products for project execution and collaboration at a limited cost. Other platforms exist for mobile applications that allow to focus the development on the process without having to develop the fundamental mobile synchronization and ergonomics framework.

If you want to influence the product roadmap, choose a smaller provider

For level 2 and level 3 approaches, it may be important to be able to influence the product roadmap to align with the business strategy. This is only possible if the provider is relatively small and the client has a relatively strong purchasing power. This carries other risks, as the provider might decide to pivot to some more promising products or disappear entirely. The right level of size for the provider must be analysed, and in some cases it may be appropriate to consider acquiring the provider to incorporate the capability and ensure its sustainable development.

Best practices in systems development

When advising clients that have decided to go for the product development route, we provide the following advice.

Have the project driven by the business

The development project should be driven by a project manager who knows the business and can easily communicate with future users. It is dangerous to leave such developments to the IT department.

Choose a local developer to enhance communication

In particular when developing an innovative product, strong communication between the future user and the developer is essential to ensure success of the project. Remote developers ("offshore") might look cheap but are generally a disaster in terms of delivery and functionalities because those specialised products cannot just be developed based on a specification. Therefore, even if the time-rate may look more expensive, it is always better to favour local developers, that can easily drop by in the office to have informal discussions and to get questions responded to.

While developing systems in-house may be an adequate solution in some cases, it is generally not recommended except in specific cases.

Ensure the developer has a robust project development framework

It is important to make sure that the developer is using a development framework and a development language that is foreseen to be maintained and developed in the future. Ensure the proposed software development framework is sustainable and is being actively developed, covering new areas such as recent mobile extensions.

The project management framework should be formalised and is generally close to the Agile approach so as to include feedback from the user in a quick manner.

Beware also of the usage of proprietary libraries, which mean that it will not be possible to hand-over the further development of the software to someone else, even if the source code is fully available, because of copyright issues.

Be aware that the productivity of developers can vary by orders of magnitude

We know of substantial software that have been developed by a very limited number of developers. The productivity and effectiveness of developers might vary by a factor of 10 or 20 and this is not so noticeable on the price. Therefore, plentiful teams are not necessarily a marker of reliable delivery. Get feedback from the market about the developer's reputation in solving tricky issues and about his productivity in general.

Ensure that non-regression checks are properly performed for any upgrades

Nothing is worse than upgrading a system and losing perfectly performing functions. The developer must have a thorough Software Assurance framework in place including non-regression checks for upgrades.

Do not underestimate the cost of ownership, maintenance and development

Companies generally tend to underestimate the cost of administration and maintenance of software systems, be they developed or off-the-shelf. For developed systems the user must in addition be weary of evolutions of the underlying technology.

Since the dynamics of technological evolution is so quick, it is important to understand what may become obsolete in the development approach in the next months or years. This requires an active monitoring of existing solutions and quick migration to new technology when they become available. Therefore, the cost of maintenance of a developed system (specifically for levels 1 & 2) may be substantially higher if there is a need to upgrade the underlying technology of the software.

Conclusion

While developing systems in-house may be an adequate solution in some cases, it is generally not recommended except in specific cases.

If this solution is chosen, the consequences in terms of additional cost for maintenance and upgrades must be understood. It is also essential to ensure that there is an ongoing and viable relationship with the developer to make sure that continuous improvements can be implemented as required by the business and to keep the underlying technological up to as technology evolves.

Reference

For recommendations on the implementation of commercial software White Paper, refer to our White Paper 2018-04 'How to Implement Information Systems in an Effective Manner in Project-Driven Organisations'

.