A Business Consulting Advisory Paper

Understanding The Impact Of ROI In Business System Decisions.

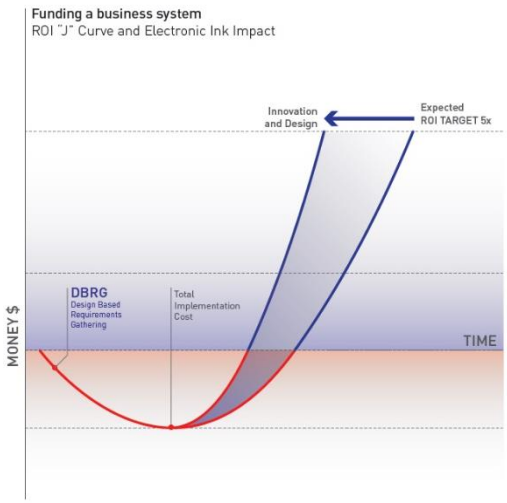
By Mitchell Chi
General Manager, Enterprise Software - America

Modern organizations rely on business systems to operate. In order to improve efficiency, businesses invest in technology to provide advantages. Technology investments have historically failed to deliver the promise of increased efficiency and maximum ROI. A recent study of 7,000 ERP implementations by the NorthPoint Group revealed the following; only 32.2% reached the ROI goals, user adoption was below 62.4%, 73.5% exceeded the approved funding, and 53.3% missed the go-live date.

For any other investment, these would be horrible results. For technology it's not only the norm, it's expected. Imagine buying a dozen eggs, finding four are rotten, and going back to buy a dozen more of the same brand from the same shop. This is what is happening with technology investments.

The question to ask is, “why do companies continue to approve funding for technology projects when the original return on investment is often missed?” To understand the answer, you need to first understand the process companies use to analyze and fund a technology-based business system. Aside from complex terminology such as hurdle rates, IRR, reverse trending, and the like, companies analyze funding an investment using a widely adopted financial tool called the "J Curve".

Understanding the three components of the J Curve will answer the "why". In the J Curve figure below, ROI is plotted on two axes: Time and Money. The initial cost expenditure is a negative cash outflow over the implementation period. This expenditure includes the actual software, any customization required, additional hardware and training. When the business system is deployed, cost savings are realized and continue until the initial expenditure plus financing cost is reached. This is the Breakeven Point. After Breakeven the cost savings needs to continue until the ROI target is reached for the implementation to be "successful".



The trajectory of the J Curve is critical to hitting the ROI target. Just like a rocket, adjusting the J curve, even slightly early in the trajectory, can have a dramatic effect on the final proximity to the target. Adjusting the J Curve late in the timeline is not only expensive, but frequently too late to hit the planned ROI. Making the right adjustments at the start of a project result in increased accuracy, reduced costs, and in many cases increased ROI.

The question is what adjustments should be made? There are three different patterns for business systems implementation, and we shall look at each one in turn.

1. Rush to Build – RISK #1

Technologists typically rush to build. They forgo true business research, and instead leap straight into the development. Technologists are prone to making assumptions that are not evidence-based, and do not stand up to scrutiny. Some of these assumptions are:

- That users are similar to themselves, and thus have similar preferences and mental models of their world;

- That since a solution worked for a similar previous project, it will work for the new project;
- That all that really matters is the data is present;
- That what the business tells them about processes and workflows bears any relationship to reality.

The "rush to build" pattern leads to systems that are not fit for purpose. Such systems are poorly designed with design structures that are inappropriate because they are based on incorrect assumptions. They often show very little internal consistency (this is especially a problem with Agile software development). While they may be usable, in the sense that it is possible for users to do their job with the system, they present many obstacles for the user.

Symptoms of a Rush to Build System

Users do not adopt the system. They complain that they do not understand how it works, they don't like how everything is structured, and they find the system present them with obstacles rather than really helping. They find that the system does not really reflect the way they work, and start to try to find workarounds. Frequently they abandon the system altogether because they find that it is not worth using.

Change Management is a partial solution, but why? Unlike software of the past, technology and software today must adapt to business needs without extensive user training, work-arounds and change management. *Would YOU buy a car if you needed to learn how to drive again?*

Business systems that have been built by technologists leaping into the build have several hallmarks. First, the implementation of these systems is not smooth, as there are many change requests. Business stakeholders become frustrated that what they are getting is not what they asked for, and not what they imagined. The technologists make profound mistakes about the design of the system. It is very common that they solve the wrong problem, or have fundamental misconceptions about the nature of the processes involved in the work of the business.

Visually, the system will be cluttered and confused. Design principles, if there are any at all, will be haphazardly applied, and different areas of the system will not be consistent with each other in aesthetic treatment, presentation structures, navigation, and behaviour of components. Often, such systems will be built around tabular structures which offer only limited advantages over keeping data in spread sheets.

Businesses who rush to build find that the users of a new system need extensive training in order to become proficient. Large training budgets are required, and it can be some weeks before users can demonstrate performance equivalent to what they were able to achieve on the previous system.

2. Traditional Requirements Gathering – RISK #2

Businesses often realize that rushing to build a system is counter-productive. In order to avoid the symptoms of rushing to build, they develop a complex system of requirements gathering and procurement. Businesses correctly identify that the problem with technologists is that they do not really understand the business. They solve this by introducing business analysts whose job is to gather the requirements.

Business analysts are sent out into the business with the brief to create a document that comprehensively describes the new system. To achieve this, the analysts technique to elicit the requirements is simply to ask the managers of business units what they want. Sometimes they also ask the users, and they record their findings in bullet points and complex process diagrams.

These requirements are ultimately transformed into highly technical schematics and prose which can run to thousands of pages. These requirements become the crucial document against which software vendors are measured.

A list of potential suppliers is drawn up, and wording of the documentation is adjusted to reflect the products that already exist in the marketplace. A large amount of legal work takes place to make sure that the RFP is worded satisfactorily.

Vendors respond to the document by effectively placing a tick against every piece of functionality that they can deliver. The vendor that satisfies the most requirements (over a certain threshold) is chosen, subject to cost. The business agrees with the vendor precisely what is going to be delivered in a finely worded contract. Once this is completed, the business is satisfied that they have conducted all the processes that will guarantee them a system that will generate a satisfactory ROI.

Symptoms of a System Built with Traditional Requirements Gathering

Businesses are often surprised that the implementation of their system is not smooth, despite all the groundwork that they have put in. They think they have a safety net, with the legal assurances and the finely defined requirements, but the system that is delivered is not satisfactory.

Despite the enormous requirements documents that the business and the vendor have agreed to deliver upon, the business still finds itself sanctioning change requests at extra cost. Early builds of sections of the software arrive, and they do not meet the users' needs. Looking through the requirements document, stakeholders note that one of their requirements is that the system is usable. The vendor, however insists that the system is usable, so long as it is used in the way they envisioned when they designed it.

Despite all the change requests, the final system is implemented, and immediately further problems are noticed. Users are not adopting the system, despite the fact that they were originally consulted by the business analysts on what they wanted.

Users still require expensive training courses, and take a while to become proficient at the new system. Even once they are trained, they still find the system does not fully support their workflows and processes. There are disconnects where they still find themselves sticking notes to the bottom of their screens to remind them of elements such as codes, passwords and navigation structures.

The J-Curve for Traditional Requirements Gathering.

The ROI for traditional requirements gathering is often disappointing. The return target is often extremely delayed. This is due to two factors:

1. **Change requests.** During the implementation, the business realizes that what they are getting is not what they need. The vendor argues that what they are delivering meets the requirements, and demand a further fee for implementing the change request. These change requests often have a domino effect, as they involve changing an "off-the-shelf"

product. When updates are made in the future, another change request will be required each time to maintain the customization in the updated software. This involves further cost.

2. **Theoretical efficiencies do not materialize**. The business analysts defined the current state structure and workflows before the project. From this they could create the requirements for a new system that would be more efficient because it would be faster and take fewer resources. In reality, users find that these effects are partially or totally cancelled out by barriers and obstacles that the new system presents to them. They leave their comfort zone of the previous system, and struggle to adapt to the new system.

Despite the vast effort that goes into creating an RFP and evaluating tenders, the process has not adequately defined the properties of the system. In some cases, deficiencies can be solved by change requests, but in other cases this is not possible due to technical constraints and cost. In essence, a lack of precision during the requirements gathering process results in massive budget overruns during implementation, and a large delay in the return on investment.

3. Design-based Requirements Gathering – Best Practices

Business has in the past responded by attempting to improve the components of the current system. One intervention is to improve the training of business analysts. This involves more in-depth education on business processes, a higher attention to detail, and the generation of more finely nuanced workflows. This solution is costly, as every business analyst will require more investment before he or she is ready to contribute to requirements gathering, and the increase in documentation will also require more labour during the process. Some benefit may be garnered from better-trained analysts, such as more consistent documentation, but the change request problem persists.

Other approaches have seen the adoption of agile processes for development. Agile is a process which breaks development of software into one or two week “sprints”. The product is divided into components, each of which can be developed during these sprints. The rationale for agile is that the business gets an earlier view of components of the system, with the benefit that components which are insufficient in design or function are fixed before the final delivery.

The disadvantage of agile technology is that it does not eliminate change requests. The same problem exists, namely that futile coding efforts that solve the wrong problem still occur. Agile merely distributes the blame more evenly between the coders and the business. Agile also has the disadvantage that it often leads to designs that are internally inconsistent. Since there Agile dictates that there should be no prior preparation and no documentation, overriding design principles that ensure a consistent and homogenous feel are often omitted.

One solution to the problem of insufficient precision of requirements is to create a new process for requirements gathering. *Design-based requirements gathering* posits that there are three key factors that make traditional requirements gathering is deficient:

1. A lack of, or insufficient consideration of the users;
2. A system of documentation which makes it very hard to describe what the product should be, and impossible to describe what it should not be;
3. Insufficient room for innovation.

The Most Important Component: The User

RFPs that are based on traditional requirements gathering processes are incomplete because they do not take into account the set of environmental, social, psychological and cultural variables that define the users' behaviour.

The worst possible approach to gathering requirements is to ask users what they want becausehere is a difference between wants and needs. It is *needs* that should form the requirements.

Users' needs are complex; so complex that they are in general unable to describe what they need, because they do not have either the vocabulary or the imagination to articulate a future state. Business managers are willing to offer their opinion of what the user needs, but their contribution is often worse because they suffer the same lack of articulation combined with a lack of familiarity with the minutiae of the users' everyday experience. Technologists similarly have a poor conception of the life of a user.

The tools to uncover what the user needs exist in the social sciences. A raft of qualitative research techniques can be used to determine latent (i.e. previously unknown) requirements. These include contextual inquiry, where the trained researcher sits with a sample of users and observes their work within the context of their environment, and interviews, where the researcher can use dialogue to map the users' mental model of their work. Such research can inform design that truly reflects the way the user sees the world.

Documentation

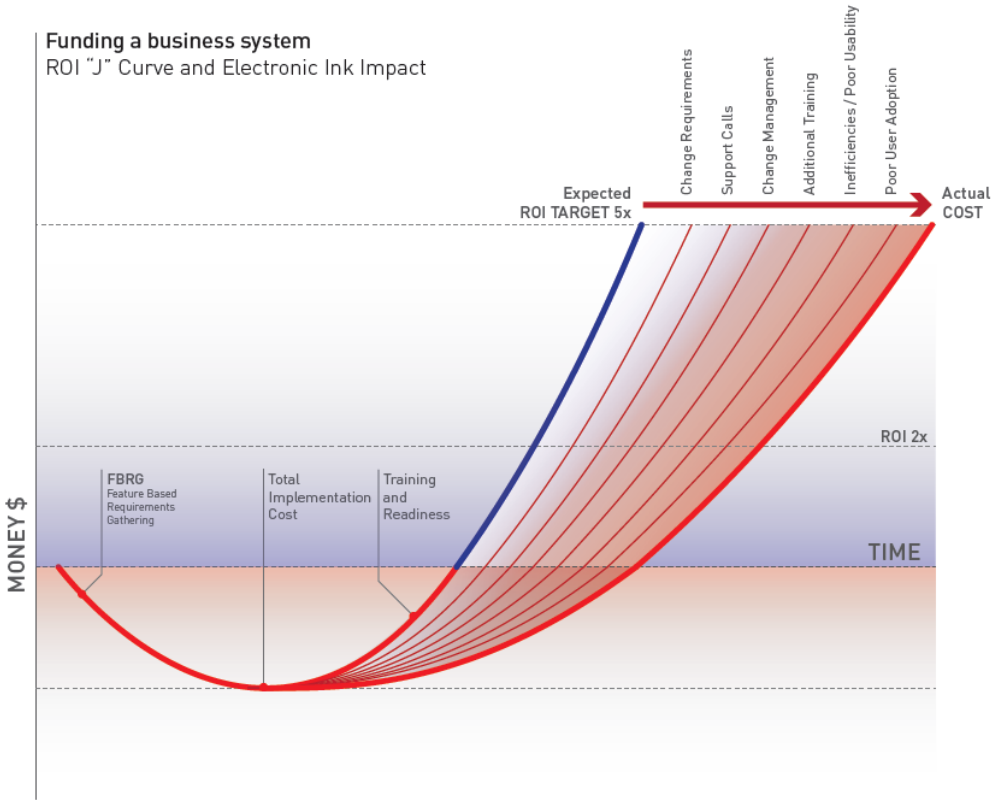
Design-based requirements gathering, as the name implies, uses design to define what a new system should look like. Expert designers use the same techniques as product and industrial designers to identify the problem that needs to be solved, and then explore how it could be solved. First creating an overall architectural view, and then successively increasing the granularity, designers can create a system that has a logical overall structure, and with components that are mutually consistent, and service the needs of the user. Evidence gathered by researchers informs the designers of latent requirements, and further validation of the designs can be gathered by testing at each iteration in the process.

The final result of the design-based documentation process is a set of specifications that design what the screens that are optimal for the user look like to a pixel-perfect level, how the screens behave in relation to the user, and their overall function.

The main benefit of bringing design to RFPs is that it takes vital design decisions out of the hands of those who really should not be making them: The developer. The developer has a very poor understanding of the user, and is very likely to have a hazy grasp of the business as well. Letting developers make these decisions is extremely dangerous for the success of the project.

These specifications also vastly simplify the process of grading responses to RFPs. Because they show the functionality, but also precisely define how the functionality will be presented to the user, the respondent to the RFP merely has to state what their system or capabilities cannot deliver.

In contrast to **design-based requirements gathering**, a requirements-based RFP will consist of a list of requirements. It is perfectly possible to create a system that meets all of the requirements, but is not actually usable at all. This is because it is impossible to define, in prose, all of the ways a system should *not* be designed.



Innovation

Businesses often pay lip service to innovation, but the traditional process for requirements gathering all but excludes the possibility of introducing it into business systems. Business analysts can make changes to processes and workflows, but these are based on poor quality evidence (if at all) and often bare little relationship to reality. Vendors of business systems often give the impression of innovation by re-skinning older applications with a shinier veneer, but this makes no difference to the underlying functionality, and its ability to support the user.

Design-based requirements gathering places innovation in its proper place in the implementation process. Before any code is written, it provides a low risk environment to create and test ideas. Workshops and collaborative design systems allow the user to feel that he or she has a stake in, and has made a contribution to, the design process.

The Design-Based Requirements Gathering J-Curve

The effect that design-based requirements gathering has on the software implementation J-curve are profound: Change requests are eliminated, because the design is already validated and highly detailed before a line of code is written, and a well-designed system is comprehensively adopted by the users, and improves their efficiency and effectiveness. Training budgets can be dramatically reduced because the system reflects the characteristics of its users, and the business' ability to deal with large amounts of data is improved.

The upturn of the J-curve is different: The upturn does not have any kinks in it, because there are no longer any change requests, and it is steeper because the improved design of the system has improved the ability of the users to do their jobs.

Conclusion

Traditional software implementations are risky, but that risk may not be obvious. The use of traditional documentation and processes to create RFPs has resulted in documents that may appear to be comprehensive and refined, but actually give considerable latitude for interpretation, and ignore certain types of requirement (especially those concerning humans). Waiting until after the software is procured to make key decisions about the design of software results in frustration, budget overruns, and sub-optimal final products.

Design-based requirements gathering can solve the challenge of defining the nature of a new product, bring focused innovation to bear on business processes, provide a deep insight into the real needs of the user, and provide comprehensive documentation that ensures the delivery of the optimal solution for the business.

About Mitchell Chi, CPA MBA

Mitchell Chi is General Manager of the Business Process Reengineering ERP practice at ICCG, Inc., a global technology firm. Since 1983, he has worked with many of the Fortune 500 companies and hundreds of fast growth start-ups in designing low-cost agile business systems to stay ahead of the competition. He has authored numerous white papers, been a keynote speaker at national conferences and continues to advise companies in building a "Strategic vision with tactical results".

Mitchell held executive positions with PriceWaterhouse, Goldman Sachs, Oracle, Infor and 11 successful start-ups. In 1991, the technology company he founded in 1983, MicroSage, won Small Business of the Year from the Small Business Administration. In 1990, MicroSage was awarded Minority Sub-contractor of the Year from the Department of Energy.

A dynamic and passionate speaker on staying relevant in today's economy by empowering technology.