The Problem with Tech Specs

An argument in support of Rapid Prototyping

The Problem

Out of decades of experience with software development, managers have come to realise the limitations of textual specifications in software projects.

With technical projects clients are generally unfamiliar with both the concepts and the terminology but also with the functionality that is being put to them. The pace of technological change in industry is such that experienced practitioners have trouble keeping up with it let alone professionals from another field.

The typical situation that occurs is that all the experts go through a lengthy and detailed design process and deliver a custom built solution tailor made to fit the client's requirements.

At this point the client looks at it and says : "That's not what I want!"

And the project manager says : "But that's what you asked for!"

There's been a fundamental breakdown in communications and no-one can understand why.

The problem stems from the fact that the project manager is an experienced technical practitioner and the client is not. The project manager is used to interpreting the specification and the client is not. When the project manager sees "a multi-function hierarchical picklist" in a spec they understand what it means. The client sees convincing techno-babble delivered by a competent technical expert and nods vigorously. Hence the problem.

It doesn't matter how good a spec you write or how much detail you put into your design documentation there will always be a necessary gulf of understanding between you and your client. This is called the "expectation gap".

The Solution

If client's aren't good at interpreting written specification documents, what can they understand ? How do you bridge the expectation gap ?

Client's *are* very good at taking something tangible and visualising how it is going to work in their own environment. After all who knows the environment better than they? They are an invaluable source of information about how your proposed solution is going to work in the context of their own environment.

But how do you provide them with something to assess ?

It is obviously infeasible and impractical to wait until the product is complete to ask them to evaluate it. By then it will be too late to implement any changes and you will be simply breeding dissatisfaction. The best way to resolve this is to use a prototype.

A prototype is a simplified version of the full product for design purposes. It can be a full-scale prototype with much of the same functionality or it can be a model, such as a drawing, diagram or simulation. It can be delivered in much less time and at much

less cost. Architects have long recognised the value of prototypes and models and use them extensively in the approval process for new projects.

By showing a prototype to the client which interprets their requirements in a tangible form you can much more accurately gauge how it deviates from those requirements. By using a visual, tangible form you are speaking a language the client understands and much less technical interpretation is necessary. Putting something tangible in front of them also reassures them about your capability to deliver a solution.

But that seems like a lot of work ? Surely investing time and effort in constructing prototypes could be better invested in delivering the real thing ? Not necessarily. If you consider that research shows that correcting errors during design is up to one thousand times less costly than correcting them in the field it makes sense to get your design right. This means that your time spent in getting the design right can be recouped later in the project when you don't have to correct mistakes made earlier in the lifecycle.

Also the essence of prototyping is to treat the prototypes as a design tool and not as final product. This means spending a minimal amount of time on them and generating prototypes that demonstrate a single design decision. By quickly producing, evaluating and refining your prototype you can correct your design and deliver what the client wants. This is the art of rapid-prototyping.

Rapid prototyping

The essence of rapid prototyping lies in speed.

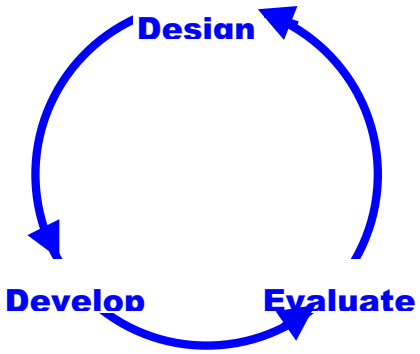
Prototyping is a resource intensive approach to the process of design and if it is to be of real use then that cost in resources must be minimised. Prototypes must be designed, developed and evaluated as rapidly as is possible. The strength in prototyping lies not in the prototypes themselves but in the rapid iteration of design ideas and the user's input into the process.

There is no time to refine or polish prototypes, they should be turned out in front of users as rapidly as possible. Prototypes can be constructed as actual systems and software products or they can be simulated with screenshots, mockups and even paper-based representations. The focus at this stage is not to build a final product but to decide what that product should look like and how it should act.

The Process of Rapid Prototyping

Rapid prototyping follows a process of design-develop-evaluate.

The emphasis however is very definitely on design. The development team spends time generating ideas on how to satisfy end-user requirements. These ideas are then *rapidly* developed into prototypes and put in front of users for evaluation. In later phases of production the emphasis will shift to each of the other phases in turn.



Any issues that arise during evaluation are then noted and fed back into the design process to generate new ideas and a new prototype for evaluation. A typical cycle should happen as rapidly as possible, preferably on the order of days to weeks and not months.

How long do you spend in this design ? As long as your schedule allows or until you have the approval of all your stakeholders and a consensus in your development team.

Software engineering sources typically propose spending up to 40% of overall development time on design and my own observations would support this figure. By eliminating as much uncertainty as possible before committing to the messy business of development you can streamline your production. Often the development phase introduces more people to the project team and hence more complexity in both communication and execution. Resolving questions about how to implement the software during design will save you time in the long run.

The Benefits of Rapid Prototyping

Responsive development – because the project goes through iterative phases of design, development and evaluation it is possible to adjust the design on-the-fly. By not committing to massive chunks of development without evaluation and redesign, the development team minimises the chance of misdirected time during development.

Rapid turn around – because design, development and evaluation are integrated into a tight control loop the chances of staying on track are much better. Large scale, linear projects have a horrible tendency to save up trouble for the final phases of the project and blow both schedule and budget only during the final phases of evaluation. By spreading this through the project it is possible to head off this problem and to resolve issues sooner rather than later.

Customer centric design – the use of prototypes allows the inclusion of customers or end-users in the design process. By producing and evaluating prototypes as the project progresses it is possible for the end-users to directly influence the design. Rather than being a black box process, development therefore becomes a much more open and flexible. It also begins to shape the expectations of end-users and customers as they see the unfolding design. In contrast the first understanding you have of user expectations during a traditional development cycle is at the launch when you might come to understand what is meant by a “big-bang” implementation.

The Pitfalls of Rapid Prototyping

Using the prototype as a basis for production - It must be realised that the prototype is not the product, nor even a precursor to the product. The prototype is a tool by which you make design decisions which are then implemented in the product. The danger of building prototypes into final product is that shortcut and approximations of the prototype are built into the final product along with the correct design elements. This means that what should be an inherently robust, well engineered product will be riddled with potential bugs. Prototypes should remain physically separate from final product.

Losing control of the cycle – because this process is inherently more flexible than a linear model like the waterfall model there is the risk of it running amok. Unless careful control is placed on the process it is possible to iterate round the circle of design, develop and evaluate ad infinitum. Without strong managerial control this model become unworkable because the synchronisation is very difficult.

There should be a distinct point in the schedule where the emphasis ships from design into development and the project transitions to the next phase. This point is usually determined by the project manager and the commercial pressures upon the project.

Lack of version control – because there is a tight cycle between the three phases things can move pretty fast. If your infrastructure is unable to cope with rapid iterations in design your specifications and prototypes may well become out of synch and it will not be clear where the product currently rests. This is also the point at which a prototype is in danger of becoming the basic of an actual product.

Lack of design and evaluation – there is also a tendency for development teams to dominate this model as in most other models. Careful balance must be kept between all teams to make sure that maximum benefit is gained from the model. If development teams dominate the process then it becomes less an exercise in design and more an extension of the development phase of the project. Developers should be (physically) restrained from writing core product code at this point since it will distract them from the design process and also possibly hijack the future direction of the project.

Summary

The purpose of design is to eliminate pitfalls, contradictions and complexity by resolving pivotal decisions before the commencement of development work. Accurate design also allows developers to proceed without constant referral to stakeholders or clients.

By using a technical specification or similar design document the project team attempts to outline how they will satisfy stakeholder requirements. This is used in conjunction with rapid prototyping to validate these decisions and further refine the design.

Rapid prototyping is the process of rapidly producing simple models of particular design elements to validate their design. The prototypes are then evaluated by stakeholders who's input is noted and the design modified accordingly. Strong management control must be maintained over the process so that design changes are properly incorporated into the final design of the project and that the design phase concludes on time. The design is then passed to the development team to implement.

For more discussions on Rapid Prototyping and the benefits of iterative development methods, see my other papers on the Iterative Project Model (IPM).