

Have We Finished Yet?



by
Suzanne Robertson
The Atlantic Systems Guild Ltd

In his keynote talk on dependable software at the 2005 Requirements Engineering conference, Daniel Jackson's urged us to "move away from infatuation with completeness". This started me wondering – how often does anyone ever finish anything? In our everyday lives we say we have finished cleaning the bathroom, cooking the dinner, watering the garden, ironing the shirt, cleaning our teeth and a myriad of other things. But we don't really mean that we have finished. Instead we mean that we do not have any more time to devote to that task and we have done the best that we can within that constraint. We accept the fact that we have limited time and that few of life's daily tasks are finished to one hundred per cent perfection. Why should building software systems be any different?

A common complaint of software developers is that they don't have enough time to finish a project. People in other professions for example engineers, architects, doctors, composers have the same problem but they have learnt to treat this as a normal constraint of their profession. They accept that:

- there will always be more to do than fits into the time available
- dynamics of the world mean that there will be changes that necessitate negotiation and replanning
- they need to be able to communicate their plans to their clients

This perspective is about how software developers can use these principles to free themselves from "infatuation with completeness".

Early Prioritisation

If you accept that you will never have enough time to finish any project then it makes sense, as early as possible, to have some way of deciding which parts of the project should be given the highest priority. So you need some consistent way of identifying the scope of the problem on which you are going to spend your time. A quick Stakeholder, Goals, Scope analysis is an effective way of getting started.

You analyse the goal so that you can quantify the reason for making the investment in the project. You need this understanding so that you can make the right decisions about what to spend your time on. The idea of the stakeholder analysis is to discover the stakeholders whose input is needed either from a business or technical point of view and to identify whether there are any gaps. Draw a context diagram to define the scope of your investigation. You identify the boundaries between the part of the world that is the source of your requirements and other people, organisations and computer systems. And the scope of the product, this will be fuzzy until you have done some of the investigation and identified the boundaries of the software you intend to build.



Using this Stakeholder, Goals, Scope checklist you can ask these questions at the start of your project. The results of this preliminary analysis provide you with these countable components that you can use as input to your first stab at quantifying the size of the project:

- Number of input data/material flows on the investigation context
- Number of output data/material flows on the investigation context
- Number of business events within your investigation
- Number of human systems adjacent to the investigation
- Number of automated systems adjacent to the investigation
- Number of stakeholders involved in the investigation
- Number of uncertain connections around your investigation

For example, suppose that you have done your quick Stakeholder, Goals, Scope analysis and have estimated that you need to investigate requirements for the responses (or business use cases) for 50 business events. And suppose that you have been given 25 business days to get the requirements. Then, very simplistically, you need to complete 2 business events per day. But all events are not of the same complexity or of the same priority.

So it would make sense to do a quick prioritisation of the events. Sort them by asking which events provide the greatest business benefit in keeping with the stated goals of the project? You can involve key stakeholders in this early prioritisation. Naturally different people will have different opinions and you will need to settle on the priority rating that most reflects the organisation's goal for the project. At this stage, depending on the number of people involved, the expected rate of change and the risk profile of the project you can decide your potential for agility.

For example can you take the top 5 business events and take them through the whole of the development process and deliver part of the working product very early? If this does not work in your environment then at least you can devote the early investigation to the highest priority events and use this as a way of providing early feedback to your client.

Dynamics and Change

In the words of Steve McMenamin “humans hate change”. We like to know where we are and what is going to happen. But nothing is more certain than change so, instead of wishing that it won't happen, it makes sense to expect

it and plan how you intend to deal with it. To do this you need to know what you are changing from.

If you have a project plan that is based on your initial estimates then anything that changes those estimates also necessitates a change to your plan. For example if there is a change in the law that causes a new input or output to your investigation context then that causes a new business event that in turn will need time to be expended on investigating, discovering, designing a solution and implementing its requirements. If your current plan has already allocated all the time that you have available then clearly something has got to give. Either you get some more time or you reduce the scope of the project. And the further you have progressed towards completion then the more impact the change is likely to have. Yes, this is all basic common sense. So why do I so often hear “we need to stop the business people making changes so that we can get on with building the product”. This is not going to happen in today’s world. The purpose of software systems is to provide a service that helps people to do work. If the work changes then the software also needs to change to support the new work.

No, I’m not advocating uncontrolled change – quite the opposite. Instead take control early, as discussed earlier, by doing an early and communicable analysis of what you are intending to build. Whenever there is a change to your initial Scope, Goals, Stakeholders then: analyse that change as a separate mini project, assess the impact of the change on what you have already done, communicate the impact along with the options for dealing with it, integrate the change into the scope and make the agreed changes to the project plan.

Continuous Communication

The responsibility for building relevant software systems does not lie solely with the systems analysts and software developers. The business people have a responsibility to provide the business requirements and to provide input that results in software that delivers business value. However most business people are not good at knowing articulating their requirements nor are they good at prioritising their requirements. This is not really surprising, as they are not trained as analysts or developers. In order to involve business people in discovering requirements and providing feedback throughout the

development process you need to be able to talk in a language that reflects the business subject matter.

Let's return to the idea of doing a fast, early analysis of the Stakeholder, Goals, Scope. By doing this you are expressing the project in terms of the people who need to be involved, the reason the business is making the investment and work done by the business that will be assisted by the product. You can talk to business people in terms of the business response to each business event. You can illustrate how the investigation scope contains all the business event responses. You can communicate which stakeholders need to be involved in which of the business event responses. You can identify atomic functional and non-functional requirements and review them with the appropriate stakeholders. All of this is using the language of the work. When you understand enough about a business event response then you can design the most beneficial product boundary and translate the requirements inside the product into: product use cases, modules, code, tables, pointers or whatever other building blocks you are using for your software. To be able to deal with change, you maintain traceability between the software building blocks and the business requirements.

Finishment?

Eventually the time allocated to your project runs out. Have you finished? Probably not, but you can say what you have done to what degree of detail. You can say something like: "We have implemented software to support these 35 business events and we can show you precisely which business requirements are supported by which pieces of software and what business value is being provided. The current plan is to add the remaining 15 business events to the second version of the software when the funding is approved."

If you make the problem comprehensible early then you can make it clear that you will not finish everything in the allocated time because there will always be uncertainty, new ideas and change. However, with the help of the business people/subject matter experts you can make a plan that focuses on finishing the parts of the software solution that provide the highest business value.

For more on prioritisation techniques and Stakeholder, Goals, Scope:
<http://www.volere.co.uk>

Mastering the Requirements Process by James Robertson and Suzanne Robertson, Addison Wesley. Edition 2, 2006
Requirements-Led Project Management by James Robertson and Suzanne Robertson, Addison Wesley 2005.

Suzanne Robertson is a principal and founder of The Atlantic Systems Guild
<http://www.systemsguild.com>
and joint originator of the **Volere** requirements techniques.
You can contact her at
suzanne@systemsguild.com