



Time Constrained Requirement Engineering – the Cooperative way

By Ole Jepsen, JAOO Academy, Denmark

Preface

Requirement Engineering is easy! And having very short time to do it – makes it even easier! But only if you do it right – and yes: There are a lot of traps on your way through the work with requirements. But many of the traps are well known – and can be avoided by following a few basic strategies.

In this paper I will invite you to take a tour with me through a couple of my experiences with requirements. And I will extract some survival strategies out of the experiences...

(Many of my ideas are much in line with the Agile Alliance Manifesto. Link to www.agilealliance.org if you want to know more.)

Experience 1: Requirements for Internet-bank – version 1

A timeline of this project is shown in Figure 1.

In the middle of June 2000 I received a phonecall from a large Scandinavian bank:

- “Will you help us build an Internet bank solution for our private customers? You will be working with requirements and use cases...”

At that time I wanted to get some experiences with web system development and use cases, so I answered:

- “Yes”
- “When can you start? We need to get going very soon...”
- “July 1th” I suggested
- “Can you start earlier?”
- ...

One day use case workshop

We started up with a one day workshop on July 3rd. Participants were

- one use case mentor
- the main user
- one business woman
- the project manager
- one analyzer
- and me.

Only one or two had prior experience with use cases, so the mentor gave a 1 hour presentation of use cases. The project manager presented the goal: “We want a net-bank

up and running before the end of this year”. Then we started playing with use cases and a draft of the main page of the Internet-bank. The only written input was one Powerpoint slide with words like: Account overview, transactions, payments and transfers, etc.

After lunch the project manager said to me:

- “Ole, I want you to be responsible for the gathering of requirements. I have some other important issues to deal with - like staffing the project...”

I accepted – and at the end of the day I found out, that the development kick-off meeting was already scheduled... On August 7th about 8 free-lance programmers would show up – expecting some specifications to be ready...! So I had 5 calendar-weeks – in the middle of the vacation period – to get the requirements settled. Scary!!!

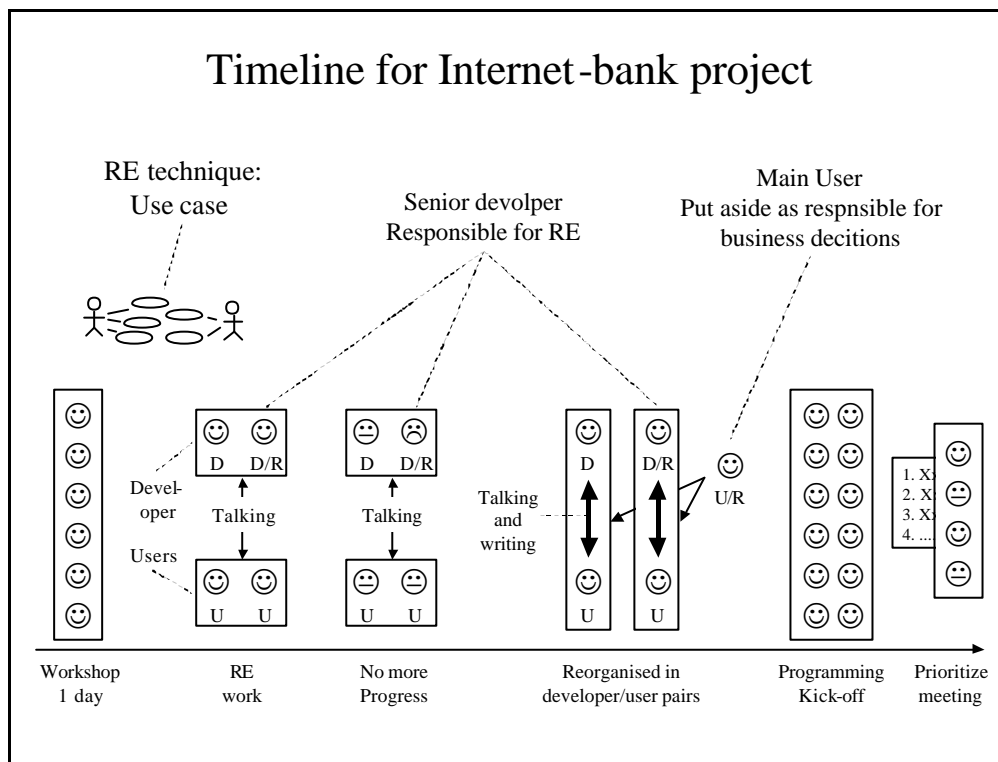


Figure 1

The following week

After the one day workshop - we had a rough list of actors and use cases. The next day the analyzer and myself moved into the office, where the main user and the business woman had their desks. The analyzer and I started to interview the users to get more details. Everything they said – we wrote down in the use cases. And after just a few days we had a fairly good picture of what the system should look like...

Problem with progress – and the solution

But then – after the first week or so – things started to slow down. We had 40-50 use cases. Some use cases with just a few notes, and some with more details. Discussions amongst the users started to take time, and I felt, that all progress stopped.

To solve the problem, I reorganized the work by forming two pairs – each with one user and one developer. Then we assigned a few use cases to each pair – and agreed to finish these use cases completely before continuing with other use cases. We also defined some maturity-steps for each use case to go through – with QA activities for some of the maturity levels. When I formed the pairs – I did not put the main user in a pair. Therefore he has not a part of the actual work with the specifications anymore. The aim was for the main user to have the time to supply the two groups with business decisions, answers and QA-reviews – and it worked!

After making those changes - the magic started! Having the one “use case partner” made it so much more motivating to write something. And the mix of IT- and bank knowledge in each group made it possible get the details sorted out – completely – in “pair - talk and write sessions”, that became the preferred way of working...

Programming kick-off

On the development kick-off on August 7th I presented an overview of the system by going briefly through all of the use cases, we handed over the specifications for most of the use cases for the developers, and they started to build the system, while we wrapped up the last use case specifications.

Prioritize meeting

Shortly after the kick-off we realized, that we needed to prioritize the use cases and other requirements. The leads of the project gathered a whole day to prioritize, and we had some fairly intense discussions. After a 10 hour meeting (!) we went out to celebrate, that we had agreed on priorities for all of the use cases.

Some four months later we had the first version of the Internet-bank up and running! Everybody was happy – and everybody had a fun and learning experience!!!

Strategies – part 1

Figure 2 shows where the following strategies fit into the project history.

1. *Use a simple and collaborative Requirement Engineering technique – I suggest use cases*

OK – I admit it: I’m crazy about use cases. They are the best requirement technique I have ever experienced. Because users actually understands use cases (unlike all other modeling techniques I have tried to drag the users through). I think the reason is, that use cases focus on the *interaction* between the actors and the system. And even when you dig down in details about normal and alternate flows – you stay focused on the actor-system interaction. This is important *and interesting* for both users and developers – and therefore working with use cases encourages and supports a collaborative working climate.

Also the use case technique supports an incremental approach to RE. As described in Alistair Cockburns “Writing Effective Use Cases”, you can work in four steps:

1. Identify Actors and use cases
2. Describe normal flow
3. Identify alternate flows
4. Describe alternate flows

This gives you two advantages:

- It gives you control over the situation (which use cases are at which steps)
- It gives you an overview of all the functional requirements – very quickly. An overview, that will help developers understand the whole idea of the system – even if only a part of the use cases are completely specified – step 1 to 4.

2. *Have one senior/manager-type of person responsible for the requirement process – and nothing else.*

When you need to get your requirements finished in a short time-frame, this work needs full attention from a management point of view. Problems will probably occur along the way - and they need to be solved instantly to keep the requirement process up and running. The project manager could run the requirements process – but he will have many other responsibilities. The 100% focus on requirements from one person will solve RE problems quickly, and that is essential when time is short.

3. *Pair up with the users – and work in pairs*

When you need to clarify and specify the requirements in a hurry, you simply do not have the time to “wait until the next meeting with the users”.

Many developers and analysts communicates with users mainly on meetings. After the meeting they go home and write what they got out of users on the meeting. Then they start organizing the next meeting, and after some days they meet with the users again- to discuss their writings and to get answers on questions that got to them between the two meetings...

Anyone who have tried to *be* with the users – in the same room – full time – know, that the “meeting-approach” is *so slow*, when you compare the two.

My experience is, that you speed things up even more, if you work in pairs with a user and a developer on each pair.

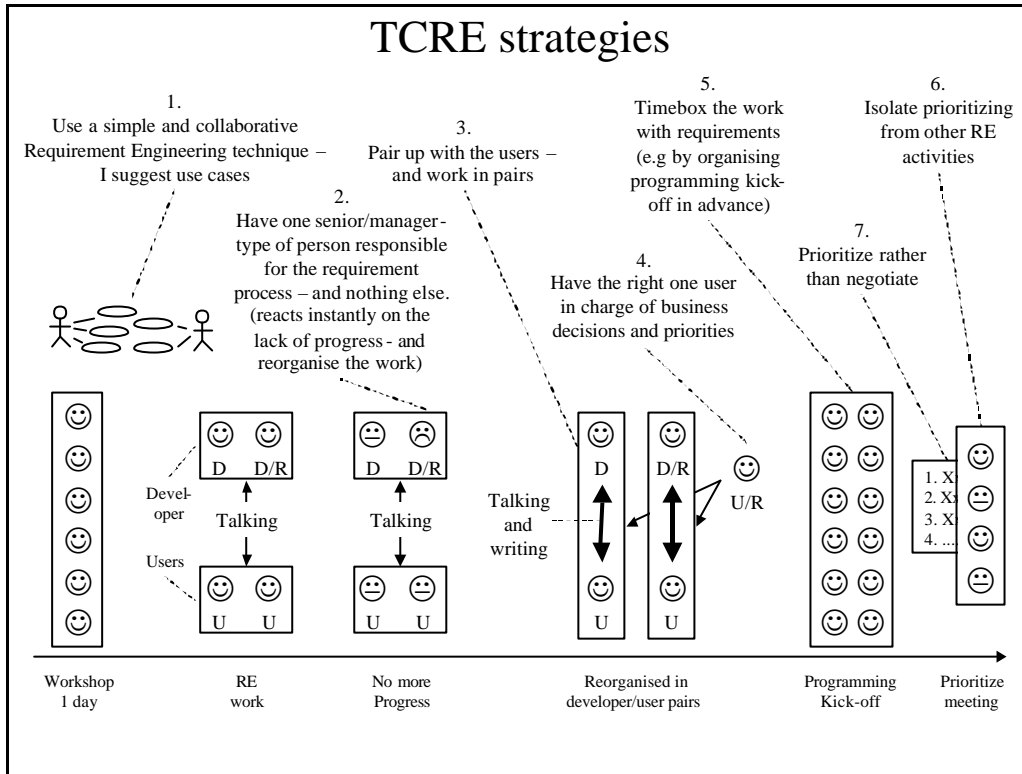


Figure 2

4. *Have the right one user in charge of business decisions and priorities*

Most development projects brings up a lot of questions, that are new to the business. Different business-people will probably have different opinions – based on the ir different backgrounds and interests in the project. Business discussions and qualified decisions are clearly essential to the project..., but so is speed. Having one user in charge speeds things up.

This one user is a key-player in the project – and needs to be the “right person”. He must know very much about the business, he needs to have a good network with other business experts in the company, he must have a strong personality, he must know the project sponsor priorities – and he must have a close and trusty relationship with the project sponsor.

5. *Timebox the work with requirements*

As any other activity in the project, being under a certain pressure helps you speed things up. In RE it will put a pressure on the developers *and* it will put a pressure on the users. This pressure will speed up the specification of what we know – and it will speed up the clarification of what we don’t know – and (very important) it will speed up the business decisions.

A timebox, where you have a whole team of programmers showing up at a certain date, is very motivating. Reason is, that it is very obvious that you waste money every hour, if requirements are not ready. Other ways would probably work also, as long as you make sure, that “taking a little extra time” is *not* an option.

6. *Isolate prioritizing from other RE activities*

Discussing the money, the time and the functional scope of the project is an inevitable part of any development project. There’s no way, that we can avoid these discussions – and they should be taken. It is the only way to ensure, that the development budget is spent on the most important projects and functionality.

However – having the priority-discussions being a part of the work with specifications will give you several problems:

- It slows down the specification work, because the priority-discussions sneaks in everywhere. In total the time spent on prioritizing will be much less, if it is isolated from other RE activities
- If you do not control the prioritizing – everybody involved in the project will think, that their opinion should count. That is probably not what you want.
- When you are deeply involved in details about a specific requirement, there’s no way you can have the total overview of the whole project at the same time. People trying to prioritize in this “detailed state of mind” will be doing a poor job. You need to be in an “overview state of mind” to prioritize.
- Prioritizing will usually imply negotiations between users and developers.

Negotiations does not support the cooperative working climate, that you want in any development project. Therefore it is a good idea to limit these negotiations to a few key-players – and to a few meetings during the project.

7. *Prioritize rather than negotiate*

Although there will always be some negotiation on the prioritize meetings, you should try to negotiate as little as possible. Rather than negotiate whether this or that use case or functionality can be solved in the next release of the system – do everything in your power to get a prioritized list of use cases as a result of the meeting. In order to do that, you need to understand the business impact each of each use case.

Prioritizing rather than negotiating will hurt the cooperative climate less – and it will support the cooperative climate amongst all of the people in the project. Users and developers now have one common goal: To implement as many use cases – as much functionality – as possible.

Experience 2: Requirements for Internet-bank – version 2

The heritage from version 1

After implementing the first version, we had a great team!! The cooperative climate was really good, so we had great chance to add functionality to the Internet-bank in a cooperative and efficient way. One thing, that saved us a lot of negotiation-like

discussions – was a rule, that came into play on the test-priority-meetings just before we went live with the first version:

- “I don’t want to hear any more discussions about whether a problem-report describes a bug in the code - or a new or changed requirement” – the project manager said.

After that we only spent time on priorities, risks, size, etc, for each problem-report. There were no more discussions about “who are responsible for this mistake”– or “who have forgotten that” - and what a relief!!! Now focus was: “What is most important right now” and “how do we fix this” – and the team-spirit came sneaking back into the test-priority-meetings.

Getting started on version 2

In the second version, the sponsor wished to add two services to the Internet-bank:

- Application for – and even establishing of – loans and credits
- Credit card services like “order new card”, “report loss of card”, etc.

The project manager asked me:

- “Ole, do you have any ideas to how we can repeat – and maybe even improve – the way we worked with requirements in the first version”
- “Let me think about it.” - I said.

The requirements this time were not as straight forward as before, because it would be completely new to the business to actually let the customers create loans and credits on their own. At the time, the bank had routines up and running, that calculated credit scores for all customers periodically. So the basis for the self service loans was in place, but still...

Based on that background, I came up with a suggestion after about a week. I suggested two workshops in a row – and after one more week of planning, we had the following two workshops:

Workshop 1 – Business Processes

The purpose of this workshop was to play a little more with the business experts for both Loans and Credit Cards.

We assigned two days for the workshop – one day for each subject – and had “everybody” there:

- a process modeling mentor
- the users from version 1
- credit card business experts
- loan and credit business experts
- the project manager
- most analyzers and developers
- one business person – now responsible for the requirements



- a workshop facilitator (me)

We were around 20 people at the workshop. We wanted to make sure, that the workshop would result in some kind of “product”, as the input for the next workshop. To ensure the creation of a product, we split up the workshop in two groups about halfway through – after some presentations and discussions. Each group was supplied with “brown paper” and sticky notes – and was asked to come up with

- a process model for the self service solutions
- problems and ideas

After that, we had more time, and we weren’t really sure what to do... There were a lot of business decisions to be made, but that didn’t really happen right there – at the actual workshop. Business experts had to take many of the new business issues back to their departments for further business-assessment, etc. That made me a little nervous... “Now the requirement work will be put to sleep again!”, I thought...

We agreed to move on a little – so we asked the people at the workshop to start working on use cases, and got quite far actually...

I felt, that the work with use cases, brought a lot of clearness to all of the “fluffy” new business ideas, that had come up during the workshop... The use cases somehow made it easier for the business experts to move from “discussion mode” into “decision mode”...

After a lot of hard work on that Thursday and Friday in November 2000, we went home. All were exhausted, but also very pleased and proud of what we had achieved in only two days...

Workshop 2 – “Use case camp”

The experience from version 1, where all progress with the requirements suddenly died after about one week, was very present in my mind. I absolutely did not want that to happen again...

To keep the steam up, we had a second workshop – straight after the first one. This was a 5 day “use case camp”, where about half of the people from the first workshop worked hard with the use cases.

We had the workshop happening in one room with the brown papers from the first workshop on the walls, we had white boards – and we had one computer for every two people. Pairs of two were formed – preferably with one developer and one business person – and each pair had a group of use cases assigned to them.

Most of the core business experts were not participating in this second workshop, but we deliberately arranged this 2nd workshop to take place in the office building, where the business experts were situated. Next to the workshop room, we had meeting rooms, where we had clarification-meetings with the business experts – and many business decisions were made...

During this week other meetings, other assignments, e-mails and phonecalls were cut down to a minimum.

This whole setup around the “use case camp” created an intense and productive atmosphere – and after this week, we were very far with the requirements for the second version of the Internet-bank. Everybody were VERY happy and proud about how far we were – after only seven days of work.

Personally I had never seen anything like it before – in my 20 years with system development. It was perfect...!

Strategies – part 2

Figure 4 shows where the following strategies fit into the project history.

8. *Treat errors and change request the same way: Evaluate and prioritize*
Some project managers say to me, “No, no, Ole – if users can freely change their requirements, we will never finish anything.” I agree to that statement, but they haven’t got my point.
My suggestion is, that any request for a change in the system – as it is at any given time – should be evaluated and prioritized together with all the other requests. Decisions should be taken based on risk, estimates – and value for the business. And I can’t see why errors should always – per definition – be more important than new or changed requirements...
So I’m not saying, that developers should make *more* code-changes. I’m just saying, that they should make the right code changes – seen from a business perspective.
This approach is most likely to work, if there is a trustworthy relationship between the users and the developers – and if the contract is based more on time and money, than it is based on functionality – as described in the next strategy.
9. *Time and money – over functionally*
Many projects try to describe which functionality, they need to develop, and then – based on the amount of functionality – they estimate the cost and length of the project. (Se figure 3).
I suggest that you turn this upside down: Start agreeing with the project sponsor about the time and the money. Then make a very, *very* rough prioritized list of functionality with the project sponsor.
Functionality will be something, that you talk about during RE and the rest of the project – no matter how you organize the project. In these discussions, there are no absolute right and wrongs, so requirements and functionality will be – *and should be* – discussed during the whole project. These discussions about functionality are likely to take place in friendly atmosphere – because there are common interests for both users and developers: To explain/understand the

requirements without misunderstandings and to implement as much functionality as possible.

Time and money are – on the other hand – are black and white: Either the system was delivered within the budget – or not. Either the system was on time – or not.

Because of the nature of time and money, most people can't take it, if promises about time and money are not fulfilled.

In short: Functionality is “elastic” – and therefore well suited for cooperative discussions. Time and money are “firm” – and therefore better suited for negotiations.

You want a cooperative working climate in any project. Therefore: Agree on time and money, and then cooperate and prioritize to make the amount of functionality fit into the budget.

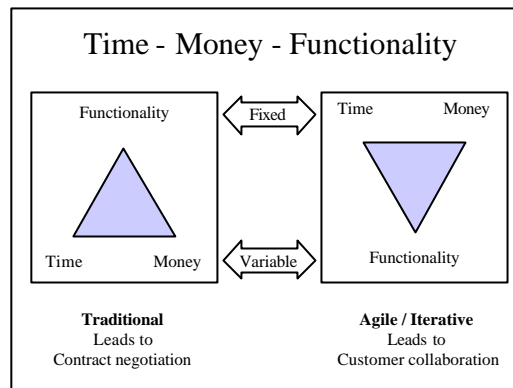


Figure 3

10. *Workshops are great – also for writing sessions*

Working in workshops gives two major advantages:

- Having the right people together at the same time
- Minimizing distractions

The value of these two things is enormous. It is a great way to kick-start your project, so if you do not have much time, workshops is the only way to go.

Traditionally workshops have mostly been used for discussions and brainstorming, but they are also very well suited, when you need to have some specific work – like RE – done in a limited time.

Do not underestimate the value (and the work) of planning a workshop: Getting the right people to come, having the right environment (meeting rooms, white boards, brown paper, computers, ...) and knowing exactly what product, you want the workshop to produce.

I sometimes run a “pre-workshop” with one or two others – just to be prepared of what might happen...

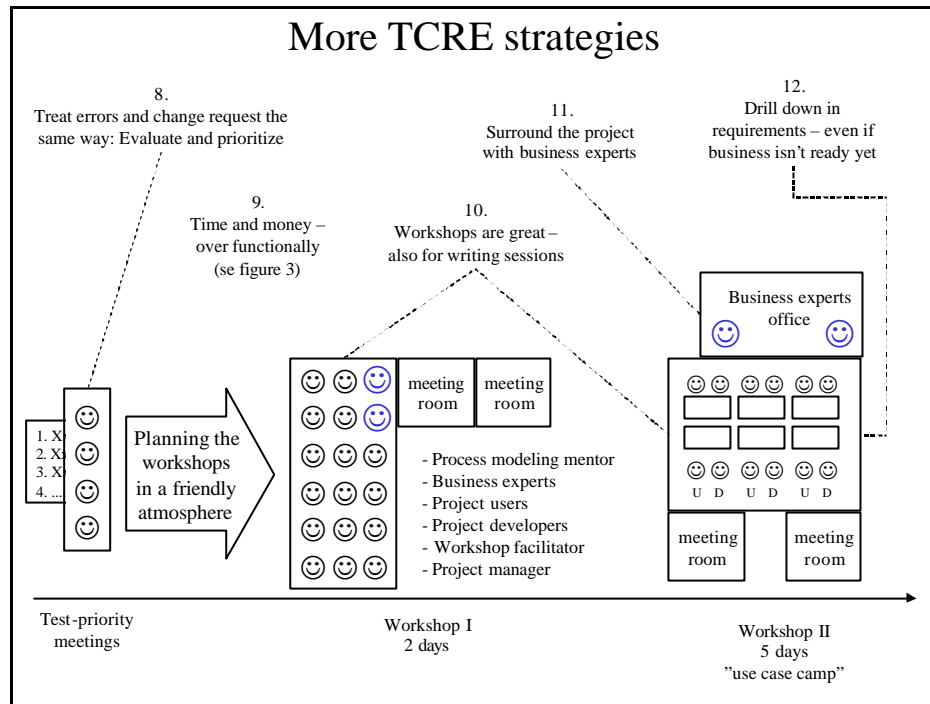


Figure 4

11. *Surround the project with business experts*

This may be stating the obvious, but I still see so many projects, who think they can run projects without business people... (“But the users have sent their requirements, so...”).

If you want to have success – you need to have good users and/or business people assigned to you project – preferably 100%.

But even when you have good users on board your project, it is a major advantage to have the core business experts just down the hall. It cuts down the wait for information and decisions dramatically – and therefore makes your RE and your whole project much more efficient.

12. *Drill down in requirements – even if business isn't ready yet*

Sometimes I experience business experts, who “need more time to think”... Fine with me – but only if they actually spend the time thinking – and only if their thinking brings the m closer to the business decisions, that we need so badly in our development projects.

When I occasionally get the feeling, that the business people aren't getting any closer to any decisions, I trick them: “Let's make some use cases and some screen layouts”, I say. And suddenly it becomes clear to them:

- Which implications their decisions will have on their business and on the system
- That the projects success is depending on them



Summary

Cooperate, cooperate, cooperate!!!

My experience from several projects is, that cooperation between users and developers is essential for the success of a project. The “we” instead of the “them” and “us” is everything for the working climate in the group. It turns the group into a team with common goals – and that is the only way to succeed, especially when time is short and Time Constrained Requirement Engineering is on the agenda.

Working in a group with an intense team spirit, that meets its goals, is such a great experience and so much more fun!

Project management is: Take control, and change things that doesn't work

When I teach project management, I hear a lot of “but's” – like:

- “But I do not have a single senior person in my group to put in charge of RE...”
- “Right Ole... but my users doesn't like to leave their own desks...”
- “But there are so many stakeholders from the user side on my project, and they will never agree on which one should be in charge!”

In my opinion, it is the project manager's responsibility to use his power to change these things. For if he doesn't – then he can't deliver value for money to the sponsor. And if he doesn't deliver value for money, he will cost the business a lot of money!!!

In the above examples – appropriate actions would be:

- Spend the money to get a person who *can* lead the RE work. In the long run it will pay off.
- Explain the users – or the users management – that there is no way you can deliver the system quickly and cost efficiently, if you can't have the users full attention for a while. Get the sponsor to help you – if necessary.
- Spend some time to “find the power” in the organization. Who has the final word at the end of the day? Sometimes it helps to ask the sponsor who he will trust to take care of his interests and his money in the project. Then put this one person in charge of the business decisions and priorities of the project.

Ole Jepsen
oj@jaoo-academy.com
May 2002