

Development and sourcing : Managing IP, Assets and Costs

Author : Valerie Thorn BSc FRSA

Class: Embedded System Show 2009 – October 7th 11^{am} - 12^{am}

Venue: FIVE International Show Centre – Farnborough – Expo Floor

Expected key learning outcomes from the class

- Understand the differences between IP, assets and costs.
- Recognise how team dynamics effect these elements.
- Increase knowledge of techniques for making management decisions

Abstract

Development of embedded software is costly and reliant on highly skilled labour. Companies however often find that the goals of the design team and the goals of higher level management are mis-aligned. This presentation highlights three critical components of the development team project and its outputs. It exposes the issues that team managers are faced with when sourcing or developing software and explores some techniques which can help them make decisions which align more naturally to company goals. The presentation draws on results of research undertaken at AND Technology Research Ltd into project goals and outcomes. The research also draws on academic research in the field of Technology Management and Innovation. References and further reading are included.

1 A real-life problem

1.1 A project example

The importance of embedded software to manufactured goods is easily observed. The technology in cars for example to control the engine, provide in-car entertainment and ensure comfort and safety could not be realised without the efforts of software engineers and the embedded software they create. It has long been recognised that embedded software development is costly and this therefore presents a number of investment challenges to companies.

The risks involved in new software development and the need to limit risk and cost has been a significant driver to companies looking a different ways of achieving their product development aims[1,5]. Leveraging value from investment is a further driver. Re-engineering code to produce an enhanced product, selecting specific development platforms and tools for company-wide use are examples of this.

Reports however from industry indicate that whilst re-engineering is commonplace writing 'green-field' code, is also commonplace. Code re-use is not favoured by engineers, particularly where the engineers are unsure of the code quality and the ways in which it could be used in a new project. It would appear that the risks

attached to re-using code are often perceived as greater than the risk of authoring code from scratch[1]. Managers also face a difficult challenge; if existing code is seen as 'risky'; how do they quantify this risk. An identifiable disconnect is evident between the requirement to leverage value and the need to manage risk.

In order to investigate this disconnect, case study research was undertaken at AND Technology Research Ltd based on a group of multi-media based projects. The study analysed views of the projects from differing perspectives. Interviews were undertaken with differing stakeholders, including an engineer, a manager, a consultant and a director, and project documentation was reviewed. The aim of this study was not to look at the issues faced by engineers when they come to re-use code, but rather how they can identify code that could leverage value. Results of this analysis, lessons learnt and methods used to overcome the challenges faced are used to inform the activities undertaken in this class.

1.2 Defining the project goals

If a requirement from investment in embedded software is that its value can be leveraged, then clearly this should be a goal. But differing contexts, for example industry factors and business conditions mean that goals are often unique for each situation. The first question raised therefore is what goals do various project participants have.

View from	Type of goal described	Goals
Director	Company goals	<ul style="list-style-type: none"> • Provide product & services • Invest in R&D to increase revenues
All	Product/ service goals	<ul style="list-style-type: none"> • Create a product, • Ensure product to spec and quality
Manager/ engineers	Embedded software development goals	<ul style="list-style-type: none"> • Complete a project on time and in budget, • Gain knowledge and experience

Table 1 Definition of goals

What is evident is that although the goals of all the stakeholders align around the product and service itself. The company goals (focused on the company offering and investment) and those in engineering (focused on the project and the engineers themselves) were not directly shared. Further none of the goals mentioned embedded software explicitly or the need to leverage value. The next sections seek to understand the reasons behind this.

2 Development and sourcing

2.1 Taking a cost driven approach

The engineering team are focused on completing the project. This includes not only software engineers but also the managers and consultants. Cost, functionality and time are three parameters used to dimension a project. Adjustments can be made in all areas but essentially one parameter will be overriding. This may be cost, if there is a finite budget; time, if delivery for a show or market launch is set; or functionality if performance is key. Balancing these three is termed the software economics [2].

Cost often comes under the closest of scrutiny driven by budgets and finance input. For embedded software there are a number of sourcing options both internally (essentially using software 'made' in-house) and for buying or acquiring externally. In recent years the trend has been to outsource software development [5]. Results show however that outsourcing often does not realise expected cost savings. A lack of understanding of the differences between in-house production costs and costs for acquisition is crucial if a component of any product is to be purchased. But further to this it has been seen that to outsource software projects successfully a full and detailed knowledge of the software development process is required in order to ascertain when to engage outsource partners. The context in which the outsourced software is to operate, domain knowledge and careful supplier selection are also key factors. It seems to appear that often in the drive for cost savings partners can be wrongly selected or selected at the wrong time or for the wrong purpose. Where cost is the main measure of success it has been seen that success is limited.

It must be remembered however that 'in-house' software projects are also noted for failure or 'not achieving a significant number of the targets'. Maybe one reason is that as software economics dictates that trade-offs between cost, functionality and time will occur and if the success criteria cover all three parameters, then failure is inevitable. For example to meet functionality criteria, the time, the cost or both may have to be extended. But if extension of cost or time are not possible the clearly the level of functionality will suffer. Maybe different metrics are required. To explore this further the case study was used to determine whether or not by examining the outcomes of the project in terms of successes and failures when compared to the goals defined, a different perspective could be found.

2.2 Assessing the outcomes

View from	Type of goal described	Successes	Failures
Director	Company goals	<ul style="list-style-type: none"> • Work well rewarded, • Cost effective robust product created. 	<ul style="list-style-type: none"> • Extensive communication to customer required more time
All	Product/ service goals	<ul style="list-style-type: none"> • IP created, software and techniques • New processes trailed • Extra testing paid off 	<ul style="list-style-type: none"> • Project overruns due to lack of communication in the team
Manager/ engineers	Embedded software development goals	<ul style="list-style-type: none"> • Knowledge gained, digital standards • Experience gained particularly for multi-media platforms 	<ul style="list-style-type: none"> • Dependency on hardware caused delays, communications stressful

Table 2 - Successes and failures.

Table 2 shows that whilst project overrun was clearly an issue the focus of attention of the failure was not the time or cost involved but communications. Further the failures recorded could be interpreted in at least two ways. For example, one could deduce that extra time was required for communications and that this clearly had caused project overrun and stress to the engineers. But one could easily also deduce that communications was stressful for the engineers which caused the project to overrun which meant that more extensive communication was required with the customer.

Looking at the successes, matches with the goals can be seen. For example a robust product was created, knowledge and experience was gained. However even so the successes are not measured in terms of cost, or the development project itself, they instead reflect the output. Essentially they focus on assets created and the rewards obtained.

In summary the case study showed that metrics based on the project parameters of cost, time and functionality are not reported readily by engineers. The focus of their attention was on communications and project outputs. Different metrics based perhaps on the project outputs could be more useful.

3 Building competitive software

3.1 Taking a value approach

Work in the area of NPI (New Product Innovation) [3] shows the necessity of adding value when introducing new products in a supply chain. Referring again to the results in table 2, if the measure of success was value creation rather than matching cost targets then success would certainly have been seen to be achieved.

An alternative approach is therefore to view the project as a means to add value rather than to develop code.

3.2 Competencies and asset creation

The resource based view of a company (RBV) looks at how a company's offering can be competitive in the market place. Here an assumption is that if the offering is not competitive then the company is likely to fail. Simply, assuming that the starting point of a build process is a set of resources, using these resources competently should ensure a competitive outcome [4].

Embedded software development requires significant levels of human resource. Taking a resource based view (RBV) it is reasonable to assume that if the building software requires significant amounts of human resources then the competencies of the engineers will have a significant effect on the competitive nature of the outcomes. It is well recognised that the skills and knowledge of the engineers are a significant asset to their companies and although important, this point alone seems almost trivial. Maybe however what is less trivial is to think about the resources the engineers use in their work which enhance their competencies and increase their competitiveness.

In an earlier section it was noted that re-use of code was not favoured by engineers at least in part because of the risks involved. In contrast re-engineering or re-cycling code is common [1]. The reason for this is clear. Re-engineering familiar or trusted code does not contain the risks associated with fixed re-useable code blocks and in that sense the trusted code can be seen to enhance the engineers competence. .

Essentially this section has described two different types of resources that have been used, knowledge and code. Where these resources are used multiple times and where they are used effectively it is clear that value is leveraged. The resources can then be viewed as assets which attract a value as a piece of IP (Intellectual Property). Metrics based on this IP may then be developed .

3.3 Intellectual Property

Software is often described in terms of Intellectual Property (IP). But what is meant by IP?

The term IP tends to be used generically when intangible assets need to be transferred from one party to another, sometimes with a value attached although often this is not explicit. For example development project contracts often include statements relating to ownership of the IP at the end of the project. The meaning behind these clauses is reasonably clear. They enable either one or other of the parties the rights to further develop code, use code in products etc or protect one or other party from abuse of further development or copying. However this does not explain why software itself can be termed IP. Nor does it explain the relationship between intangible assets and rights. Three definitions of IP are given below in order to provide clarification.

Defintion	IP example
Asset	Intangible property such as the software code or human capital, ie knowledge described above
Legal	The rights in law attributed to authored works, brands etc. In the case of software copyright law applies and in some cases patents. Licences can also apply.
Economic	A collection of intangible assets and rights which are brought together in order that a value can be attached.

Table 3 –Definitions of Intellectual Property

The combination of assets and rights together form an IP set to which a value can be assigned. This is important for development projects for a number of reasons.

Firstly the need for metrics based on value of assets has already been highlighted. Assigning value to knowledge or code is difficult to achieve. A view of the knowledge of code as IP may help to achieve a meaningful valuation.

Secondly if competitiveness is to maintained, stocks of code and knowledge must be identified as assets and preserved and/or protected. Whilst this stock may not necessarily need to be valued for accounting purposes (the argument being that as the stock is intangible its value in a final product may not exist). It is important to recognise its value in terms of competencies and the value it adds to the creative process. Without this a false view of the value of the output would appear inevitable.

3.4 Packaging IP for trade or valuation

Valuation of tangible products, particularly those in mature market places, is relatively easy. For a development team creating intangible assets which may not yet have been proven in any market place this can be difficult. Companies face these difficulties in a number of ways.

Taking a lead from the IT industry, software commodities with accompanying licences can be traded. It follows that if the development project assets created can

be packaged as commodities for trading then it must be possible to assign values to them.

The interdependence of embedded software, hardware and engineering know-how mean that creating embedded software commodities is difficult. Development techniques such as object oriented coding and abstraction allow some companies to pursue simple commodity routes. Another clear option is to embed coded logic in a specific device such as an FPGA. Where the packaging of simple code based commodities is not an option the most common route is to offer a combination of components. Knowledge hardware and code can be combined into products such as development platforms, application notes and demonstrators for example.

Whichever of these routes is chosen a clear view at the beginning of a development project of the way in which the outputs are to be valued is undoubtedly required if metrics based on the outputs are to be set. The question then arises as to how IP can be valued.

3.5 Assigning values to IP

Table 2 showed that the engineering team in the case study recognised that IP had been created when assessing the goals. The next phase of the study was to see if they could assign values to this IP. The assignment process used three steps and values were obtained.

Step	Activity	Result
Identification	<ol style="list-style-type: none"> 1. Split the software into constituent layers (based on the ISO 7 layer model) 2. For each layer identify separate components 	<ul style="list-style-type: none"> • Drivers, • Timing algorithms, Codecs, , File system, Scheduler, User interface • Test suite
Value assessment	<ol style="list-style-type: none"> 1. Take each component 2. Team review to decide if it is seen to be valuable 	<ul style="list-style-type: none"> • Timing algorithms, • Codecs, , • File system, • Test suite
Value assignment	<ol style="list-style-type: none"> 1. Examine each valuable component to determine they a method by which the value could be assigned. 2. Use method to assign a value 	<ul style="list-style-type: none"> • Test suite valued as a product • Codecs & Timing algorithm as IP that could be sold or re-used internally • File system as code for re-use

Table 4 - Assigning values to an example project

From table 4 it can be seen that valuations were successfully assigned to relevant assets. This is not to say that the assets were to be sold externally. Whilst the valuations raised that possibility of trading the assets, the valuations were required in

order to assess the value that had been added to the project and the value of the stock code and knowledge used.

If values can be assigned to the output the clearly project metrics based on value could be used instead of those based on cost, time and functionality.

3.6 How valuable is IP?

A note of caution however must be introduced. Whatever value is assigned to IP on completion, the real value of any piece of IP can only be assessed when the IP is used. In some instances the IP may be valuable and in others it may not. This will very much depend on the context in which the IP is to be used, the complexity or the interfaces and the competence and attitude of the engineers involved [1].

Many problems are faced by engineers when unpacking IP and re-using code. Further, whilst the problem associated with unpacking IP may be greater for externally sourced items the sentiment applies to internally source code as well. The real value of the IP will be realised if the engineers are able to unpack it efficiently and use it effectively.

3.7 Handling team dynamics

Throughout the paper a number of issues have been identified as being in part reliant on the attitude of engineers, the interpersonal skills and essentially the dynamics of the team.

This case study used highlighted two specific issues, one of value assignation to IP and the other communications. A common understanding of the issues is recommended if the team is to perform efficiently and effectively. One way forward is to define a set of interfaces which represent the project team structure and then to pose one key question at each interface to address any areas of concern. Table 5 shows an example of this which addresses the areas of concern encountered in the case study.

View from	Key value question	Key communication point
Customer / Management interface	Are we capable of adding value in this project?	How do we communicate success/failures, shared goals and value created?
Project interface	Do the economic and technical requirements mean we can create outputs we can value?	How do we identify communication points and assess the value of the project outputs ?
Team interface	What resources and competencies do we have to ensure we create maximum value?	How do work together to create the maximum value and how do we communicate this ?

Table 5 Adressing common understanding

4 Analysis and conclusions

4.1 Aligning goals

The need for managers to minimise risk leads them to assessing projects against metrics that can be set at the project planning phase, e.g. cost, time and functionality. Although these metrics are important, the case study showed that the project teams focused on created outputs rather than project statistics. Communications and assets were their main concerns.

4.2 Leveraging value

This aim of the paper was to look at ways in which value in embedded software investment could be leveraged in the context of a development project. The paper has looked at two different views of assessing development projects a cost based view and a value based view.

It was proposed that planning and risk control favours a cost-based view. However the results of the study showed that the cost based view did not provide clear link between the project goals and the value of the output. The value based view did provide that link and furthermore it was found that leveraged outputs could be identified and values assigned.

4.3 Team dynamics

Engineers are the key resource in embedded software development and the traits of individual team members have been recognised as important to a successful project outcome. Communication skills, technical competency and attitudes have been highlighted.

For managers therefore it would seem essential that team dynamics including the mix of competencies and interpersonal skills should be as much at the fore-front of project planning as the cost and the time. This may be difficult when proposing a project however as team members may not have been assigned or the team may not exist, but this situation could also be seen as an opportunity to ensure that the best team is selected. A method of addressing the need for common understanding in the team is proposed.

4.4 Summary of results

The results of the case study showed the following:-

1/ That a value based view of the project can be used effectively to set metrics

2/ Assets in the form of code and knowledge can be identified at the end of a project and valued.

3/ Team dynamics are important in assessing the project outputs as well as ensuring the best use of team resources. Managers are recommended to consider team dynamics when planning and resourcing a project.

References and further reading

[1] Hunt F; Shehabuddeen, N. 2000: How to make the most of what you've got - A practical guide for supporting re-use, IFM Cambridge University, Vol. 2

[2] Boehm, B, Chulani, S, 2002. Software engineering economics, Software Management 6th Ed IEEE Computer society Press, 2002: 239-271

[3] Canez, L., Platts, K. and Probert, D. (2000), *Make-or-buy: a practical guide to industrial sourcing decisions*, IfM, Cambridge

[4] Barney, J, 1991. "Firm Resources and sustained competitive advantage", *Journal of Management*, 17, 1: 99-120

[5] Hunt F.H., N. Shehabuddeen, C.J.P. Farrukh, D.R. Probert and S. Wilson. (2004): A process for sourcing software for embedding in products. PICMET 2004, Seoul Korea, August 1–4