Finding the Common Ground for Legacy Interfaces

Chris Roast, Richard Brophy and Andy Bowdin
Sheffield Hallam University
Sheffield, UK, S1 1WB
+44 (0)114 225 5555
c.r.roast@shu.ac.uk

Abstract:
It is well established within interface development that user requirements are conventionally identified and documented prior to design and implementation. With the growing uptake of information technology the development process also needs to manage the re-engineering of legacy systems. Re-engineering not only provides an opportunity to enhance user interface quality, but also to broaden the user base.

This paper reports on the techniques developed in the re-engineering of two distinct legacy systems with the aim of identifying and meeting common user requirements for both. More generally, it is proposed that these techniques should benefit user-centred design in the context of widening user bases.

1. Introduction

The growing uptake and awareness of information technology has provoked the proliferation of legacy systems. Developers are being required to re-engineer existing systems, which were considered perfectly adequate, but now fail to satisfy the requirements of more a sophisticated user base. In addition, to enhancing the system interaction, cross product re-engineering provides an opportunity identify and meet common user requirements more effectively.

User Interfaces for All encourages the recognition of the range and diversity of users for which interactive systems should be usable, focusing upon the non-exclusion of user groups. Within the context of interactive system development this not only concerns issues of effective and appropriate interaction but also the identification of the user tasks and requirements that are conventionally determined early in development (Ryan and Sutcliffe 1998, Carroll 1995). The work in this paper reports upon techniques that have been used to contribute to identify the user requirements common to distinct systems.

We report on the adaptation and application of Requirements Engineering (RE) and Human Computer Interaction (HCI) techniques to enable effective re-engineering for a broader user base. The work has been driven by a case study of a small software house engaged in re-engineering two legacy software products, with the aim of exploiting commonality between them. The specific legacy systems are the result of a history of poorly documented product evolution, hence, re-engineering activity can be classified as one of "design recovery" (Chikofsky and Cross, 1990) for the reverse engineering aspect of the project. The two products provide similar services to two vertical markets with

distinct user bases. Hence, the activity naturally concerns itself with identifying and meeting user requirements that are common to the two user groups. The ideal outcome

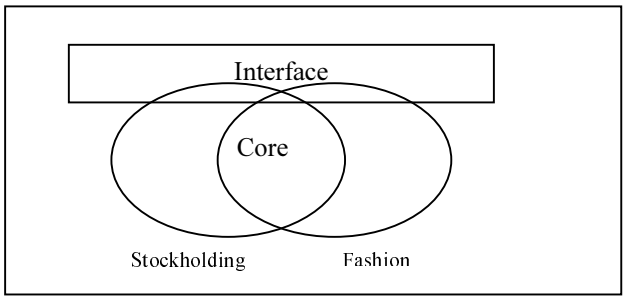

Figure 1. Stockholding and Fashion Systems areas of commonality

of the activity is to have an interactive system fulfilling the user requirements common to both groups and in addition satisfying the distinct requirements of a specific user group. We look towards maximising the core elements thus ensuring a degree of flexibility satisfactory for both existing user groups and new user groups.

Below we further characterise the specific nature of the cross product re-engineering process and the selection of the techniques used (Section 2). Following this, the process adopted is detailed, emphasising the enhancements necessitated by the desire to maximise a broader user base (Section 3). The application of the techniques within the case study is described (Section 4). Finally we reflect upon the effectiveness of the techniques employed.

2. Selecting a Process

In addition to the object of maximising commonality between the existing systems, the adoption of a particular process was influenced by other development based concerns. In particular there was a desire to employ contemporary object-orientated and model-based techniques. Also, there was a need for flexibility within the process to enable adaptation to the specific constraints governing development, including: enabling re-engineering activity, closely matching existing development practices, and accommodating essential, though sporadic, access to users.

There has already been different approaches to enhancing legacy systems, Merlo et al (1995). Their process involved the creation of a meta graphical language from the character based interface. This meta language was then used to create a graphical user interface. Tools where used to make the process semi-automated. Our approach is different in two key aspects. First, that Merlo et al (1995) are looking at representation issues of an interface while we are looking at structural issues. Second, that they are not combining two legacy systems into one to extract the core functionality, but are translating one old system into one new.

Another method of requirements acquisition for an object orientated system would be 'Use Cases'. The 'Use Case' method can be described as a 'case of use' and is suggested by Booch, Rumbaugh & Jacobson (1999). There are problems with their application, if they are used on their own as a method of elicitation. Arlow (1998) pointed out that;

1. Use Cases can tend to trivialise business requirements, which means key requirements can be lost.
2. If functional requirements are being clumped together into one 'Use Case' then this can lead to a problem of prioritising later on in the project.

The acquisition of business requirements is one of the major reasons for reverse engineering as they are not available else where. When time and resources are limited then being able to prioritise key aspects of the development becomes crucial.

The process selected was the "Combined Model" developed by Kaindl (1998), this provided a process in which user requirements and interaction are clearly linked. More importantly, the model is accommodating of re-engineering activity and flexible enough to allow requirements to be developed and documented in a relatively opportunistic manner. Specifically, Kaindl's approach combines the use of goals, scenarios and functions, which enables requirements elicitation to proceed by developing coherent and complete inter-relationships between sets of all three of these concepts.

Despite the appropriateness of Kaindl's approach, it does not explicitly address our primary objective of identifying common requirements for legacy systems. However, the process offered by Kaindl is sufficiently rich and flexible to enable the consideration of identifying and maximising a common core.

The issues to be addressed in the adaptation of the Combined Model process:

- **Diverse Requirements** - How to combine diverse requirements associated with different user groups.
- **Legacy Features** - How to identify if characteristics of a function in the legacy system reflect valid user requirements, or if they are a design feature reflecting implementation details.

A more general requirement for the process adopted is that of its integration with existing development process, in particular its use within an object-orientated design process. The Combined Model integrates the documenting of scenarios and functions both of which can directly contribute to object-oriented development. However to help initialise and structure the process a task oriented framework based on Forbrig (1999) was employed. Forbrig encourages the development of inter-linked task model, object model and user model, for the existing system, the envisioned system and also for the application model.

3. The Adapted Process

The overall process is an adaptation of the Combined Model in which a common task model is developed to provide initial goals and scenarios. Given this information, inter-linked sets of goals, functions and scenarios are identifiable - the functions being a

product of the analysis. In order to effectively address issues surrounding diverse requirements the process is enhanced to enable requirements to be classified as to their origin and appropriateness. To tackle the problem of legacy features the process is to focus upon deriving functions which are then compared with those evident within the legacy systems.

Preparation

Initially, user profiles of the two legacy systems were developed to clarify the context and nature of use. Following this, a common task model was developed, this represents the most specific account of task that can considered common to the use of both systems. For example, in order to find a stock details record one legacy system enabled search by "batches", where as the other system enabled search by "analysis grouping", hence, finding stock details was not decomposed any further. The common task model provides a reference to activities common to both systems and initial sets of goals.

Iteration

Kaindl's Combined Model describes dependencies between types of requirements, and articulates guidelines for an iterative process by which sets of goals, scenarios and functions can be developed and their consistency and completeness assessed. Two examples of the type of guideline:

If some goal is known, then try to link it to one or more known scenarios

If some scenario is known, then identify the goals that are achieved through it

(Paraphrased from Kaindl 1998)

Figure 2 shows the primary dependencies between these entities, in general:

- user goals can relate to system functional requirements by identifying the purposes of particular functions,
- scenarios can relate to functional requirements by identifying the behaviour relevant to particular functions,
- scenarios can relate user goals by considering the plans of activity.

Plans, purposes and behaviours have been classified as intersection entities to support the management of elements originating from different legacy systems. This provides an easy way of cross checking between functions, scenarios and goals to ensure that one legacy system scenario is not being solved by another's functionality.

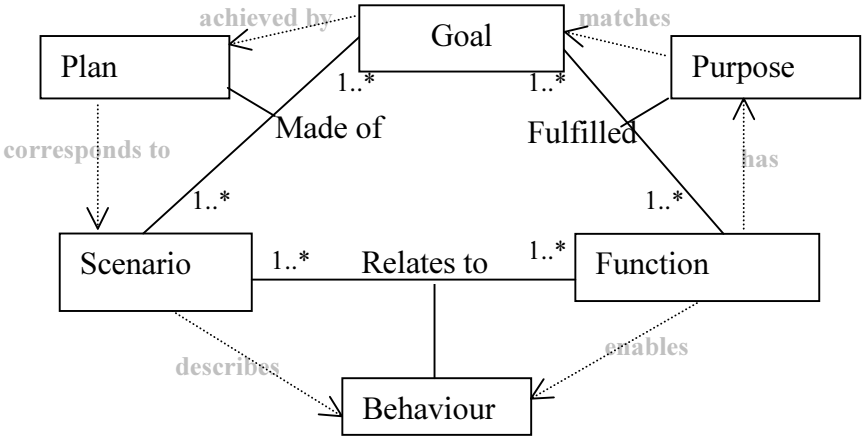

Figure 2. Adapted Model

In the above diagram (figure 2) the dotted lines and grey text relate to the original Kaindl (1998) combined model relationships between entities. They have been added to give extra clarity and to show the differences in the models.

Dealing with Diverse Requirements

Addressing the problem of maximising common ground between diverse requirements, each goal, scenario and function is classified as being of one specific legacy system or both. Given these classifications, inter-relationships such as the behaviour linking a function and a scenario, can be assessed in terms of whether common functions (or scenarios) are legitimate. In particular, we would expect to see a consistency in the treatment of goals, scenarios and functions of each class.

Identifying Legacy Features

The problem of focusing upon required functions evident in the legacy systems, as opposed to legacy design features, was tackled by comparing functions identified by the above analytic activity with those of the existing legacy systems. Any apparent omissions can be individually assessed as either legacy implementation dependant functions, in which case they were ignored, or required functions, in which case they were added to the function set. The importance of recovering a design by separating out implementation bias functions has been noted by Byne (1991). The overall process of consistency checking means that newly added functions naturally demand the identification (and possibly creation) of goals and scenarios to support them.

Validation

Throughout this process users can be involved in validating the requirements identified. However access to users for validation is an often limited, yet highly valuable resource.

Hence, it is envisaged that the preparation phase should be the primary focus of validation activity, providing the information on which the subsequent analysis relies.

In ideal circumstance we would to ensure user involvement in the other phases of analysis especially in the assessment of legacy features and the generation and validation of scenarios, as demanded by the analysis. The Combined Model is of particular value here since it enables user validation based on differing requirements representations. User feedback can be gained on functions with a reference to situation specific examples, so a user can more easily relate the two.

Completion and Use

The iterative process of identifying and modifying requirements of each type continues until each set is considered to be internally consistent and complete and the relationships between each set are also coherent. For example, if there is function which is not described by any of the scenarios, the analysis may consider the legitimacy of the function or identification of a relevant scenario.

Once completed, the scenarios and functions identified could be employed as inputs to object orientated design. The scenarios can be used as a starting point for 'use case' analysis mapping into object orientated development. Functions can be used to identify object competence's (Booch, Rumbaugh & Jacobson 1999) as well as feedback to help prioritise the future development. While the Task model, used to create the goals, can be used as a basis for the design and construction of the interface.

5. Case Study

The case study is that of a small software house developing a new system combining two legacy systems. Although the systems are developed within the same company they are the product of separate and evolving product teams and have limited explicit or planned overlap. The systems have been developed, enhanced and maintained over several years.

The company culture would be defined as 'Club Culture' by Handy (1990). Club company culture can be characterised by being centred on the leader of the company and staff personal knowledge and experience. This means that the organisation is rich in personality. All software development has to take place in this context, as pointed out by Henderson-Sellers (1999).

The three main reasons the company is redeveloping its products are;

1. Efficient use of resources – At present there is duplication because of the two systems.
2. Market Trends – As noted in the introduction users are becoming more sophisticated.
3. Opening Markets – Identifying commonality will make it easier to create new niche market software.

The legacy systems are for the Fashion industry market and the ferrous and non-ferrous metals Stockholding market. Despite the two distinct markets, there is sufficiently common functionality for the processes of development and maintenance to benefit from identifying and modelling the common ground.

The Common Task Model

The preliminary phase involved was the creation of the user profiles and common task model, in both cases these were validated with members of the development teams familiar with the two user groups. The user profiles provided background material for

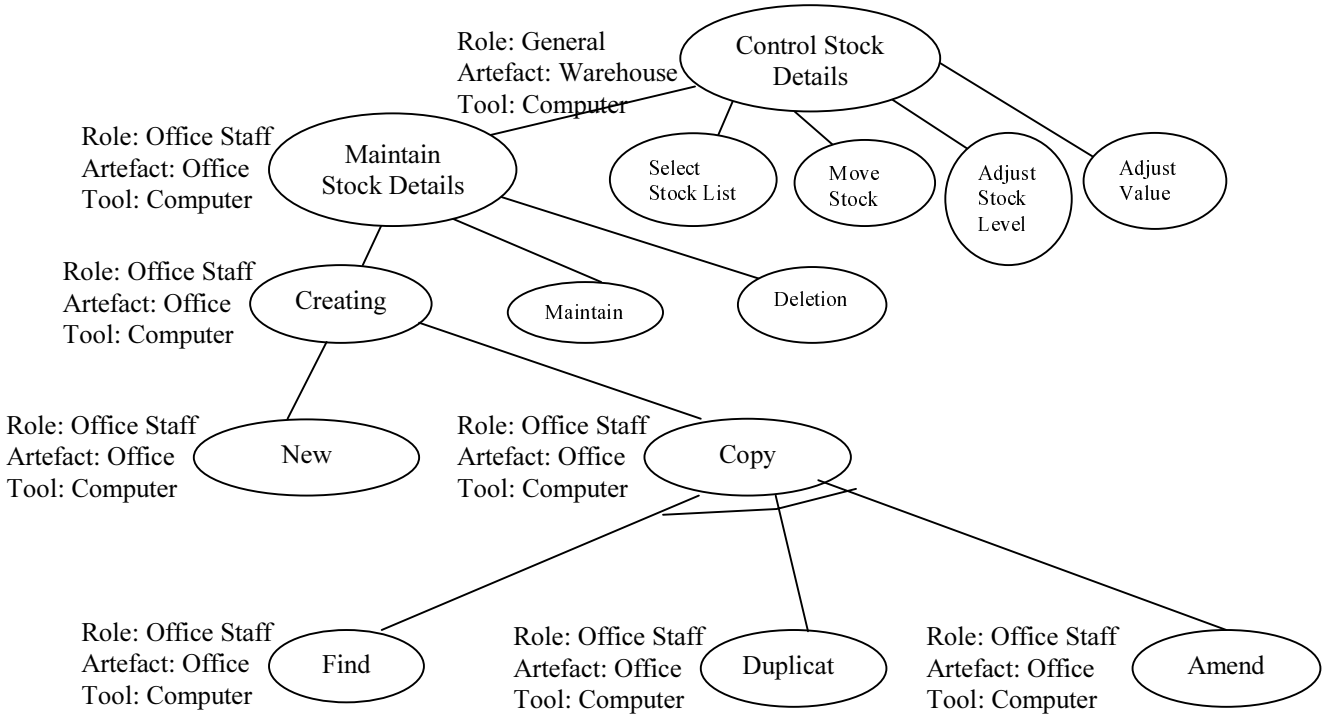

Figure 3. A Task Model

the development of the common task model. The task model employed a light-weight notation in which each task, the role, artefact(s) and tools were identified, conjoint or disjoint sub-task sets are positioned below tasks. Figure 3, illustrates one task within the common task model. Being common to both legacy systems, task decomposition was limited to task that failed to discriminate between the two systems. For example, within the figure 3, the task of "Find" is not broken down further since the selection criteria for finding in the Fashion system differed considerably from those in the Stockholding.

Using The Combined Model

The common task model provided the initial input for the use of Combined Model, in the form of an initial set of common goals. In addition, scenarios conforming with the task model, yet specific to a legacy system were elicited from users. These initial scenarios proved to be problematic, they were large and detailed, included aspects of

legacy system behaviours and characteristics outside the focus of the re-engineering process. For example, while focusing upon say the stock control, a scenario may refer to packages, such as customer accounts, which was not subject to revision or analysis. These scenarios provided little scope for use in the Combined Model, and as a consequence it was decided that they should be segmented into smaller scenarios, and where possible, those which did not related to system being consider were excluded. These scenario segments served as a more effective initial set for the application of the Combine Model.

To assess and manage the origin and relevance of different requirements, each goal, scenario and function was tagged as follows:

B	-	Both	(Core System)
S	-	Stockholding	(Stockholding System)
F	-	Fashion	(Fashion System)

Hence, goals within the initial goal set were tagged B (since they were derived from the common task model), and scenarios within the initial scenario set were tagged S or F depending upon their origin. Figures 4 and 5 illustrate some of the scenarios and goals used.

The analysis within the Combined Model then proceeded by the iterative process of checking for consistency and coherence within the set of tagged scenarios, goals and functions, and refining the sets appropriately. This relied upon applying guidelines such as those of Kaindl illustrated in the previous section. However, in addition to developing the representations, we identified tag consistency rules to promote the identification of common requirements, while wishing to avoid the un-managed transference of requirements for one legacy system to the other.

Generalising over the analysis of the specific case study, the following tag consistency rules were proposed. The rational of these rules is of trying to find convergence in the two systems with a goal of closure. In general these have the effect of enabling common goals and functions within the analysis and encouraging system specific scenarios.

- **If a plan links, goal(s) X 'composed of' scenario(s) Y:**
  if goal(s) X is common (B), then its composed of scenario(s) Y can be both common (B) and specific to a system (S or F), e.g. G 1 B from Figure 6 is composed of S 1.4 B, S 1.3 B, S1.4 F & S1.5 S.
  if goal(s) X is specific to a system (S or F), then its composed of scenario(s) Y must also be specific to the same system (S or F).

- **If a behaviour links, scenario(s) X 'relates to' function(s) Y:**
  if scenario(s) X is specific to a system, then it relates to function(s) Y can be common or specific to the same system, e.g. S 1.4 F relates to F 1.7 F.
  if scenario(s) X is common, then it relates to function(s) Y which is common, e.g. S 1.7 B relates to F 1.3 B.

- **If a purpose links, function(s) X 'fulfils' goal(s) Y:**
  if function(s) X is specific to a system, then goal(s) Y can be common or specific to the same system, e.g. F 1.7 F fulfils goal G 1 B.
  if function(s) X is common, then goal Y is common, e.g. F 1.1 B fulfils goal G 1.1 B.

Only some of these rules were applicable within the case study, since not all circumstances arose. For example, the initial goals taken from the common task model were all classed as common (B), at no point did a goal specific to a system arise. Greater testing will be taking place in the future as the tag rules are applied in the rest of the systems elicitation acquisition. The first of the above rules is worth commenting on further. Since, both common goals and system specific scenarios formed the initial inputs to the process, to develop coherent and complete sets it was important to associate the two. This association is enabled by the first rule linking common goals to specific scenarios.

Complementing the above point the analysis also benefited from identifying more general scenarios based upon the common characteristics of more specific scenarios. For example, figure 5 shows scenario S.1.3 (B), as abstracting over S.1.4 (F) and S.1.5 (S).

At regular opportunities the representation of requirements in terms of goals, scenarios and functions was subjected to validation drawing upon the experience of developers familiar with the existing legacy systems and the two user groups. This activity served two purposes:

- The functions developed within the Combined Model could be compared with those known to be present in the legacy systems. In this way it was possible to address the issue of legacy features.

  One such case of implementation bias arose in the consideration of transferring data between modules of the system. The analysis of goals and scenarios motivated the requirement for a "transfer data" function. However the legacy system in fact provided two separate functions ("dump data to pool" and "get data from pool") closely mirroring an implementation detail.

- The overall integrity of the developing representation could be validated, in most cases recommendations and revisions can could simply be accommodated as additions or deletions of the elements of the requirement represented. As a consequence, the process could continue with little difficulty.

Goals

G 1	Create product details	B
G 1.1	To be able to create a product details record	B
G 1.2	To be able to make a duplicate of a product details record.	B

Scenarios

S.1.1	The end user is setting up the system, they already have all the stock codes structured and prepared beforehand. Data is entered in a batch.	B
S.1.2	The end user is creating a PO (Purchase Order)/SO (Sales Order)/ Quote and the item does not exist. It needs to be quickly created then and there to allow the creation of the document.	B
S.1.3	An end user will order from a supplier. They do not know the exact details until that item has been received and invoiced. This is because a container load is bought. The quantity is known but not the sizes or colours as this has to be checked on delivery. The quality of each item is not exactly known as their has to be a stock validation check when delivered.	B
S.1.4	The exact details needed, for S.1.3, are the size and colour	F
S.1.5	The exact details needed, for S.1.3, are weight and finish	S
S 1.6	There is a new item in product that is nearly the same as an existing item. It should be possible to copy the existing stock record and then amend the details of the new product details record in the specific area. If a code is entered that already exists then a copy of that record should be shown.	B
S 1.7	When creating a duplicate it should not be possible to create a new product without amending some detail on the new product because it will cause confusion if the only difference is the code.	B
S 1.8	The list of product details records should mainly show what records are still active, not deleted, stock details records. Though there are times when this information is needed e.g. when an item is being undeleted.	B
S 1.9	Miscellaneous notes about a product should be able to be entered as needed about the specific product.	B

Figure 4. Extracts from the Goal and Scenario sets

Functions

F.1.1 If all the stock details information is known when the record is being created then all that information can be put in from one screen. B

F.1.2 Notes about a product details should be able to be entered when needed on any screen that has anything to do with a product. B

F 1.3 The copy function will display the details of the old product details record into the new stock details record. It will not copy the code as that will have to be entered by the operator. An occurrence will only be added when the product details have been amended. B

F 1.4 If a duplicate code is created details for that code are shown. B

F 1.5 The create functions should be accessible at any time from other packages. B

F 1.6 If all the product details information is not known upon creation then the record can be created as a prospect record. This record can be used only as a limited product details record until all the needed information is entered or it is deleted. B

F 1.7 Size and colour should be able to be amended separately. The function should be accessible in other programmes. F

F 1.8 Weight and finish should be able to be amended separately. The function should be accessible in other programmes. S

F 1.9 The list generated of product details should be switchable so deleted items, prospect and active items can be turned on and off. B

Figure 5. An extract from the function set

Control Stock Details and Maintain Stock Details

Plan		**Behaviour**		**Purpose**	
Goal	Scenario	Scenario	Function	Function	Goal
G 1.1 B	S 1.1 B	S 1.1 B	F 1.1 B	F 1.1 B	G 1.1 B
	S 1.9 B	S 1.2 B	F 1.5 B	F 1.2 B	G 1.1 B
G 1 B	S 1.2 B	S 1.3 B	F 1.6 B	F 1.3 B	G 1.2 B
	S 1.3 B	S 1.4 F	F 1.7 F	F 1.4 B	G 1.2 B
	S 1.4 F	S 1.5 S	F 1.8 S	F 1.5 B	G 1.2 B
	S 1.5 S	S 1.6 B	F 2.2 B	F 1.6 B	G 1 B
G 1.2 B	S 1.6 B		F 2.3 B	F 1.7 F	G 1 B
	S 1.7 B		F 2.4 B	F 1.8 S	G 1 B
	S 1.8 B		F 1.3 B	F 1.9 B	G 1.2 B
	S 2.1 B		F 1.4 B		
	S 2.2 B	S 1.7 B	F 1.3 B		
	S 2.3 F	S 1.8 B	F 1.9 B		
	S 2.4 B	S 1.9 B	F 1.2 B		

Figure 6. Mapping goals to scenarios to functions and back to goals.

To position the tasks involved in the creation of the combined model within the requirements acquisition process and to give a holistic view of the process a meta model has been created (see Figure 7). The process starts with data collection, which included the creation of user

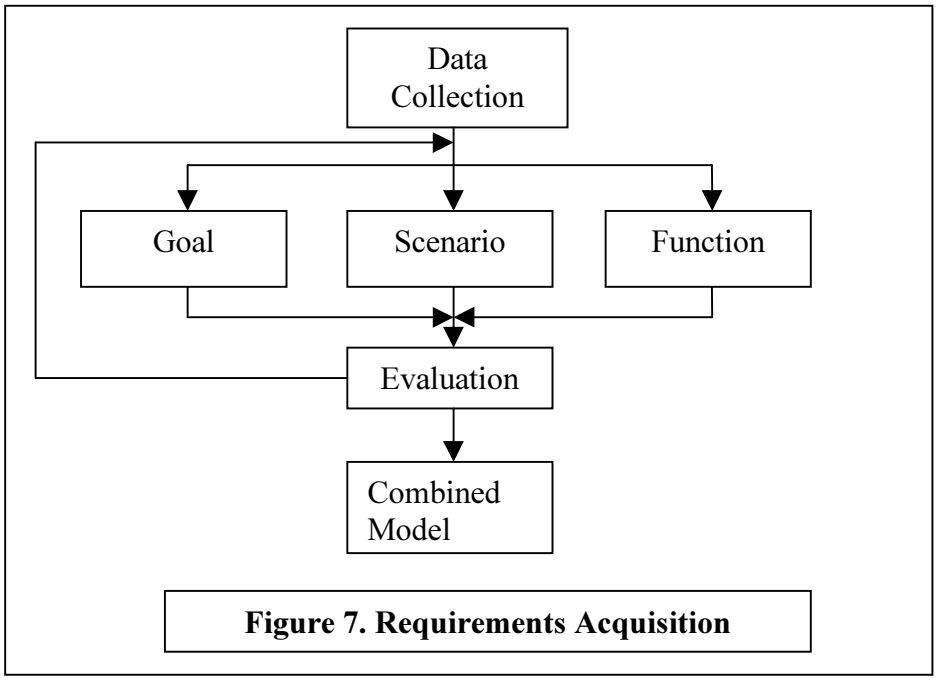

Figure 7. Requirements Acquisition

profiles and initial scenarios. In our case study we started with the creation of the goals, which included the creation of the common task model. Then segmented smaller scenarios were extracted from the initial scenarios. Last the functionality was extracted from the data collected from the legacy systems. This was evaluated and the cycle started again until all goals, scenarios and functions where combined and nothing was missing. The company culture does not support formal, rigid methods of elicitation. Hence the fact that the process is flexible and informal, supporting company culture and therefore gaining user support.

6. Discussion

The experience of applying the proposed process model for requirements analysis was governed by a wish to examine how the model performed as well as ensuring that appropriate decisions were made for the specific case study. Here we summarise these the departure from the envisaged process and consider their implication for other cross product re-engineering problems.

It was envisaged that the initial input for the requirements acquisition would be goals and scenarios drawn from the common task model. Despite the value of this initial goal set the process benefited greatly from legacy system scenarios which were specific to a single system. The scenarios could not be immediately articulated in terms common to both systems. In terms of the activities involved the scenarios provided for the analysis

were motivated largely during the validation of the common task model. The analysis framework offered by the Combined Model could easily accommodate this form of input. On reflection the provision of specific scenarios complemented the goals evident in the common task model.

The nature of the scenarios employed during the analysis phase was considerably different from those envisaged. Although the role of scenarios is often to provide highly situated representations of how a system will be used, specific details appear to limit their applicability in the planned process.

But for these two points the planned process largely fulfilled our expectations. Following the spirit of Kaindl's Combined Model we prefer to identify practical descriptions of how the model can be used as opposed to prescribing specific approaches. Thus, we prefer to view these differences from our planned process as being indicators as to alternative ways of using the same model. For example, the guidelines for the Combined Model could be extended to reflect our treatment of scenarios, by including:

If two scenarios appear to have a general characterisation, consider including this as a more generic scenario.

Additional guidelines, such as this, represent further opportunities for the use of the Combined Model.

In terms of meeting the original objectives for the case study, the adapted process of analysis has served its purpose. The analysis identified and increased the set of common requirements crystallising functionality that was specific to the individual systems. The representation of common user requirements has driven the object based system development of re-engineered versions of the two systems, with a significant shared elements. The key features of the process which have contributed to this are the identification of functions prior to considering those of the legacy systems, and the use of representations tagged to reflect their applicability.

It has been noted that the description of the functions and the goals themselves are general and are not specific at all. This was done on purpose. The reason for this is that it was felt the very specific goals and functions would later constrict programmers. Which would have the effect of confining or constricting their ability to create software solutions to a function.

In addition to these features, the overall process benefited from minimising conflicts of ownership and priorities when considering common requirements from distinct sources. In the specific case study, the common task model appears to have offered a ‘starting point’ that was not biased to either legacy system. As a consequence, the task model appeared to facilitate co-operation between users in the subsequent validation activities. In particular the co-operative spirit was reflected in the developers of one legacy system requesting that specific functions become common.

Future

A series of similar re-engineering activities are to be carried out using the process described in this paper. This will enable the further refinement of the approach we have taken. Amongst the issues to be addressed in the future are;

- The treatment of non-functional requirements.
- Closer mapping of the requirements acquisition 'combined' model into the Object Orientated Design (OOD) phase of the project.
- Using the task model as a basis for the creation of the interface.
- Greater testing of the tag consistency rules.

7. Conclusion

The work described has been motivated by the desire to identify user requirements that are common to distinct legacy systems and use them as a basis for re-engineering those systems. The issues addressed in the activity are highly relevant when we come to consider User Interfaces for All and wish to consider identifying and meeting common user requirements. Clearly, a specific system is developed with a specific purpose in mind and as such will be distinct, however where there is commonality between systems, both prospective users and developers can benefit from the recognition of commonalties.

The process of requirements identification described in this paper provides one effective mechanism for identifying common user requirements, that is applicable in the highly relevant context of re-engineering of legacy systems.

Acknowledgements

This work has been supported by UK Teaching Company Scheme. In co-operation with PCI Systems and especially Andy Bowdin for all his help and support. The reviewers of an earlier version of this paper for their constructive criticisms.

8. References

Arlow (1998) - Jim Arlow, 'Use Cases, UML Visual Modelling and the Trivialisation of Business Requirements' Requirements Engineering, Vol. 3 No.2 Pg 150-152, Spinger-Verlag London Limited

Booch, Rumbaugh & Jacobson (1999). - Grady Booch, James Rumbaugh & Ivar Jacobson. The Unified Modeling Language, Users Guide, Addison-Wesley 1999

Byne (1991) – Eric J. Byrne, Software Reverse Engineering: A Case Study, Software – Practice and Experience, Vol 21(12), 1349-1364 (December 1991)

Carrol (1991) – John M Carrol, 'Designing Interaction, Psychology at the Human-Computer Interface', Cambridge University Press, 1991.

Carrol (1995) - John M Carrol, 'Scenario Based Design: Envisoning Work and Technology in System Development', John Wiley & Sons.

Chikofsky and Cross (1990) – Elliot J. Chikofsky and James H. Cross II, 'Reverse Engineering and Design Recovery: A Taxonomy', IEEE Software January, 1990

Forbrig (1999) - Peter Forbrig, Task and Object Oriented Development of Interactive System – How many models are necessary?, In DSV-IS 99 6th Eurographics Workshop Design, Specification and Verification of Interactive Systems.

Handy C.(1990) – C. Handy. Inside Organisations, 21 Ideas for managers. BBC Books. 1990

Henderson-Sellers (1999) –) – B. Henderson-Sellers, 'A Methodological Metamodel of Process', JOOP/ROAD pg 56-58, February 1999.

Kaindl (1998) - Hermann Kaindl, 'Combining Goals and Functional Requirements in a Scenario-based Design Process', People and Computers XIII, Hilary Johnson, Laurence Nigay and Chris Roast (eds), Springer-Verlag.

Merlo et al (1995) – Ettore Merlo, Pierre-Yves Gagne, Jean-Francois Girard, Kostas Knotogiannis, Prakash Panangaden and Renato De Mori, 'Reengineering User Interfaces', IEEE Software January 1995 pp 64-73

Ryan and Sutcliffe (1998) , M. Ryan and A. Sutcliffe 'Analysing Requirement to Inform Design', People and Computers XIII, Hilary Johnson, Laurence Nigay and Chris Roast (eds), Springer-Verlag.