# Goal sketching: An Agile Approach to Clarifying Requirements

Kenneth Boness
*University of Reading, Berkshire, RG6 6AY UK*
*k.d.boness@reading.ac.uk*

Rachel Harrison
*Stratton Edge Consulting Ltd, GL7 2LS, UK*
*rachel.harrison@strattonedge.com*

Kecheng Liu
*University of Reading, Berkshire, RG6 6AY UK*
*k.liu@reading.ac.uk*

## Abstract

*This paper describes a technique that can be used as part of a simple and practical agile method for requirements engineering. It is based on disciplined goal-responsibility modelling but eschews formality in favour of a set of practicality objectives. The technique can be used together with Agile Programming to develop software in internet time. We illustrate the technique and introduce lazy refinement, responsibility composition and context sketching. Goal sketching has been used in a number of real-world development projects, one of which is described here.*

**Keywords**: goal-oriented requirements engineering, agile development, evolving systems.

## 1. Introduction

Our motivation for goal sketching is to help stakeholders who need to make project critical decisions in projects which develop emergent systems. The agility here concerns the manner of obtaining and maintaining the rationale of problem and solution requirements so as to be able guide projects. Hence goal sketching applies to, but is not limited to, software projects using agile development methodologies.

Decisions about investment and requirements priorities are the responsibility of stakeholders and can only be made rationally when supported by a coherent depiction of what is known about the requirements. It is well known that this is problematical: for example the importance of "creating realistic expectations in the minds of stakeholders" has been noted [1] and the observation that "..customers on agile projects are often asked to make critical, project-defining decisions, and very little of the methodology can help them make those calls." [2].

In contrast we suggest that (at least in principle) given enough time, information and skill, goal-responsibility refinement models can be constructed to represent the stakeholders' expectations for a system-to-be that will operate in an expected environment, in fulfilment of a contract. Such models can be produced using KAOS [3] and some use-case methodologies [4,5]. Each has a structured argument framework that allows the rationale to be verified and thus affords the possibility of formulating systematic evaluation of the adequacy and feasibility of the intended system. However the prerequisite criteria (time, information and skill) are not satisfied in the situations with which we are concerned. Hence our research question which we are investigating with an action research methodology is: can a lightweight adaptation of KAOS style goal-responsibility modelling meet the practical demands of the analysts and designers?

Of paramount importance is the clarity of the disciplined structure of goal-responsibility argumentation (with its quasi hierarchical depiction) as a possible basis for capturing what is known about the requirements and the agreed rationale for their satisfaction. Our methodology has 4 objectives:-

**Table 1. Objectives of goal sketching**

| | |
|---|---|
| 1. | To maintain a coherent depiction of the intention (the agreed-upon requirements and the rationale for their satisfaction) as it unfolds over time. |
| 2. | To be simple enough to allow a project manager or analyst to achieve a first draft, at a resolution good enough to steer high level priority decisions, at the outset of the project. |
| 3. | To keep the depiction understandable to business as well as technical stakeholders. |
| 4. | To support formal rigour on a "just enough" and "when needed" basis [6]. |

The methodology we are developing is called goal sketching [7,8]. It is also the foundation for our work on appraising development projects [9] called goal-

responsibility appraisal of soft projects (GRASP). It embraces established practices evolved to cope with uncertainty such as spiral [10] and breadth before depth [11] techniques. Similarly 'just enough' approaches such as in [12] inform our approach to time-constrained development.

This paper proceeds as follows. In section 2 we introduce the concept of structurally complete goal-responsibility (G-R) models and their adaptation to our purpose. In section 3 we present the current state of our goal sketching methodology. Section 4 reinforces the description using a hypothetical exemplar and section 5 uses an industrial application to illustrate our efficacy in regard to the above four objectives.

## 2. Goal-Responsibility Models

An example of what we mean by goal-responsibility model is shown in Figure 1. Models like this are used in goal oriented requirements engineering (GORE) such as the KAOS and also (with provision for the representation of responsibilities [8]) in some use case techniques.

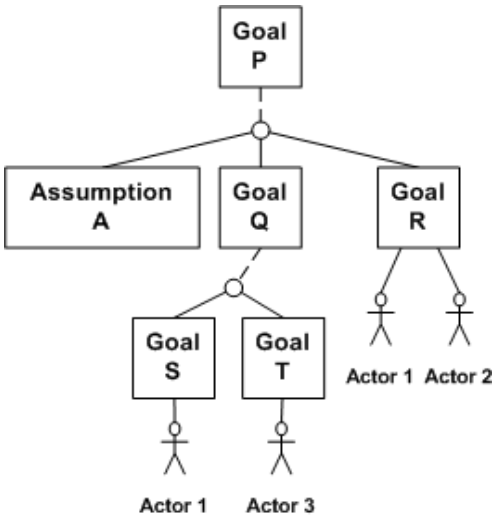


**Figure 1. Goal-Responsibility model**

Each box in Figure 1 is referred to as a 'goal oriented proposition' (GOP). In keeping with propositional logic each GOP must be defined in such a way that it may be refuted. The figure uses two types of proposition: assumption and goal. There is more to say about types but for now it is enough to note that a G-R graph can record explicit assumptions as well as goals. The aim when constructing a G-R graph is to capture the logic of the problem in hand moving by step-wise refinement from relatively abstract root propositions (e.g. goal P) to relatively concrete propositions that may be operationalized (e.g. goals S,T and R) or assumptions (which can only be trusted but not operationalized). Although the structure is hierarchical the analysis to create it is rarely top down; an analyst typically works with GOPs at all levels of abstraction. The aim (and skill) of the analyst (in goal sketching at least) is to organise the GOPs into a convincing rationale. In doing this it is usual that the analyst may discover gaps in the argument and then invents additional GOPs in order to complete it.

Each step of refinement is a satisficing argument where a proposition is refined into sub propositions such that the sub-propositions can be agreed to be collectively sufficient and individually pertinent to adequately satisfy the parent. We call this the *refinement argument criterion.*

Each argument step is deemed valid if by some defensible criteria (e.g. expert judgement and/or 'policy' such as in goal structuring notation (GSN) [13] or 'root definition' as in soft systems methodology (SSM) [18]).

The model is said to be *structurally complete* if (as in the figure) all objectives are ultimately satisfied by actors of the system-to-be. Thus P is satisfied by the combined actions and qualities of Actors 1, 2 and 3. It is important to note that in this type of representation the necessary behavior (and other qualities) that must be instantiated is described only at the leaves of the model; it is not distributed across the model. So if Q harbours required behavior to be explicitly represented in S and T then a further GOP should be added along with S and T [8].

When a G-R model is constructed in a formal logic (such as KAOS) there are calculi to verify the argumentation. Hence if the model is also structurally complete and all necessary root GOPs are included the model should amount to an adequate intention for the stakeholders. Further, if the responsibilities are individually and collectively operationalizable within the constraints of the project the intention can be said to be feasible.

This potential for systematically evaluating the adequacy and feasibility of an informally produced G-R model is a key intended benefit of our goal sketching technique; especially since for our purposes (with our assumption of a incomplete information) structural completeness is only possible if the analyst places assumptions and, or very low precision GOPs into the rationale. A G-R model constructed this way, out of necessity, is a rich resource to draw on to promote informed negotiation among the stakeholders.

## 3. Goal Sketching Technique

In this section we present the details of the technique so far developed through our programme of action research. We outline a methodology for using the technique and then proceed to particular details concerning the support of building refinement arguments.

### 3.1. Using the Technique

Our goal sketching technique starts with the creation of a goal graph which expresses the high level motivations behind the intention to develop the software. This is typically a coarse but structurally complete sketch of what is understood about the overall intention. In general there is often a vague long-term vision coupled with some short-term clarity. A series of staged developments are planned using the system graph as a guide. This compliments the practice of sprinting in Scrum [14], and the increments in an iterative and incremental development process [15]. Each stage is preceded by taking a portion of the system graph in its current state and refining it so that there are no remaining vague intentions. This is called the 'stage graph'. In the execution of any stage it is possible that the stage graph will be updated as a result of the usual agile practice of improving the quality of the work in hand. At the completion of each such stage its graph is used to update the system graph. Thus the true goal graph emerges by successive iterations and refactoring and so becomes the inventory, recording the associated rationale for posterity.

When preparing each stage the goals are refined only as far as necessary for the stage in hand (a technique called *lazy refinement)* relying on stories, use cases or activity sketches. (This does not preclude the use of formality as problem frames [16] or temporal logic etc may be used when necessary.)

We advocate using *pair sketching*, in which the goal graphs are sketched by two people working together (often the analyst with a stakeholder) to ensure that the refinement argument is sound, in a manner akin to pair programming. Once an acceptable goal graph has been produced it is incorporated into the system goal graph. The system graph may need to be re-factored for the next stage.

The goal graphs are exported to a database for subsequent analysis. From the database we can produce matrices to expose *composition* issues which may arise from cross-cutting concerns for analysts, designers, developers and testers.

### 3.2. Refinement Argument Supports

In our goal sketching the GOPs are written in natural language and must satisfy the *refinement argument criterion*. This is a very simple principle but in practice it can be very difficult to do. Errors that we have observed in students and would-be industrial practitioners, and ourselves, include:-

1. Mixing two or more problem contexts (e.g. mixing operation with construction of the system-to-be) in a confusing argument.
2. Expressing 'milestone' refinement patterns [3] as multi-level rather than single level refinements. This leads to an invalid though seemingly structurally complete G-R model.
3. Volatile functional refinement arguments that depend upon the current outlook of the analyst.

As mentioned above, *pair sketching helps* but we have found that it is very important to to be mindful of four aspects of a GOP, which we list Table 2 as support to the practitioner.

**Table 2. Aspects of GOPs**

| | |
|---|---|
| 1. | The type of proposition: e.g. assumption or objective. |
| 2. | The proposition owner: e.g. a stakeholder role or a system. |
| 3. | The problem context of the proposition in terms of where operationalization can be enacted; e.g. in the domain of the operating system-to-be or the domain of the development of the system-to-be. |
| 4. | The refinement level: i.e. to try to keep all sub-propositions of a proposition at similar levels of abstraction. |

These supports are are discussed in the following sub-sections.

**Type.** A goal oriented proposition is a refutable statement written in natural language which as shown in Figure 2 we specialize into five types.

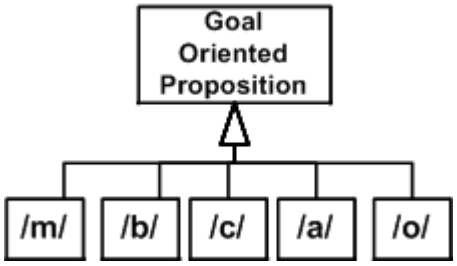


**Figure 2. Goal types**

**/m/** is a motivation goal representing the agreed-upon concerns of the stakeholders that motivate the project; in terms of KAOS they are likely to be "soft goals". They may harbour refinement implications that require refinements in different problem contexts with different time-spans (e.g. project time or system run-time). Even when they apply to a single context they can only be satisficed [17] and their refinement should include an assumption giving the justifying world-view (similar to weltanschauung [18]). For example the refinement of 'achieve greater efficiency' might include goals such as 'provide data at point of need' but it would depend upon an assumption linking the provision of data to greater efficiency.

**/b/** is a behavior goal explicitly required by the stakeholders or by force of circumstance and necessary for completeness. It 'affords' [19,5] an option or freedom to a user, whether or not the user chooses to exploit it. It combinines the capability and condition elements of a 'well formed requirement' in the IEEE recommendations on systems requirements [20].

**/c/** is a constraint: a nonfunctional requirement that limits the possible system implementation solutions. It is a factor that is imposed on the solution by force or compulsion and may limit or modify the design changes. This is consistent with [20].

**/a/** is an assumption: something that is stated on trust but is necessarily true for the rationale to present a defensible argument; i.e., it is 'load bearing' [21]. An assumption may be a simplifying argument used to find an acceptably easy goal refinement argument or it may be a KAOS domain property [3].

**/o/** is an obstacle used as in KAOS to oppose the satisfactory fulfilment of any other proposition. These propositions are not discussed further in the following models as they remain substantially as used in KAOS.

The /b/ and /c/ propositions are strictly bound to system run-time whereas the /a/ propositions may be either project or run time.

**Problem Domains and Context.** Jackson ties requirements statements to domains in a rigorous fashion [16] as illustrated in Figure 3 where the requirement is understood as referring to phenomena in the domain.

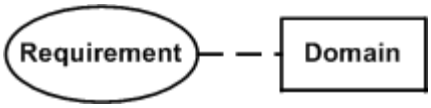


**Figure 3. Requirements context**

A requirement straddling multiple domains is shown by a dashed line to each [16] and the associated phenomena are referenced exclusively in each domain.

In goal sketching we advocate tying GOPs to their relevant domains in a similar fashion. In such diagrams, see Figure 4, we use { } to show that we mean a GOP.

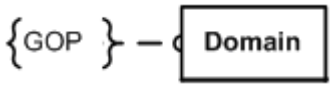


**Figure 4. GOP context**

The domain may be a large domain such as a business operation. Inside the business operation there might be sub-domains to which we attach lower level GOPs. It is in the nature of /m/ propositions that they may imply references to phenomena in the domain with different enactment contexts. We identify three such contexts in Table 3.

**Table 3. Three common contexts**

| |
|---|
| 1. The system-to-be. |
| 2. A system to manage the life-cycle of the system-to-be. |
| 3. The project to manage business change and the construction of the 'kit' [22]. |

In goal sketching we choose goal refinement steps that lead rapidly to referencing phenomena of single contexts. This technique helps to distinguish constituent domains that would obfuscate the G-R graph.

Starting with an agreed outer problem domain the *problem context* is established by attaching the root GOPs and any global constraints and assumptions (as a set of /m/, /a/ and /c/). This can be refined by then exposing the important inner domains and then attaching agreed GOPs (this time possibly including /b/ types). This technique echoes the work in [23] where problem frames are used to guide a goal refinement using business process modelling but is more lightweight.

We find that just as the goals require *sketching* (especially early-on) it is usual to sketch the domains; again adding precision on a just good enough basis.

**Owner.** In colloquial use a goal would be owned by a person (or group of people). For example: *Owner* "To make a profit."; or *user* "To reserve a book". This kind of ownership dominates use-case based GORE [4,5]. In branches of system engineering it may also be said that a goal is owned by the system (or indeed a machine) in the sense that it is an embedded objective. For example:the goal of a heat seeking missile is to find its target. This is the usual kind of ownership in KAOS. Generally in goal-sketching ownership passes from people to system as operationalization is approached.

**Structurally Complete Refinement.** In goal sketching the aim is to capture the logic moving from /m/ to operationizable propositions which will be a collection of /b/ and /c/ propositions and *en passent* it may be necessary to add assumptions /a/. We apply the following rules to guide the construction of a structurally complete refinement such as illustrated in Figure 1.

**Table 4. Goal sketching rules for a structurally complete G-R model**

| | |
|---|---|
| 1. | The roots of a goal graph must be /m/ propositions. |
| 2. | The leaves of a goal graph can only be /b/ and/or /c/ and/or /a/ propositions. |
| 3. | Every leaf of type /b/ or /c/ must have a complete set of responsible actors (see below) assigned. |
| 4. | Any /b/ or /c/ proposition may be refined into combinations of sub-propositions of types /b/ and/or /c/ and/or /a/ in the same temporal and contextual mode |
| 5. | Any /a/ may only refine into /a/ sub-propositions. |
| 6. | Any /m/ may refine into<br>i. /m/ sub-propositions, or<br>ii. /b/ and/or /c/ and/or /a/ sub-propositions.<br>In case (ii) there must be at least one /a/ that expresses the binding/justifying world view. |
| 7. | Refinement arguments must satisfy the refinement argument criterion. |

Note that as information improves it may become necessary to convert one type of proposition into another and then reconsider the refinement arguments and reapply the rules. This is typically the refactoring mentioned above in terms of stage and system graphs.

## 3.4. Lazy Refinement

Refinement should always halt when just enough detail is exposed to allow safe operationalization. Hence in goal sketching the degree of refinement applied is kept to a minimum. Often, especially early in a project, it must be halted owing to a lack of information. In terms of sprint based agile development there is an implied set of such goals pending exploration at a suitable time in the future. But it is important to capture such lack of information in a context that is informative to the sponsors and other stakeholders. In goal sketching this is left as a refinement TBD (to be determined) and is explicitly recorded on the graph.

In the interests of efficiency refinement can be halted at a relatively abstract level where the implications of operationalization are well known; i.e. they are normal [24] to the community (the key stakeholders). On the other hand where they are not understood (perhaps radical [24] to the community) a more rigorous refinement may be called for; this can be provided as problem frames and,or fully dressed use case analysis [4] and,or the usual methods of KAOS.

## 3.3. Operationalization

In Figure 1 the actors (aka agents in KAOS) are entities of the system-to-be that can take responsibility for the necessary enactments of the leaf goals. For example: Actor 1 is responsible to enact, effect or be whatever goal proposition S requires. In this case no other actor is involved. In the case of goal proposition R it requires two actors in collaboration. The nature of the collaboration will be interpreted from the specification of R.

For lazy refinements the specification may be informal such as: a simple statement, a software engineering template specification [25], an eXtreme style story or a use case. It is typical in lazy refinement to have multiple actors collaborating.

In full refinement, as in KAOS, the objective is to have a unitary relationship between a requirement or expectation (equivalent to goal propositions) and an agent (actor in goal sketching). Alternative methods of achieving and specifying full refinement, which we prefer include Jackson's Problem Frames [16], activity diagrams [8] and use-cases. These are also illustrated in the example below.

### 3.5. Composition

When creating clear refinement arguments goal sketching favors a strict policy of separation of concerns. This implies decomposition and thus necessitates a late re-composition [16] as cross-cutting concerns (e.g. collaboration between responsible agents to indicate necessary superimposition of capabilities, constraints and conditions). In our experience this approach minimizes the number of goals with multiple parents and thus reduces visual tangling in the goal graph. The price for this benefit is that the composition concerns are not explicit. However a lightweight solution is to annotate the assigned responsibilities using a system of *composition tags* (see Figure 5). In contrast KAOS uses object and operation models to accommodate composition concerns. This can be rigorous but tends to be heavyweight.


**Figure 5. Responsibility annotation**

Figure 5 shows three versions of the responsibility assignments. Each is shown as an oval with the name of an assigned agent followed by a full stop. The architectural precision of the agent depends upon the underlying domain analysis being used; e.g. an object in a UML model or a sub-domain of a Jackson context diagram [16]. An optional system of semantic tagging is allowed after the full stop. Each tag is written in the form "<MYTAG>" or <@MYTAG>. Any responsibility with a given tag (say <MYTAG>) is a target for composition with a similar named tag including the "@". Thus a responsibility marked <@MYTAG> composes with all responsibilities tagged with <MYTAG>; i.e. the goal associated with the '@' symbol is added to or changes the goal associated with the other responsibilities. This feature allows strict separation of cross-cutting concerns and subsequent re-composition. The semantic tags are created and managed by the analyst either manually or with tool support.

Any conflicts that emerge through this composition will need to be resolved by design or by negotiation.

### 3.6. Accelerating Functional Goal Sketching

In [7] and [8] we mention problems that people may experience with functional goal refinement: for example the tendency to interpret 'how' as project flow and elaboration that is unjustified in the circumstances. In [8] we introduce the idea of dual use of goal graphs and activity diagrams. The former give coherence and the latter facilitate refinement of functionality.

The approach depends on the idea that an activity diagram has a goal that is satisfied by its activities plus a special goal to guarantee its logic (guards, flow etc). Thus an activity diagram such as Figure 6can be said to have an objective GO and will be a goal proposition of type /b/. Similarly the objectives of the activities A1, A2 and A3 are G1, G2 and G3. This gives the corresponding goal graph shown in Figure 7.

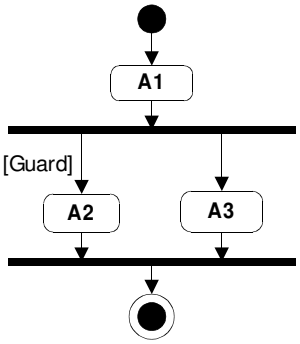


**Figure 6. An activity diagram representing a /b/**

It is important to note the goal in Figure 7 'Impose Process A' as a /b/ type proposition. Its purpose is to represent the need to guarantee the flow of the activity diagram as a leaf goal in the structure. If the activity diagram is informally drawn then the logic to be guaranteed in 'Impose Process A' can be correspondingly informal (the use of such informal sketches is an area we are currently investigating).

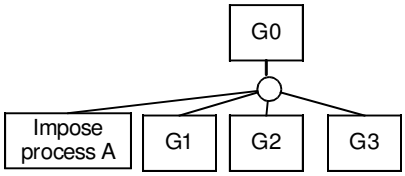


**Figure 7. Goal graph corresponding to Figure 6**

If any of the activities A1 to A3 in Figure 8 have sub-activities these are appended to their goals in Figure 6. In this way nesting of activities is a dual of goal refinement. This approach has been used in one of our industrial examples.

## 4. Example

To illustrate the technique we will use an example involving the calculation of body mass index.

> The *customer, WeighCom, wishes to develop new walk-on scales that can be installed in public places and used by any passers-by to measure their weight, height and body mass index (BMI) and receive a business card sized printed record on the spot. Normal operation is for the user to step onto a pressure mat facing an instruction screen and stand under an acoustic ranger. The measurements are made once the user pays a fee of 1 Euro into a receptor.*
>
> *WeighCom specifies that the solution must use certain components: pressure mat (PM); coin receptor (CR); acoustic ranger (AR) and integrated processor with alpha numerical visual display and user selection touch screen (IP). All of these are to be controlled through software using an API. These components support an existing assembly in which the whole is weather proof and vandal proof.*
>
> *WeighCom currently installs personal weighing equipment in public places for coin operated use by the public. They have an excellent reputation, which is of paramount importance to them, for always providing a reliable service or repaying. They have a call centre which customers can call if their installations appear to be malfunctioning.*

**Figure 8. Problem statement**

Scrutiny of the problem statement suggests the following primary concerns:-

- Operation in public places.
- Normal operation (i.e. accepting payment through to printing a card)
- Use of prescribed components.
- WeighCom's reputation.

From the problem statement we can also reasonably place these in context as shown in Figure 9 where we can see that there are likely to be concerns associated with the call centre. Further we might speculate that there is a maintenance problem domain for which we have no expressed concerns. Table 3 shows that we might associate the use of prescribed components to an additional problem domain concerned with the project but since there are no other concerns stated here we will ignore the project problem domain. We also have no express concerns about an installation problem domain; which would probably affect a maintenance domain. What matters is that we can agree with the stakeholders that Figure 9, with the attached assumptions, represents the problem under discussion.

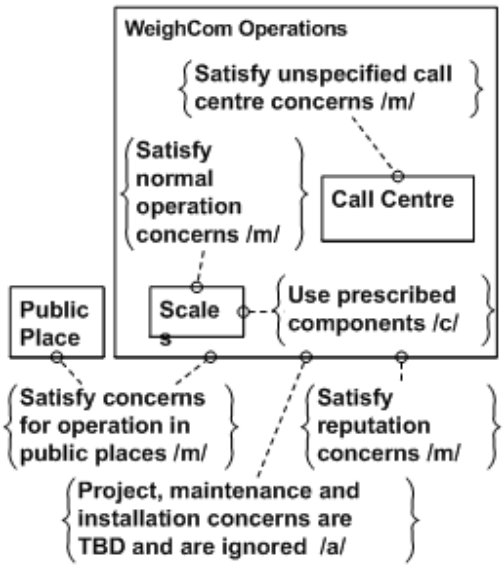


**Figure 9. Context of WeighCom goals**

The corresponding G-R modelling is shown in the first level refinement in Figure 10 where all the GOPs are owned by stakeholders.

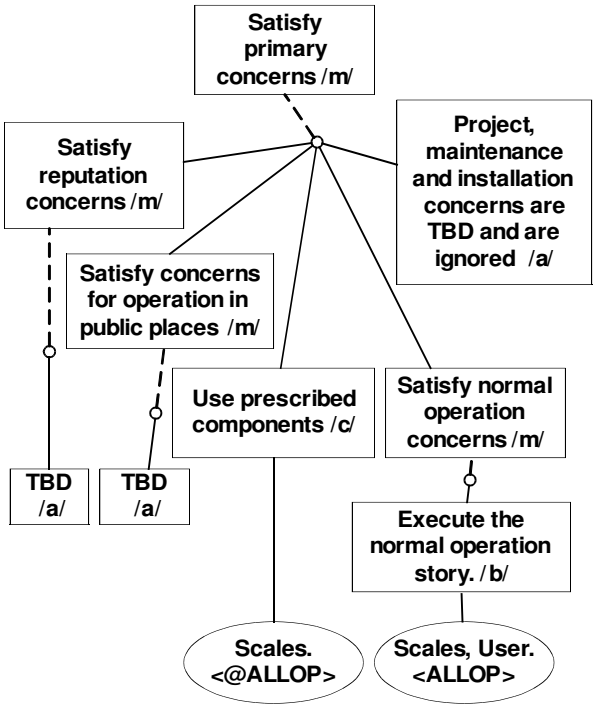


**Figure 10. Structurally complete G-R model**

If and when the assumption 'project maintenance..' is reversed a separate refinement argument would be created for other problem domains (project, maintenance etc.) and these are likely to crosscut as constraints on the responsibilities in Figure 10 .

Figure 10 has been made structurally complete by adding a lazy refinement ('execute..') and assumptions declared TBD. In this case the transition between the / m/ and /b/ goals has not needed a weltanschauung assumption as the /m/ goal itself applies to the scales domain in which all activity in the /b/ goal takes place. Note also that the responsibility to use prescribed components rests on the actors of the scales. The tag ALLOP was created, and catalogued, to refer to all normal operation behavior.

The refinement of the 'execute...' is an example of functional refinement. There are two potential problems (see Table 2) when creating a stable refinement argument at consistent levels of abstraction and getting business stakeholders to review the argument. This is a good opportunity to use the activity diagram technique. A plausible analysis is shown in Figure 11.

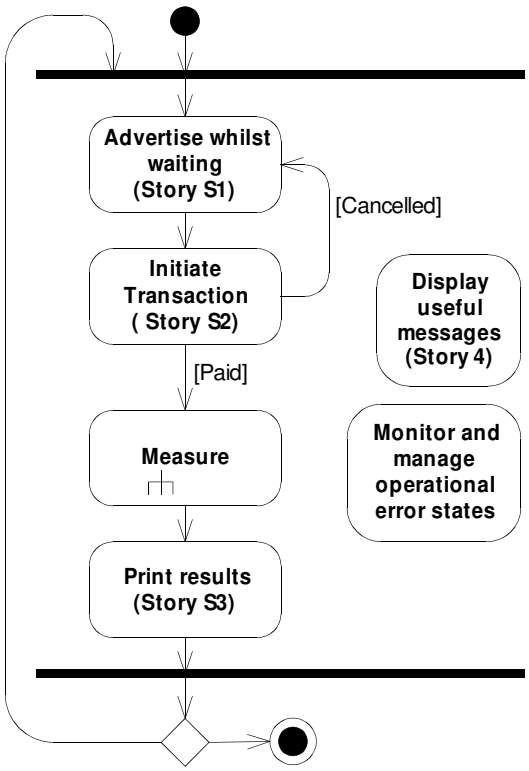


**Figure 11. Normal operation story as an activity diagram sketch**

Figure 11 itself is a sketch as there is no pretence at full rigour. However it is suitable for discussion with stakeholders to reveal the required activity. Thus the activity diagram allows the stakeholders to design a solution rationale. Each activity must be supported either by a specification (here stories have been used) or a further level of refinement (e.g. as in 'measure'). Our experience is that this approach is easier for stakeholders to comprehend than looking at mixed 'case' and 'milestone' KAOS refinements. The two floating activities are read as occurring concurrently with all other activities between the fork/join lines.

Figure 12 Shows all the activities in Figure 11 plus an 'impose..' goal as the refinement of 'execute..' in Figure 10.

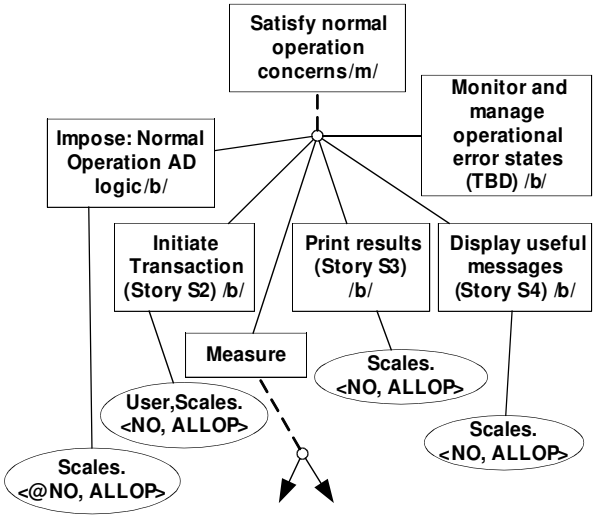


**Figure 12. Goal sub-refinement for Figure 11**

The 'impose process..' goal in Figure 12 emphasizes the need for the glue logic and can be developed to an appropriate level of precision (on a scale from leaving it to the developer's intuition, to detailed narrative, up to fully developed UML or formal logic). The 'measure' goal is further refined (not shown in detail here). The figure will be structurally complete provided that the refinement of 'measure' is actually complete and that the 'monitor and manage.' goal is replaced by an assumption that it is not to be implemented in the current stage. All these matters being negotiated and prioritized as apart of stakeholder negotiation for a stage of the development.

An example simple story is provided in Figure 13. The level of precision shown would be enough for many developments. If more precision becomes necessary then the story may be replaced with one in more detail, a use-case, a problem frame or by a full KAOS refinement.

*"When a customer pays €1 into the CR they may either confirm the payment or cancel the payment. If they cancel then the CR refunds the payment. If they confirm then the service is initiated."*

**Figure 13. Transaction initiation (Story S2)**

This simple example has allowed a demonstration of the techniques. In the next section concerning real projects we can observe how well we meet the objectives set down in Table 1.

## 5. Industrial Projects

We have improved our method using a number of industrial applications. These include products supported by venture capital, a management information system (MIS) for a food processing company, a university infrastructure project and support for services in healthcare. We start here with some general observations and then look at some details of a healthcare project and a venture capital project. The first is chosen because it seems representative of the general method and the second because it makes a slightly different use of our method and shows a situation that often arises in agile backlog driven projects. Most of the projects have been mentioned in [7,8,9].

The staffing profiles for these projects involved managers, executives, developers and testers; all with very different perspectives and analytical abilities. In all cases the managers and executives were not involved with detailed requirements analysis, whereas the developers and testers were.

The analyst (one of the authors) worked with key staff members (project and/or product managers). From the beginning it was clear that our industrial colleagues were not familiar with goal based requirements methods. In order to reach out to the executive and other non-technical stakeholders, whose participation was essential, we developed an approach which used the familiar sales terminology: 'pain' (things that are presently unsatisfactory in the problem domain) and 'gain' (new opportunities to improve the problem domain ) [26]. To this we added 'maintain' (things that should not change as a result of dealing with the pain and gain concerns).

Thus armed, our first step involved analyzing the problem domain and the stakeholders' concerns (i.e. the root GOPs). Inevitably lower level concerns (design fragments, particular functions etc) arose but they were put aside until the root problem was agreed. One of our projects was a retrospective study and it is clear that the project lacked shared understanding.

We find that the cost of reaching an agreed problem domain and root GOPs is only a few staff days unless there are conflicts that need to be resolved. The smaller (health-care) projects took only a few hours to reach this point. Importantly what they all established firmly were the 'load bearing' assumptions.

Figure 14 illustrates the root problem for one of the healthcare projects. This project was motivated by a benefactor organization wishing to sponsor a tool to be supplied to assist the care of patients with a particular disease. A group of physicians (the Forum) were to be the initial beneficiaries. They would be called upon to help specify an initial product, limited by budget, and would use the product as a support to their normal consultations and supplementing their usual medical system (MedSys).

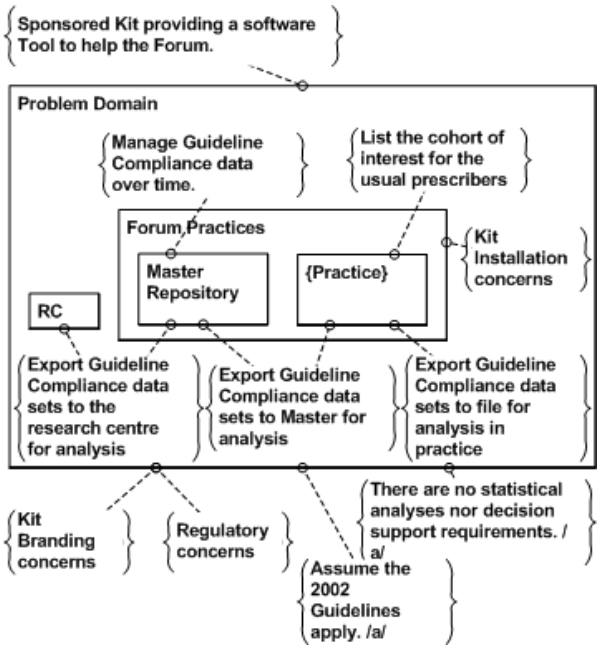


**Figure 14. The agreed problem statement**

The figure shows a key simplifying assumption arrived at after negotiation and constraints arising from data confidentiality and security protocols and from the wishes of the benefactor and Forum to have their roles acknowledged in product branding (logos and style etc). All these concerns attached to the outside of the problem domain box affect (cross-cut) everything inside the box. A research centre (RC) and the Forum Practices are the principal sub-domains of the problem domain and inside the Forum Practices are a Master repository (in one of the practices) and the MedSys and medical staff sub-domain (in all of the practices as

indicated by notation {Practice}). Inside the problem domain box are more localized concerns including a concern about 'Kit Installation'; an example of a domain context that has a different time-scale to the normal system-operation (see Table 3).

Figure 14 was constructed on the basis of a two hour discussion between stakeholders and remained stable throughout the development. It satisfied our third objective (Table 1)and laid a foundation for the first. In our earliest attempts at goal sketching we did not realize the importance of first obtaining an agreed problem domain and concerns diagram and invariably paid the price of taking much longer to establish root GOPs and this compromised our second objective.

After achieving Figure 14 it was a straightforward process to finish a structurally complete GR model by pair sketching and cross-checking with the stakeholders. We proceeded rapidly to a complete G-R model (taking about one day) but here whilst being confident that we are satisfying three of our objectives for goal sketching it must be noted that the third is challenged as it remained fully understandable only to a subset of stakeholders. The situation was remedied by a two stage process: (1) talk through the contents and (2) debate the correctness of the contents. Nevertheless the project manager could always use the representation to ensure that the right questions were asked and to ensure that the key assumptions were recorded.

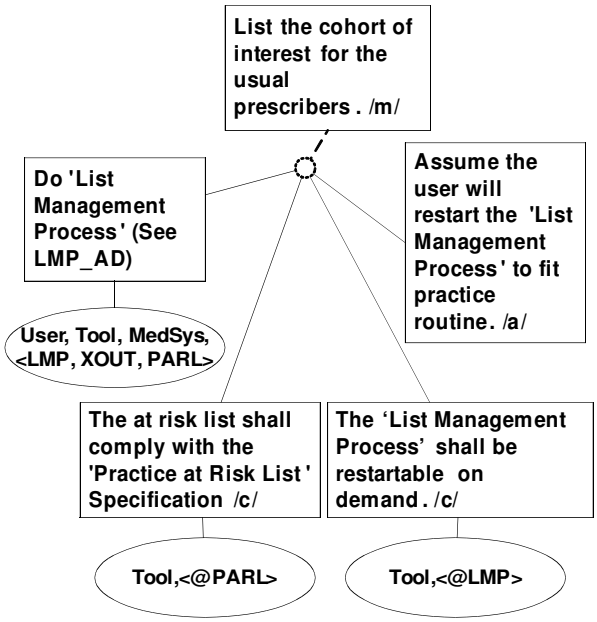


**Figure 15. Figure 2 G-R model for one concern of Figure 14**

Figure 15 illustrates one of the concerns from Figure 14and amounts to about 1/6th of the whole G-R model. It includes the responsibilities (which we usually only expose to the technical stakeholders): The actor MedSys is a sub domain of the Practice Domain. The semantic tag XOUT reflects cross-cutting of the regulatory concerns. The PARL and LMP tags show constraints acting on the "Do 'List .." goal. This particular goal is interesting as it is an example of hiding a detailed refinement that was constructed and negotiated using the activity diagram approach shown in Figure 16. The full G-R accommodates this figure in the manner illustrated in the scales example above.

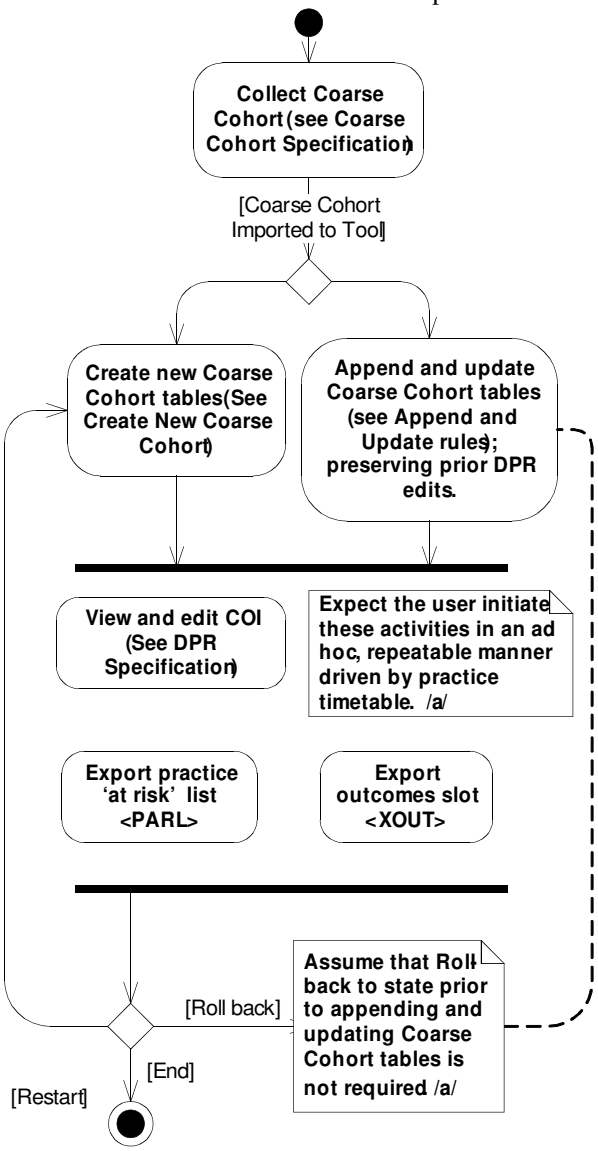


**Figure 16: List management process**

There is a significant simplifying assumption in Figure 16 agreed by all stakeholders for this stage of

development (to set aside roll-back). The need to surface such assumptions can easily be missed in less disciplined approaches. But where feasibility and adequacy are in conflict, as they were here, it is crucial to help the stakeholders make a decision. Figure 16 was reviewed on several occasions by the stakeholders; and thus improves our score on our third objective (Table 1).

Our tool support for our method allows us to annotate the G-R model. Included is a traffic-light annotation on each GOP to indicate our confidence in the refinement argument and/or its feasibility. Further we can export the leaves of the model into a project management spreadsheet to define the developments and procurements to be accomplished and the load-bearing assumptions to be monitored. This has been helpful to project managers.

In one of our venture capital supported development projects the main use of our method was to guide the development of acceptance testing. The test team found that working from a requirements backlog failed to provide sufficient understanding of the behavior that was being warranted by their product (see [8]. After several backlogs driven sprints the coherent picture of the intended user experience became unclear. This made test design very difficult and led to problems of product regression. The remedy was to use activity diagrams in the manner described here to reverse engineer the entire functionality of the product. This produced a set of four level nested activity diagrams upon which the acceptance tests could be designed. Converting these to a G-R model showed that they needed to pay more attention to the 'Impose Logic' goals described in previous sections. It also allowed the cross-cutting effects of the non-functional requirements to be included systematically in the tests. Recently the company has applied formal inspections, guided by the G-R model, to guarantee that the activity diagrams comply with all engineering and product management stakeholders' expectations. We will report further on this separately.

## 6. Related Work

We have mentioned some related work in the introduction. In addition we comment here on related requirements engineering material.

Work has been done on how some of the best practices of requirements engineering could enrich agile approaches [27]. The practices described include customer interaction, requirements analysis, non-functional requirements and managing change. The paper suggests that ways of adapting requirements management practices for agile processes are needed. However note that [27] simply describes how to include requirements engineering methods in an agile development process, rather than describing a method for requirements engineering that is agile. Similarly Nawrocki et al propose a way in which documented requirements could be introduced into XP through the use of automated tools, the Web and on-line documentation [28].

Cao and Ramesh have reported on how agile requirements engineering differs from traditional requirements engineering [29]. Their study showed that the agile case is more dynamic and adaptive than the traditional.

Orr suggests that it is possible to combine requirements and agile development by using up-to-date hardware and sophisticated graphical software [30]. Prototypes are suggested as a way to improve the process of defining requirements. However this work emerged from practice rather than from a theoretical technique such as goal-oriented requirements engineering.

Ambler describes an agile approach to modeling requirements, utilizing approaches such as the planning game of Extreme Programming and the Scrum methodology [31]. Similarly Leffingwell and Widrig discuss an agile requirements technique that is based on use-case specifications [32]. They also provide guidelines for selecting which requirements method (extreme, agile, or robust) is right for a particular project. However, again these approaches do not have a formal method such as goal-oriented requirements as a basis.

## 7. Further Work

The work reported here concerns the basics of the goal sketching technique. We are undertaking the following investigations to advance the work:-

1. Application to more industrial projects to confirm the applicability and practicality of the method for use in Agile projects.
2. The relationship between SSM[18] and the problem of transforming stakeholders concerns into goals.
3. Development of tools to accelerate the speed of sketch drafting and refactoring. In this area we are currently exploring the use of UML diagrams such as activity diagrams as these are well suited to the problem of determining behavioral goal refinements.
4. Development of metrics and supporting tools to exploit the structure of goal graphs in conjunction with expert judgments to quantify the adequacy and

feasibility of the intention expressed in a goal graph. It is anticipated that this will contribute significantly to the better planning of project stages and the improved sharing of expectations amongst the key stakeholders.

5. Tools to export goals sketches into KAOS for cases that justify upgrading from a goal sketch to a rigorous KAOS analysis.

## 8. Conclusion

In this paper we began by observing the problem of helping stakeholders set realistic expectations and take decisions. The problem is particularly pronounced in agile projects but is not limited to them. We have proposed a disciplined method of goal responsibility modelling as the basis for supporting stakeholders but also argue that success depends upon a set of practical objectives. We have also presented a goal sketching technique aimed to satisfy these objectives. Our experience shows that goal sketching in its present state performs well against our objectives although more validation is still needed.

The emphasis of goal sketching has been to provide a disciplined method of appraising the validity of a set of requirements for a project. Our method can be used alongside other requirements methods (especially use-cases) and can play an important part in reinforcing the coherence of agile requirements engineering based on backlogs.

## 9. Acknowledgements

The authors would like to acknowledge their industrial collaborators. In particular: Nick Gradwell, Product Manager of ClearPace Ltd; Dr Steve Moyle and James Wilson of Secerno Ltd., Ian Lycett KTP Associate at Image Farm Ltd; and Sean O'Mahoney, Martin Roskell and Richard Olearczyk of Oskis Informatics Ltd.

## 10. References

[1] D. Nevo, and M. Wade, "How to avoid disappointment by design", Communications of the ACM, 2007, Vol 50, No.4 .

[2] A. Desilets, "The Agile Physician", letter, IEEE Software, vol. 24, No. 3, 2007.

[3] A. Dardenne, A. van Lamsweerde, and S. Fikas, "Goal-Directed Requirements Acquisition", Science of Computer Programming Vol. 20, pp. 3-50, North Holland., 1993, pp. 3-50.

[4] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, Boston, 2001.

[5] I.F., Alexander, and R. Stevens, *Writing Better Requirements*, Addison Wesley, 2002.

[6] A.van Lamsweerde, "Goal-Oriented Requirements Engineering: A Round trip from Research to Practice", 12th IEEE International Requirements Engineering Conference (RE'04), 2004.

[7] K. Boness, and R. Harrison, "Goal Sketching: Towards Agile Requirements Engineering," ICSEA, International Conference on Software Engineering Advances (ICSEA 2007), 2007, pp.71-6.

[8] K. Boness, and R. Harrison,"Goal Sketching with Activity Diagrams," ICSEA, International Conference on Software Engineering Advances (ICSEA 2008), 2008

[9] K. Boness, and R. Harrison, and A. Finkelstein, "A lightweight technique for assessing risk in requirements analysis", Software, IET, 2008, Volume: 2, Issue: 1 pp. 46-57.

[10] I. Sommerville, and P. Sawyer, *Requirements Engineering – a good practice guide*, Wiley, Chichester. 1997.

[11] S. Adolph, and P. Bramble, *Patterns for effective Use Cases,* Addison-Wesley, 2003.

[12] A.M. Davis, *Just Enough Requirements Management*, Dorset House Publishing, New York, 2005.

[13] T.P. Kelly, and R.A. Weaver, "The Goal Structuring Notation –A Safety Argument Notation." Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004.

[14] L. Rising, and N. Janoff, "The Scrum Software Development Process for Small Teams," *IEEE Software* July/August 2000.

[15] B. Boehm., "A Spiral Model of Software Development and Enhancement", *Computer*, May 1988, pp. 61-72.

[16] M. Jackson, Problem *Frames: Analysing and Structuring Software Development Problems,* Addison Wesley, 2000.

[17] J. March, and H.A. Simon, *Organisations*, New York: Wiley, 1958.

[18] P. Checkland, and J. Scholes, Soft *Systems Methodology in Action,* John Wiley and Sons, 1990.

[19] J. Gibson, "The Theory of Affordances". In Perceiving, Acting, and Knowing, Eds. R. Shaw, and J. Bransford, ISBN 0-470-99014-7. 1977.

[20] IEEE, "IEEE Guide for Developing System Requirements Specifications" IEEE Std-1223 (1998).

[21] J.A. Dewar, and C.H. Builder, et al., "Assumption-Based Planning: A Planning Tool for Very Uncertain Times", Santa Monica, RAND. 1993.

[22] I. Alexander, "A Taxonomy of Stakeholders," Int'l J. Tech. and Human Interaction, vol. 1, no. 1. 2005.

[23] S.J. Bleistein, K. Cox, and J. Verner, "Requirements Engineering for e-business Systems: Integrating Jackson Problem Diagrams with Goals Modelling and BPM", Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC'04) IEEE, 2004.

[24] W. G. Vincenti, "What Engineers Know and How They Know It:", Analytical Studies from Aeronautical History: The Johns Hopkins University Press, 1990.

[25] T. Gilb, *Principles of Software Engineering Management*, Addison-Wesley, 1988.

[26] S. Deep, and L. Sussman, *Close the Deal: 120 Checklists for Sales Success,* Sandler Institute.

[27] A. Eberlein, and J. Cesar Sampaio do Prado Leite, "Agile Requirements Definition: A View from Requirements Engineering", International Workshop on Time-Constrained Requirements Engineering TCRE'02, Essen, Germany, Sep, 2002.

[28] J. R. Nawrocki, M. Jasiński, B. Walter, and A. Wojciechowski, "Extreme Programming Modified: Embrace Requirements Engineering Practices", Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering (September 09 - 13, 2002). RE. IEEE Computer Society, Washington, DC, 303-310.

[29] L. Cao, and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study". IEEE Software. 25, 1 (Jan. 2008), 60-67. 2008.

[30] K. Orr, "Agile Requirements: Opportunity or Oxymoron?" *IEEE Software, 21, 3* (2004), 71-73.

[31] S.W. Ambler, *Agile Modelling: Effective Practices for eXtreme Programming and the Unified Process*, John Wiley & Sons, 2002.

[32] D. Leffingwell, and D. Widrig, 2003 *Managing Software Requirements: a Use Case Approach*. 2. Pearson Education.