Departament de Llenguatges i Sistemes Informatics

UNIVERSITAT POLITÈCNICA DE CATALUNYA

**GESSI**

Grup d´Enginyeria del Software per
als Sistemas d´Informació

Claudia Patricia Ayala Martínez

# Goal-Based Reasoning in the Construction of Taxonomies for COTS Components

GUIDED RESEARCH PROJECT

# Director:

# Dr. Pere Botella, Dr. Xavier Franch

# Table Of Contents

# Abstract

In a previous work, our research group GESSI has proposed the construction of a taxonomy for classifying COTS components by means of characterization attributes to arrange domains which COTS components belong to, and also grouping these domains into categories [4]. In this report we present our first applicability study of GBRAM (Goal Based-Requirements Analysis Method) as a goal-based reasoning method for the construction of taxonomies of COTS components; more concretely for exploring the "characterization attributes" that are used to browse the taxonomy through an example: the context of the Application Development Tools. We illustrate the main aspects of customizing GBRAM to this objective and remark the methodological aspect we want to achieve in our future work: to propose guidelines for the construction of any taxonomy of COTS components.

# Introduction

The use of commercial off-the-shelf software components (COTS) is becoming an economic and strategic necessity for many organizations in a wide variety of different application areas, including finance, health-care, logistics, administration, manufacturing, commerce and many others.

However, employing COTS in building applications is not a painless business. As the COTS market evolves, COTS users must face new challenges to successfully and effectively acquire, select and integrate commercial software components in applications and systems. The statement of methodologies and the construction of tools to support them can be a key point to enhance:

a) The efficiency of the software process, through the reduction of production and maintenance costs.

b) The accuracy and reliability of the decisions relative to the acquirement, integration and use of applications in the most critical points of the target companies from a strategical perspective.

Therefore, there is an increasing need for arranging the types of available COTS products to improve the efficiency and reliability of selection processes. The need of having these taxonomies is evident; we can see that it arises in very different contexts, from huge consultant companies like Gartner [1], to organizations such as INCOSE [2], or websites as www.componentsource.com or eCots from Thales company [3].

In this context, we have detected that there are not studies deep enough about the identification of properties that can help to organize the taxonomies. As a result, our research group (GESSI -Grup d´Enginieria del Software per als Sistemas d´Informació-[1] at UPC) has been working in many aspects addressed to build a taxonomy for classifying COTS by means of characterization attributes; this work is described and applied in [4]. The existence of such taxonomy provides a framework for the whole selection process and is a basis for structuring a knowledge base on the field (these aspects are presented in next sections).

Nevertheless, more important than the concrete form that a taxonomy takes, is the rationale behind its construction, i.e. which are properties that may help to arrange it and how the taxonomy can be searched (this is especially true when considering not just the construction of the taxonomy, but its evolution). In this sense, the remaining sections show the context and our first approach in a method for enhancing the mechanism to build the taxonomy by means of goal-based

[1] Software Engineering for Information Systems Group.

reasoning. Last, we close the report with a discussion and explanation of our overall idea and outline the need of future research to achieve our objective.

# 1. A Taxonomy For COTS Products

The proposal of building taxonomies made by GESSI in [4] is an example of the specific support for improving the Business Applications (BA) selection processes.

It is considered that BA belong to one or more BA domains, which appear as leafs in the taxonomy. A domain encloses a significant group of functionality. Domains are grouped into categories, and also categories can be grouped themselves to form a multi-level taxonomy. Dependencies among domains that belong to the taxonomy are included in the hierarchy itself and are represented using i* SD notation [6]. The taxonomy can be incorporated to COTS selection practices, which are based on the use of quality models to assess the adequacy of components with respect to requirements [7,8]. Those quality models are attached to nodes in the taxonomy, supporting model reuse by inheriting quality models downwards the BA hierarchy [9] (see Figure 1.1).

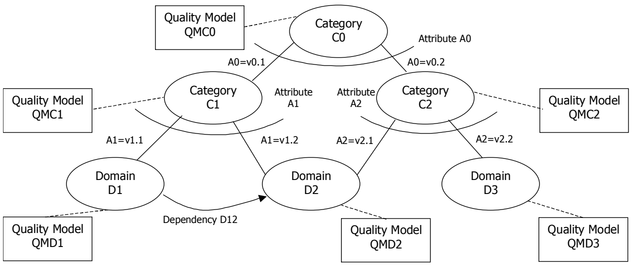


Figure 1.1 The fundamental elements of a taxonomy

## 2. Goal Based Reasoning

Requirements Engineering (RE) research has increasingly recognized the leading role played by goals in the RE process [10]. Such recognition has led to a whole stream of research on goal modeling, goal specification, and goal-based reasoning for multiple purposes, such as requirements elaboration, verification or conflict management, and under multiple forms, from informal qualitative to formal.

Goal-based reasoning reviews how goals are used in basic activities such as requirements elicitation, elaboration, verification, validation, explanation, and negotiation; particularly for difficult aspects such as conflict management, requirements deidealization, and alternative selection [10].

We are exploring the applicability of GBRAM [5], as a goal-based reasoning method, in a different context from those cited above, that is the construction of taxonomies from goals (specifically in the Application Development domain –in this report-). In this sense, we are working for defining a guide of goal-based reasoning, which will help us to generalize, formalize, enhance and clarify the process for building taxonomies; more concretely to help in the process of identify and evaluate the most suitable characterization attributes, including the question-answer pairs for the construction of the optimal taxonomy in any specific area.

In particular, we emphasize the concept of characterization attribute as a rationale for building this taxonomy in any category of COTS, together with question-answers and examples assets [4], making their use easier and dynamic.

With the primary focus on the transformation of enterprise and system goals into requirements, GBRAM was formulated by Annie I. Antón [11], more specifically to assist analysts in gathering software and enterprise goals from many sources and to support the process of discovering, identifying, classifying, refining and elaborating goals into operational requirements. The method's chief contribution is the provision of heuristics and procedural guidance for identifying and constructing goals.
The high level activities of GBRAM briefly explained are:

- Goal Analysis that concerns the exploration of available information sources for goal identification followed by the organization and classification of goals.

- Goal Refinement that concerns the evolution of goals from the moment they are first identified to the moment they are translated into operational requirements for the system specification. It includes activities as refine, elaborate and operationalize of goals.

These high-level activities just described provide an overview of the GBRAM.

The suitability to apply this method to our purpose of obtaining the characterization attributes is based on these facts:

a) It provides guidelines and heuristics for exploring, identifying, and organizing goals (potential characterization attributes) which guide us towards a high probability of success while avoiding wasted efforts.

b) It offers a guide for applying an inquiry-driven approach to goal-based analysis, that can be useful for to enhance our questions-answers mechanism linked to mostly characterization attributes [4].

c) It assumes that goals have not been previously documented or explicitly elicited, so we must work from all available sources of information to determine the goals. This is a helpful aspect because in our context, we have much diverse information related with the domain that we want to create the taxonomy.

## 2.1. Customizing GBRAM to our approach

We cited above that GBRAM was conceived for improving the early stages of the software development process in complex environments, because its output is the software requirements document and its inputs are diverse sources of information (e.g. interview facts, interview transcripts, policies, requirements, textual statements, charts, diagrams, corporate goals, mission statement, etc.).

Therefore that we adapt some issues from the original method customizing them to our approach. We adjust the inputs (all information related with the domain, e.g. existent taxonomies, standards, etc.), and modify the output. Certainly we are working in adapting heuristics and activities of analysis and refinement from the original method, treating the issues not as operational requirements of a system but characterization attributes in order to obtain as output the taxonomy of COTS.

Figure 2.1 shows the activities which are involved in the applying of GBRAM to our approach.

The box in the top left corner contains the possible inputs, which may vary in accordance with the documentation available. We suggest that at least some of the next sources should be exist:

- ✓ **Domain Information**: Is important to gather as much relevant information as possible to understand the design implications of goals. These information sources or descriptions may be provided in such diverse formats as textual statements, transcripts of interviews, diagrams, process descriptions, or even explicitly stated goals. It is important to remark that the i* diagrams of the domain are an important issue that can be very helpful, they are constructed after some information of the domain was gathered, and show a high level picture of the domain in order to represent and organize the domain knowledge and the related activities. In the strict sense they are a product of the activity of Goal Analysis (because they come from gathered information for understand the domain) , but we take as Input due to its purpose that is organize the information in order to guide us to understand the domain and facilitate the Goal Analysis stage.

- ✓ **Diverse Existent Taxonomies**: In the exercises that we have realized, we take into account existing taxonomies (given by any consultants or organizations, i.e. Gartner, INCOSE or IDC), because they are a good knowledge base for a deep understanding of the domain and, knowing the existence and trends of products in the market.

- ✓ **Related Standards**: Due to the methodological level that we want in the taxonomy to be built, is necessary taking into account the available standards related with the domain.

- ✓ **Other Sources**: Many sources should be taken into account, from definitions or activities related to the domain to information about available products in the market for the domain, vendors, forums, etc.

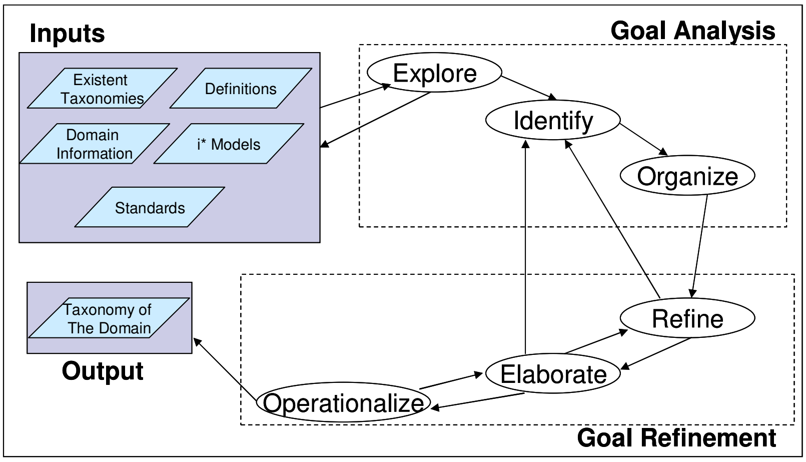


Figure 2.1 Overview of GBRAM in our context

The ovals located within the dotted box on the upper right corner of the figure denote the goal analysis activities. The goal analysis may be summarized as follows:

- *Explore* activities entail the examination of available information.

- *Identify* activities entail extracting goals applying heuristics.

- *Organize* activities involve the classification and organization of goals according to goal dependency relations. This aspect is very important, in our approach these dependency relationships are expressed in i* notation [6].

The ovals within the dotted box on the lower half of the figure denote the activities that take place during goal refinement. The goal refinement activities may be summarized as follows:

- *Refine* activities entail the actual pruning of the goal set.

- *Elaborate* refers to the process of analyzing the goal set by considering possible goal obstacles and constructing scenarios to uncover hidden goals and requirements.

- *Operationalize* refers to translating goals into requirements that can be represented as characterization attributes, and then as a hierarchy that is the taxonomy itself.

## 2.2. Application of Heuristics

Obtaining characterization attributes in the taxonomy proposal by our group in the context of Business Applications [4] was empirical; therefore we are exploring the use of GBRAM heuristics to enhance this process.

GBRAM heuristics aid us by providing prescriptive guidance for managing varying levels of detail in the information available. There are four general types of heuristics used in GBRAM: identification, classification, refinement, and elaboration heuristics. Some of them are straightforward and generic, not require employing a specific inquiry technique. Others make sense only in conjunction with specific questions about the system.

Many heuristics showed in GBRAM can be mapped directly to our subject, but many others should be more adjustable, and also some new heuristics for the specific domain can be created which should be documented for the growing and evolution of the taxonomy.

Applying our customization of GBRAM we can achieve a high probability of success finding the characterization attributes in a more formal way while avoiding wasted efforts.

Two tables are presented in order to give a general idea of the applicability of the GBRAM in our purpose by means a simple example.

Table 2.1 includes a few identification heuristics for goals ("HIG" heuristics) and constraints (the "HIC" heuristics) from GBRAM.

Table 2.2 contains an excerpt of the characterization attributes related with Business Applications (labeled as Attribute in the table) obtained empirically in [4], their corresponding values, their questions and answers. This table represents information that we want to obtain as the output of GBRAM. The purpose of the example is only to show how we can get in a formal way through GBRAM requirements that we obtain empirically.

| Cod. | Heuristic |
|---|---|
| HIG$_1$ | Goals are named in a standardized subset of natural language, in which the first word is a verb that describes the target of the goal. For example, AVOID is used to name goals that are satisfied as long as the target condition is false. |
| HIG$_9$ | Stakeholders tend to express their requirements in terms of operations and actions rather than goals. Thus, when given an interview transcript, it is beneficial to apply the action word strategy to extract goals from stakeholders´ descriptions. |
| HIC$_2$ | Constraints can be identified by searching for temporal connectives (i.e. during, before, after, etc.). Restate statements that describe when some condition is true or when a goal can be completed as a constraint. |

Table 2.1. Example GBRAM heuristics for identifying and analyzing goals and constraints.

| Level | Cat. | Attribute | Values | Question | Answers |
|---|---|---|---|---|---|
| 1 | Root | Number of users | Single user, Multi-user | How many users has the system? | One, More than one |
| | | | | Does the system reconcile the interests of many stakeholders? | No, Yes |
| 2 | a | Objective | Management, Operational | Is for management or operation? | For Management, For Operation |
| 3 | c | Orientation | Data, Process | Is it data or process oriented? | Data-oriented, Process-oriented |
| 4 | … | Data processing | Acquisition, Storage, Preparation, Analysis | What type of data processing does it perform? | Acquisition, Storage, Preparation, Analysis |
| 3 | d | Utility | Technical, Office | Is it technical or office oriented? | Technical-oriented, Office-oriented |
| 2 | b | User's location | Internal, Internal &External | Where are the users located? | Just inside the company, Inside & outside the company |
| 3 | e | Orientation | Data, Process | Is it data or process oriented? | Data-oriented, Process-oriented |
| 4 | f | Data type | Operation, Support | What type of data does it process? | Operation, Support |
| 4 | g | Type of group work | Coordination, Communication | What is the use of the system? | Coordinate teamwork, Communicate people |
| 3 | … | User's role | B2B, Customer | Is it for suppliers or customers? | For suppliers, For customers |

Table 2.2 An excerpt of the attributes, question and answers to browse the taxonomy of Business Application.

Heuristic HIG9 (Table 2.1) discusses stakeholders´ tendencies to express requirements in terms of operations and actions. Thus, in an interview fact stakeholders expressed their goals for BA in terms of the functions the tool must support. After some activities of refine and organize these were categorized into four: Acquire, Store, Prepare and Analyze, then successively applying corresponding heuristics for classifying, refining, and elaboration we obtain the suitable characterization attribute (in this case "Data Processing") the same obtained before empirically.

We want to outline that obtaining requirements guided by customized activities and heuristics of GBRAM is firstly more formal and confident, and secondly has a high probability of success while avoiding wasted efforts than the empirical work.

# 3. Case Study. Software Requirements Area

Our purpose in this section is to provide the reader with a general idea of the activities that we should perform applying GBRAM [11]. We notice the core issues related with to adapt the method. In this report we show as an excerpt, the Software Requirements Area to exemplify the applicability process.

## 3.1. Goal Analysis Activities

In this section is presented the activities that we must perform in GBRAM during Goal Analysis, and also report the most information we gathered concerning with our purpose and give you an overview of some of the key concepts involved in the software application development process (specifically in Software Requirements).

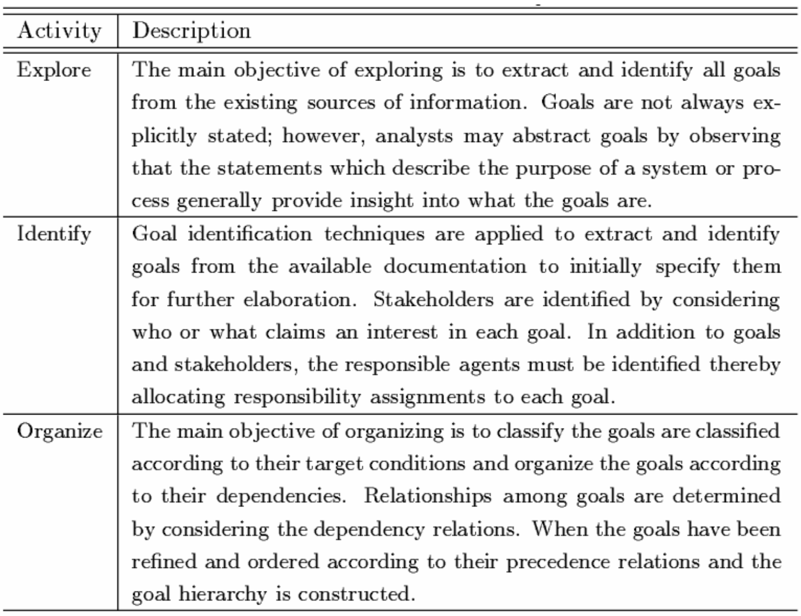
Due to our approach is based in three important sources:

- The information related with the domain and standards (i.e. IEEE, EIA) about Application Development in Software Engineering [14], [15], [16], [18], [19], [20], [23], [24] (see appendix A and B).

- The diverse taxonomies that many consultant and organizations provide in that field [1],[2],[3],[12] (see appendix B).

- The availability of products in the domain (e.g. vendors like Rational). [13],[21],[22].

We first present the standardized information (such as definitions, software engineering process, methods, and specific activities related with it –based on SWEBOK- it will also give us an understanding of the framework), then we show all the information related with the existent taxonomies by consultants and organization companies, latter we will use that information for matching the goals that will be used as inputs in the process of Goal Refinement for getting the most suitable characterization attributes with GBRAM.



| Activity | Description |
|---|---|
| Explore | The main objective of exploring is to extract and identify all goals from the existing sources of information. Goals are not always explicitly stated; however, analysts may abstract goals by observing that the statements which describe the purpose of a system or process generally provide insight into what the goals are. |
| Identify | Goal identification techniques are applied to extract and identify goals from the available documentation to initially specify them for further elaboration. Stakeholders are identified by considering who or what claims an interest in each goal. In addition to goals and stakeholders, the responsible agents must be identified thereby allocating responsibility assignments to each goal. |
| Organize | The main objective of organizing is to classify the goals are classified according to their target conditions and organize the goals according to their dependencies. Relationships among goals are determined by considering the dependency relations. When the goals have been refined and ordered according to their precedence relations and the goal hierarchy is constructed. |

Table 3.1 Overview of GBRAM Analysis Activities

### 3.1.1. Exploration of Existing Information

Using GBRAM, we must first explore the available information to identify and extract goals from these sources. We noted that the method is not dependent upon any specific representations. Instead, GBRAM recognizes the typical challenge of working with different sources of knowledge that are represented in different forms.

It is good practice to gather as much as relevant information as possible to understand the design implications of goals; thus, in GBRAM we explore these information sources to identify and extract goals. An important aspect in this our approach is that most information presented come from standard information (i.e. SWEBOK and IEEE standards), thus the level of confidence of the goals is based on them.

As much as possible information was gathered. In the Apendix A and B are related many of this information (but many other are only referenced).

***Definitions and Domain Information***: were gathered from diverse sources as web pages from consultants, organizations, standards or software engineering books. It provide us a base knowledge of the domain.

***Existent Taxonomies***: we take into account proposed taxonomies like: Gartner, INCOSE and IDC. For each of them we compile all information related with terminology and classifying criteria. It provides us diverse point of views for classifying and remarks certain trends of the actual market.

***Related Standards***: we are based many information on SWEBOK, because it represent the most standard information in the field and summarize IEEE standards related with the area (e.g. ISO/IEC 12207, IEEE/EIA 12207.1-1997, IEEE Std 830-1998)

***Other sources***: In one hand an significant issue that we want outline is that the information of principal vendors provide us insights of products characteristics and boundaries, . On the other had, the construction of i* models of the domain represent a big picture of the goals concerned to achieve the main objectives so they are helpful in organize the information in order to guide us to understand all as possible the domain and facilitate the Goal Analysis stage.

| Activity | Inputs | Outputs |
|---|---|---|
| Explore | Standards related<br>Domain Information | Goals |
| Identify | Standards related<br>Domain Information | Goals<br>Stakeholders<br>Agents<br>I* models |
| Organize | Goals | Matching of Information Sources<br>Reducing or Extend Goal Set<br>Goal hierarchy |
| Refine | Goal Set | Scenarios<br>Constraints |
| Operationalize | Goals Set | Characterization Attributes for Constructing the Taxonomy of the domain |

Table 3.2 Input and Outputs of adapted GBRAM activities.

For approaching the problem of creating a taxonomy for Application Development Domain, we partitioned the topic into the 10 Knowledge Areas referenced in SWEBOK (see Figure 3.1).

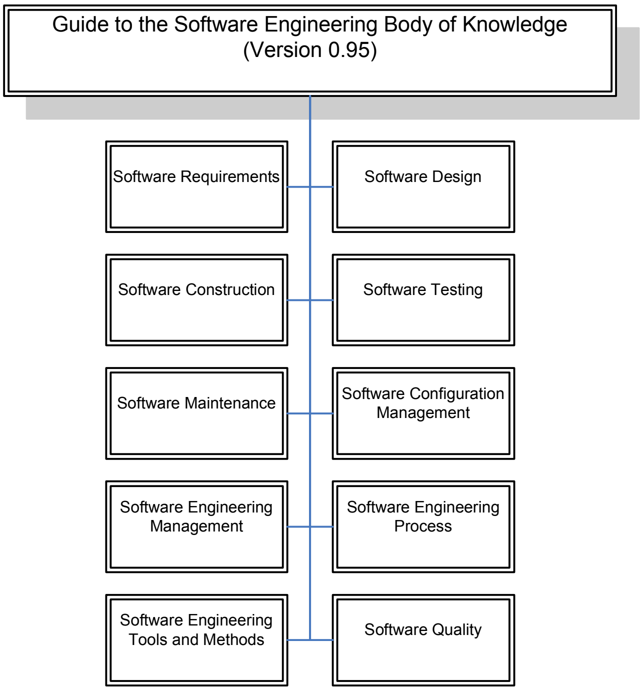

Guide to the Software Engineering Body of Knowledge
(Version 0.95)
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Tools and Methods
Software Quality

Figure 3.1 SWEBOK Knowledge Areas

In this report we take as a Case Study the Software Requirements Area (partitioned into 6 subareas, showed in Figure 3.2)

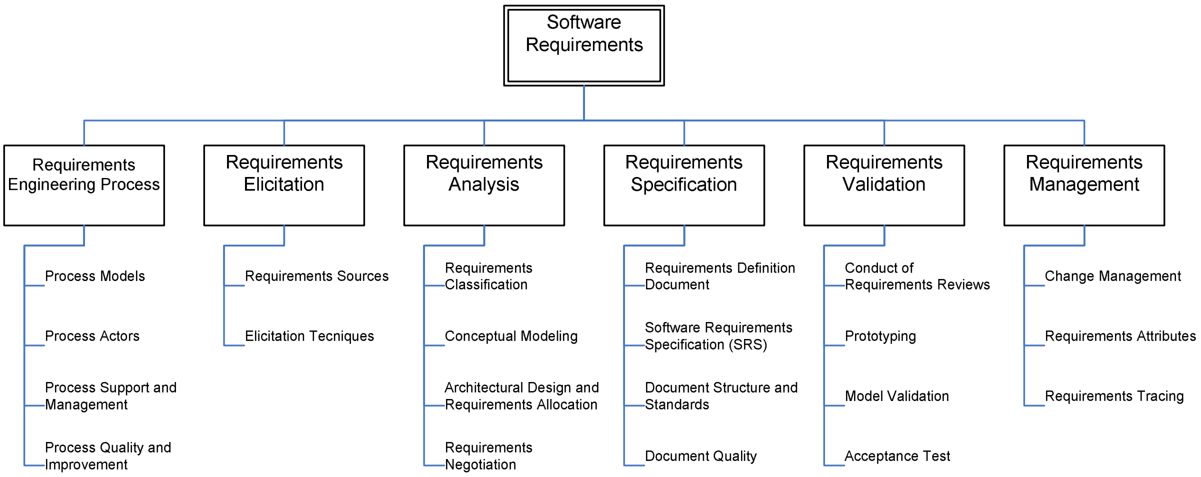


Figure 3.2 SWEBOK Software Requirements Area

It is important to stand out that in the nature of the problem we have 3 main information sources that we should to match for creating a set of goals.

### 3.1.2. Identifying Goals and Objectives

We have to report that in SWEBOK were identified the most important goals consensually validated (it has an standard level in the field), so it facilitated all the process due to these goals were supported by an standard. Thus, we worked in finding goals behind this high level goals established.

This is, SWEBOK present a hierarchical structure. Software Requirements Area is divided into 6 topics (we will take them as a high level goals): Requirements Engineering Process, Requirements Elicitation, Requirements Analysis, Requirements Specification, Requirements Validation, and Requirements Management. It follows that we have to find through GBRAM adapted activities the goals into this partitions.

For example: *Requirements Engineering Process* (this partition shows how Requirements Engineering is related with the overall software engineering process) has as subgoal *Process Support and Management* (introduces the project management resources required and consumed by requirements engineering process) and we have to identify goals and objectives behind this.

To identify goals, each source of information is analized by asking "What goal(s) does this statement/fragment exemplify?". In GBRAM is necessary to define and to document very precisely any preconditions or posconditions on the goals. In our adapting of the method as a consequence of the application area (original GBRAM was used to discover system requirements and we are adapting it in order to find characterization attributes) is not necessary to be so rigorous in that sense, however obviously it is necessary to document the preconditions and posconditions with the aim of facilitate the process to stablish hierarchies assuring the process of  evolution or growing.

**Example 1**:  Consider the following NLD (Natural Language Description) from the description of the topic of Process Support and Management [SWEBOK].

> *"It is important introduces the project management resources required and consumed by the requirements engineering process -issues of cost, human resources, training and tools-"*

As a general rule, statements which seem to guide design decisions at various levels within the system or organization point to possible goals.

By examining the statement and asking "What goals does this fragment exemplify?" some goals become evident from the description:

- ✓ Getting Project Management Resources Required and Consumed by Requirements Engineering.

- ✓ Getting costs of human resources, training and tools of the activities of Software Requirements.

The level of decomposition of goals is not defined, but it depends on the matching of the three main information sources  -described in section 3.1- (i.e. one goal that appear as "standard" should be take into account if exits in the market any tool that supports it, however discovering important goals that are not covered by any tool is a significant issue in closing the gap between tools and process).

All action words are possible candidates for goals. Goals may thus also be identified by searching for action words which point to some state that is, or can be achieved once the action is completed.  Certain types of verbs such as: allocated, completed, achieved, found out, and satisfied intimate possible goals.

To demonstrate the 'action word identification' approach, consider the next example:

**Example 2**: Requirements Analysis has as subgoal Architectural Design and Requirements Allocation.  From its description we have:

> *"In many cases, the requirements engineer acts as system architect because the process of analyzing and elaborating the requirements demands that the subsystems and components that will be responsible for satisfying the requirements be identified. This is requirements allocation."*

So, we decide to show the goal of Requirements Allocation as an independent goal.

Although goal identification is presented prior to discussion of stakeholder identification in this section, the activities do not preclude each other.  Stakeholders must often be identified before any goals can be specified. We must understand who the stakeholders are before we can even begin to develop an understanding of the goals. However, it is evident that in most cases that are on target, we depart from an understanding of who the general stakeholders are for the system prior to goal identification phase.

### 3.1.3. Identifying Stakeholders and Agents

In this stage, we have to determine who are the stakeholders involved in the achievement of goals, once the goals and stakeholders are specified, the goals must be assigned to their responsible agent(s).

Identifying stakeholders determines who or what claims an interest in each goal so that an understanding may be gained regarding the different viewpoints.

A stakeholder is any representative affected by the achievement or prevention of a particular goal. Multiple stakeholders may be associated with one goal.

Agents are responsible for the completion and/or satisfaction of goals within an organization or system.

For clarification, the difference between an agent and a stakeholder should be noted. As shown in Figure 3.3, some agents are stakeholders and some stakeholders are agents; that is Agents ∩ Stakeholders. A stakeholder may be a customer[2], actor[3], owner[4], or representatives of organizations. The agents that lie outside of the intersection of Agents and Stakeholders are not stakeholders; instead, they are system-specific agents.

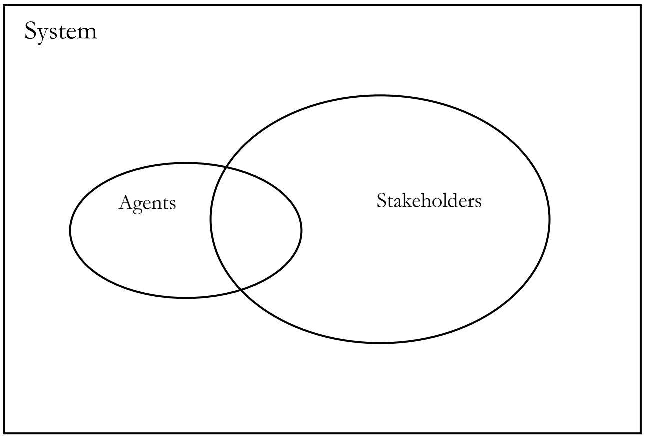


Figure 3.3. Venn Diagram Distinguishing Agents and Stakeholders

[2] A customer is a perceived beneficiary of the system.

[3] An actor is someone who actually performs functions in the system.

[4] An owner is a customer in the contractual sense.

The stakeholder for each goal are determined by asking "Who or what claims a stake in this goal?" and "Who or what stands to gain or lose by the completion or prevention of this goal?"

A logical approach for identifying the agents is to consider which agents are ultimately responsible for the achievement or maintenance of each goal by asking the question, "Who or what agent [is/should be/could be/] responsible for this goal?".

**Example:**

The goal Process Supported and Managed (from Requirements Engineering process defined) has related two goals (G1 and G2); Table 3.3 shows that both the Requirements Engineer (RE) and Project Manager (PM) have an interest or stake in the support and management of the Requirements Engineering Process. However, the RE is the responsible of realize that goal.

Here we can note the importance of the stakeholders dependencies that we have to represent through i* diagrams that facilitate the process of discover not only dependencies, but also Stakeholders and Agents.

| Goals | Agent | Stakeholders |
|---|---|---|
| G1: Resources required and consumed by requirements engineering process defined | RE | RE, PM |
| G2: Cost, human resources, training and tools stablished. | RE | RE, PM |

Table 3.3 Stakeholder and Agent Analysis Example

### 3.1.4. Organization and Matching of Goals

Organization of goals entails eliminating redundancies and reconciling synonymous goals.

It is useful first organize information related to each source, specify the goal dependencies and reconciling synonymous; then to do the match between them and apply again the goal dependency and reconciling synonymous.

As mentioned above, the i* diagrams are useful to organize the information and give us a framework of the domain, activities and stakeholders involved.

In Table 3.4 we can see the process of organization and classification of goals applied to the Requirements Engineering information framework, and in Table 3.5 we observe the matching with the existent taxonomies and vendors information.

It is essential to stand out that in the context of the existence taxonomies related, we have to do previously a simple inspection to the characteristics that were taken into account for their construction, this for knowing what taxonomies are suitable to consider or not. Then analyze all information related with it, to gather and investigate all the concepts that are used. These activities aid us to understand and to match not only the terminology, but also to have criteria of what taxonomies and in what measure should be applied.

In GBRAM, all goals are classified as either maintenance goals or achievement goals, but in our adaptation of the method (and the exercises realized until now) is not necessary and we can disregard this activity.

# Software Requirements Process

| Goals | | | | Description | Agents | Stakeholders |
|---|---|---|---|---|---|---|
| G1 | | | Process of Software Requirements defined | Showing how requirements engineering dovetails with the overall software engineering process | Requirements Engineer (RE) | RE, Project Manager (PM), Q.A. |
| | 1 | | Process Models defined | How the activities of elicitation, analysis, specification, validation and management are configured for different types of projects and constraints | | |
| | 1 | 1 | Description of how Requirements Engineering activity will be done and refined through the life cycle of the software process | | RE | RE, PM |
| | 1 | 2 | To define how to identify requirements as configurations items,and manage under the same configuration regime as other products of the development process | | RE | RE, SCM |
| | 1 | 3 | suitable process to the organization and project context tailored | | RE | RE, PM |
| | 2 | | Process Actors defined | Introduces the roles of the people who participate in the requirements engineering process | RE | |
| | 2 | 1 | Users, Customers and Stakeholders (intern and extern) defined | | RE | RE |
| | 3 | | Process Support and Management | Introduces the project management resources required and consumed by requirements engineering process. It make the link from process activities identified in (G1.1) to issues of cost, human resources, training and tools | | |
| | 3 | 1 | Expected project management resources required and consumed by requirements engineering process defined | | RE | RE, PM |
| | 3 | 2 | Issues of cost, human resources, training and tools linked to activities of process model. | | RE | RE, PM |
| | 4 | | Process Quality and Improvement | Its purpose is to emphasize the key role requirements engineering plays in terms of cost, timeliness and customer satisfaction of software products. | | |
| | 4 | 1 | Requirements Engineering covered by process improvement standards and models | | RE | RE, QA |
| | 4 | 2 | Requirements Engineering measures and benchmarking stablished | | RE | QA |
| | 4 | 3 | Planing and Implementation improvemented | | RE | QA |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **G2** |  |  | Requirements Elicitation | Its is concerned with requirements come from and how they can be collected by requirements engineer. |  |  |
|  | 1 |  | Requeriments Sources defined and analyzed | It is designed to promote awareness of different requirement sources and frameworks for managing them. | RE | RE, Stakeholders |
|  | 2 |  | Elicitation Techniques choosen and aplyed | Examples: Interviews, Scenarios, Prototypes, Facilitated Meetings, Observation | RE | RE, Stakeholders |
|  |  | 1 | Extracting Requirements |  | RE | RE, Stakeholders |
|  |  | 2 | Capturing Requirements |  | RE | RE |
| **G3** |  |  | Requirements Analysis | Its concerned with the process of analyzing requirements to: detect and resolve conflicts between requirements, discover the bounds of the system and how it must interact with its environment; and elaborate system requirements to software requirements |  |  |
|  | 1 |  | Requirements Classification |  | RE | RE |
|  | 2 |  | Conceptual Modeling realized | Conceptual models comprise models of entities from the problem domain configured to reflect their real-world relationships and dependencies. Examples: data and control flows, state models, event traces, user interactions, objetc models and many others | RE | RE |
|  | 3 |  | Architectural Design and Requirements Allocation |  | RE | RE |
|  |  | 1 | Architectural Design done | In many cases, the requirements engineer acts as system architect because the process of analizyng and elaborating the requirements demands that susbystems and components that will be responsible for satisfying the requirements be identified. | RE | RE |
|  |  | 2 | Requirements Allocation done | The requirements allocation is the assignment of responsability for satisfying requirements to subsystems. | RE | RE |
|  | 4 |  | Requirements Negotiation | Its is concerned with resolving problems with requirements where conflicts occur; between two stakeholders´requering mutually incompatible features, or between requirements and resources or between capabilities and constraints. However, a strong case can also be made for counting it as part of requirements validation | RE | RE, Stakeholders |

| | | | | | |
|---|---|---|---|---|---|
| **G4** | | Requirements Specification | It is concerned with the structure, quality and verifiability of the requirements document. | | |
| 1 | | The System Requirements Definition Document done | This document (sometimes known as the user requirements document or concept of operations) records the system requirements. | RE | RE, users/customers |
| 2 | | The Software Requirements Specification (SRS) done | | RE | RE |
| 3 | | Document Structure and Standards used | Several recommended guides and standards exist to help define the structure of requirements documentation. These include IEEE P1233/D3 guide, IEEE Std.1233 guide, IEEE Std. 830-1998, ISO/IEC 12119-1994, IEEE Std. 1362-1998 concept of operations (ConOps) | RE | RE |
| 4 | | Document Quality | | RE | RE, QA |
| **G5** | | Requirements Validation | The aim is to pick up any problems before resources are commited to addresing the requirements. | | |
| 1 | | The conduct of Requirements Reviews done | Perhaps themost common means of validation is by inspection or formal reviews of the requirements documen(s). A group of reviewers is constituted with a brief to look for errors, mistaken assumptions, lack of clarity and deviation from standards practice. | RE | RE, users/customers |
| 2 | | Prototyping done | it is commonly employed for validating the requirements engineer´s interpretation of the system requirements, as well as for eliciting new requirements. | RE | RE, users/customers |
| 3 | | Model Validation done | The quality of models developed during analysis should be validated | RE | RE |
| 4 | | Acceptance Tests done | It is an essential property of a system requirements, it should be possible to validate that the finished product satisfies the requirement. | RE | RE, users/customers,<br>Tester |
| **G6** | | Requirements Management | It is an activity than spans the whole software life cycle. It is fundamentally about change management and the maintenance of the requirements in a state that accurately mirrors the software to be, or that has been built | | |
| 1 | | Change Management controlled | It is central to management of requirements | | |
| | 1 | The procedures that need to be in place and the analysis that should be applied to propose changes defined | | RE | RE |

| | | | | | |
|---|---|---|---|---|---|
| | 2 | Requirements Attributes defined | Requirements should consist not only of a specification of what is required, but also of ancillary information that helps manage and interpret the requirements. The most fundamental requirements attribute, however, is an identifier that allows the requirements to be uniquely and unambiguosly identified | RE | RE, SCM |
| | 3 | Requirements Tracing controlled | It is fundamental to performig impact analysis when requirements change. A requirement should be traceable backwards to the requirements and stakeholders that motivated it.Conversely, a requirement should be traceable forwards into requirementsand design entities that satisfy it | RE | RE, SCM |

Table 3.4 Resulting Goals from Requirements Engineering after Goals Analysis Activity

# Software Requirements Process

| Goals | Description | Mapping with tools | Category of INCOSE Taxonomy | Description |
|---|---|---|---|---|
| G1 Process of Software Requirements defined | Showing how requirements engineering dovetails with the overall software engineering process | | | |
| 1 Process Models defined | How the activities of elicitation, analysis, specification, validation and management are configured for different types of projects and constraints | SI | | |
| 1 1 Description of how Requirements Engineering activity will be done and refined through the life cycle of the software process | | | | |
| 1 2 To define how to identify requirements as configurations items,and manage under the same configuration regime as other products of the development process | | | | |
| 1 3 suitable process to the organization and project context tailored | | | | |
| 2 Process Actors defined | Introduces the roles of the people who participate in the requirements engineering process | SI | | |
| 2 1 Users, Customers and Stakeholders (intern and extern) defined | | | | |
| 3 Process Support and Management | Introduces the project management resources required and consumed by requirements engineering process. It make the link from process activities identified in (G1.1) to issues of cost, human resources, training and tools | SI | | |
| 3 1 Expected project management resources required and consumed by requirements engineering process defined | | | | |
| 3 2 Issues of cost, human resources, training and tools linked to activities of process model. | | | | |
| 4 Process Quality and Improvement | Its purpose is to emphasize the key role requirements engineering plays in terms of cost, timeliness and customer satisfaction of software products. | SI | | |
| 4 1 Requirements Engineering covered by process improvement standards and models | | | | |
| 4 2 Requirements Engineering measures and benchmarking stablished | | | | |
| 4 3 Planing and Implementation improvemented | | | | |

|  |  |  |  |  |
|---|---|---|---|---|
| **G2** Requirements Elicitation | Its is concerned with requirements come from and how they can be collected by requirements engineer. | SI | Requirements Engineering/Requirements Management/Requirements Capture & Identification | Requirements Capture Tools accept text information from heritage sources, users, customer requirements and customer operations concepts. They assemble the information. They assist the engineer in finding relationships among entities in the information and in moving among the entities. Modern forms of these tools use natural language processing to interpret the text and hypertext technology to automatically build electronic hypertext versions of the text. These tools also accept requirements change documents and, assist in establishing where these changes impact earlier documents. |
| 1 Requeriments Sources defined and analyzed | It is designed to promote awareness of different requirement sources and frameworks for managing them. |  |  |  |
| 2 Elicitation Techniques choosen and aplyed | Examples: Interviews, Scenarios, Prototypes, Facilitated Meetings, Observation |  |  |  |
| 1 Extracting Requirements |  | SI | Requirements Engineering/Requirements Management/Requirements Capture & Identification/Tools for Elicitation of Requirements | Requirement Identification Tools aid the engineer in separating requirements in the information before him from extraneous information. Modern versions of these tools use natural language processing to identify statements containing imperatives of any kind in the information. |
| 1 Interviews |  | SI |  |  |
| 2 Scenarios |  | SI |  | There is a link to conceptual modleing because recent modeling notations have attempted to integrate scenario notations with object-oriented analysis techniques |
| 3 Prototypes |  | SI | System Design/HMI Prototyping Tools | There is a stronger overlap with the use of prototypes for requirements validation. Human Machine Interface (HMI) Prototyping techniques are used to develop or verify HMI design by modeling the interaction of operators or users with the system. This work is particularly useful when performed early and reviewed with users to verify requirements and assumptions about the system. Automation provides the capability to describe input/output screens, to depict parts of the system and to animate these presentations. |
| 4 Facilitated Meetings |  |  |  |  |
| 5 Observation |  |  |  |  |
| 2 Capturing Requirements |  | SI | Requirements Engineering/Requirements Management/Requirements Capture & Identification/Textual Requirements Capture Tools | Requirements Engineering/Requirements Generation Tools. Requirements generation tools utilize system simulation results, performance allocations, mission scenarios, and design constraints to generate lower level requirements in an organized and traceable manner. |

| | | | | |
|---|---|---|---|---|
| G3 Requirements Analysis | Its concerned with the process of analyzing requirements to: detect and resolve conflicts between requirements, discover the bounds of the system and how it must interact with its environment; and elaborate system requirements to software requirements | | | |
| 1 Requirements Classification | | SI | Requirements Engineering/Requirements Management/Requirements Classification | Requirement Classification Tools help the engineer classify the requirements based on work to be done so that the requirement analysis activity can be scheduled and tracked They help the engineer classify based on how the requirements will be used in modeling so that completeness of traceability can be monitored. |
| 2 Conceptual Modeling realized | Conceptual models comprise models of entities from the problem domain configured to reflect their real-world relationships and dependencies. Examples: data and control flows, state models, event traces, user interactions, objetc models and many others | SI | System Design/System Modeling/Static Behavioral | Static Behavior Modeling presents a static description of the behavior of the system. Static implies that the model may represent dynamic behavior, but it is not executable Examples of static behavioral models are Functional Flow Block Diagrams, State Transition Diagrams, and Object Interaction Diagrams |
| 3 Architectural Design and Requirements Allocation | | | | |
| 1 Architectural Design done | In many cases, the requirements engineer acts as system architect because the process of analizyng and elaborating the requirements demands that susbystems and components that will be responsible for satisfying the requirements be identified. | SI | System Design/System Modeling/Static Behavioral | Static Behavior Modeling presents a static description of the behavior of the system. Static implies that the model may represent dynamic behavior, but it is not executable. Examples of static behavioral models are Functional Flow Block Diagrams, State Transition Diagrams, and Object Interaction Diagrams |
| 2 Requirements Allocation done | The requirements allocation is the assignment of responsability for satisfying requirements to subsystems. | SI | Requirements Engineering/Requirements Generation | Requirements generation tools utilize system simulation results, performance allocations, mission scenarios, and design constraints to generate lower level requirements in an organized and traceable manner. |
| 4 Requirements Negotiation | Its is concerned with resolving problems with requirements where conflicts occur; between two stakeholders´requering mutually incompatible features, or between requirements and resources or between capabilities and constraints. However, a strong case can also be made for counting it as part of requirements validation | | | |
| G4 Requirements Specification | It is concerned with the structure, quality and verifiability of the requirements document. | | | |
| 1 The System Requirements Definition Document done | This document (sometimes known as the user requirements document or concept of operations) records the system requirements. | SI | | |
| 2 The Software Requirements Specification (SRS) done | | SI | | |
| 3 Document Structure and Standards used | Several recommended guides and standards exist to help define the structure of requirements documentation. These include IEEE P1233/D3 guide, IEEE Std.1233 guide, IEEE Std. 830-1998, ISO/IEC 12119-1994, IEEE Std. 1362-1998 concept of operations (ConOps) | SI | | |
| 4 Document Quality | | SI | | |

|  |  |  |  |  |
|---|---|---|---|---|
| **G5** Requirements Validation | The aim is to pick up any problems before resources are commited to addresing the requirements. |  |  |  |
| 1 The conduct of Requirements Reviews done | Perhaps themost common means of validation is by inspection or formal reviews of the requirements documen(s). A group of reviewers is constituted with a brief to look for errors, mistaken assumptions, lack of clarity and deviation from standards practice. | SI |  |  |
| 2 Prototyping done | it is commonly employed for validating the requirements engineer´s interpretation of the system requirements, as well as for eliciting new requirements. | SI |  |  |
| 3 Model Validation done | The quality of models developed during analysis should be validated |  |  |  |
| 4 Acceptance Tests done | It is an essential property of a system requirements, it should be possible to validate that the finished product satisfies the requirement. |  |  |  |
| **G6** Requirements Management | It is an activity than spans the whole software life cycle. It is fundamentally about change management and the maintenance of the requirements in a state that accurately mirrors the software to be, or that has been built |  |  |  |
| 1 Change Management controlled | It is central to management of requirements | SI | Requirements Engineering/Requirements Management |  |
| 1 The procedures that need to be in place and the analysis that should be applied to propose changes defined |  |  |  |  |
| 2 Requirements Attributes defined | Requirements should consist not only of a specification of what is required, but also of ancillary information that helps manage and interpret the requirements. The most fundamental requirements attribute, however, is an identifier that allows the requirements to be uniquely and unambiguosly identified | SI | Requirements Engineering/Requirements Management/Requirements Traceability | Requirement Traceability Tools enable the engineer to link requirements to their source, to changes in requirements and to modeling elements that satisfy the requirements. They provide traceability among the successive documents that are used to review the system development. |
| 3 Requirements Tracing controlled | It is fundamental to performig impact analysis when requirements change. A requirement should be traceable backwards to the requirements and stakeholders that motivated it.Conversely, a requirement should be traceable forwards into requirementsand design entities that satisfy it | SI |  |  |

Table 3.5 Matching of Goals from different information sources.

### Specifying Goal Dependencies

Dependency relations exist between pairs of goals. A goal dependency implies that a given goal is contingent upon another goal for completion, relying on or requiring the aid for another goal or agent for support. The objective of this activity is to develop an understanding of these relationships among the goals. We represent many dependencies by means of i* diagrams, but their not have any mechanism for denote precedende, for this reason was useful to use the GBRAM notation (it was used for construction of tables 3.4 and 3.5 -the precedence is noted in the tables by means hierarchical accommodation of columns-).

In GBRAM, goals are organized according their precedence relations, simplifying the determination of a goal´s preconditions and postconditions. The only type of dependency necessary for ordering the goals is the precedence dependency. However, two other types of dependencies are recognized: agent dependency and contract dependency; both are forms of precedence dependencies.

A precedence dependency between goals G1 y G2, where goal G1 must be completed before goal G2, is expressed as G1<G2. Organizing goals according to their precedence relations enables us to envisage operationalizations of these goals for consideration of possible elaborations and refinements.

Precedence relations are identified for each goal by asking "What goals are prerequisites for this goal?", "Do any goal depend on the availability of this information for achievement?" and "What goals must follow this goal" The answers to these questions facilitate the organization of goals with the prerequisite goals listed prior to a given goal.

A contract dependency between goals G1 y G2, where goal G2 must be achieved if goal G1 occurs, is expressed G1→G2; thus a contract dependency differs from a precedence dependency by a trigger. For example, G1 happens hence G2 must complete, as opposed to G2 requiring G1 to complete to enable G2 to complete.

Example: Consider goal G1 in Table 3.6. By asking *"What goals must follow this goal?"* and *"Do any goals depend on the availability of this information for achievement?"* it is possible determine that G1 must occur before G2 and G3 can be achieved (because RE should take into account the requirements sources analysis for choosing the elicitation techniques). Thus, G1<G2<G3 (i.e. G2 cannot be achieved before G1 is completed)

| Goals | Agent | Stakeholders |
|---|---|---|
| G1: Requirements Sources Defined | RE | RE, Stakeholders |
| G2: Requeriments Sources Analized | RE | RE, Stakeholders |
| G3: Elicitation Techniques Choosen | RE | RE, Stakeholders |

Table 3.6 Precedence Dependency Example

Once the goal relationships have been identified and the dependencies specified, the goal hierarchy may be constructed (that is showed in tables 3.4 and 3.5)

**Constructing a Goal Hierarchy**

Goals offer a rich outlining structure for organizing requirements information, addressing the need for documents that are easy to index and read. Since goals provide an organizing structure, GBRAM refers to this outlining mechanism as a goal topography.

A goal topography can be represented in a number of ways. We use the outline form, because when a goal topography is manually constructed is tipically constructed in this form. (Tables 3.4 and 3.5 show the outline form of the framework of Requirements Engineering).

The advantage of topographies expressed in the form of an outline is that they provide a clear mapping to the goals from different sources.

By organizing the goals according to the goal topography, changes in the goals can be managed. This is important because the goal topography enables us to localize the goals which are affected by a change in related or proximated goals. Since dependency relations are tracked, traceability is possible the narrowing of scope facilitates the identification of other goals that are affected as a result of changes in a specific goal.

Once the goal topography is constructed, we must systematically elaborate and refine the goal set.

## 3.2. Goal Refinement Activities

This section discusses the activities we must perform during goal refinement. In our GBRAM adaptation goal refinement concerns three specific activities which may be summarized as follows:

- *Refinement* of the goal set to prune the size of the goal set;
- *Elaboration* of the goals to uncover hidden goals and requirements; and
- *Operationalization* of the goals into characterization attributes

The remainder of this section provides a discussion of each of these activities within the context of the associated activities shown in Figure 2.1. Examples of the method´s applicability, taken from the our case study, are provided throughout the remainder of this section. It should be noted that these activities need not be performed sequentially; rather they may be performed concurrently with occasional interleaving and iteration as evidenced by the arrows in Figure 2.1. Another important aspect that we want to remark is that this phase is too a refinement phase of the i* diagrams as goals.

### 3.2.1. Elaboration of Goals

In GBRAM goals are elaborated by considering scenarios[1] and goal obstacles[2]. Goal obstacles are identified in order to consider the possible means of goals failure. They are elaborated further by identifying scenarios to develop an understanding of how the goals can be operationalized. Finally, goal constraints [3] are identified to expand our understanding of what obligations must be met for goal completion.

#### 3.2.1.1. Specifying Goal Obstacles

The objective of specifying goal obstacles for each goal is to capture any information pertaining to the goals and system objectives that might otherwise be overlooked.

Original GBRAM is too meticulous in this sense (because of its original purpose), this step depend on our rationale because we must identify and construct goal obstacles by inquiry from the available information sources.

Goal obstacles are identified by analyzing statements that illustrate an example of a goal being blocked by another goal or condition which prevent its completion.

Obstacles can also be identified by asking “What other goal or condition does this goal depend on?”, “Can the agent responsible for a goal fail to achieve the goal?”, “If this goal is blocked, what are the consequences?”, “Can the failure of another goal to be completed cause this goal to be blocked?”.

Goal obstacles may be identified in parallel to goal identification.

| Goals | Goal Obstacles |
|---|---|
| G1: Requirements Sources Identified | 1. Requirements Sources not Identified (G) |
| G2: Requeriments Sources Analized | 1. Requirements Sources not Analized (G)<br>2. Requirements Sources not Identified (P) |

Table 3.7 Prerequisite Failure Obstacle Example

For each goal, the normal first case goal obstacle is specified by simply negating the verb in the goal name (Table 3.7). These are considered general *goal failure* (G) obstacles

[1] Scenarios are behavioral descriptions of a system and its environment arising from restricted situations.

[2] Goal obstacles prevent or block the achievement of a given goal.

[3] A goal constraint places a condition on the completion of a goal

because they denote basic goal failure (expressed ¬Gi). Each general failure obstacle must then be analyzed to consider other possible obstacles (using scenarios if it is necessary). When a goal having a precedence relation is obstructed because the precedence goal fails, it is considered a *prerequisite failure* (P) obstacle. When a goal fails because the responsible agent is irresponsible, it is considered an *agent failure* (A) obstacle. In this case, the irresponsible party must be tracked down and held accountable. When a goal fails because the goal that it holds a contract relation with fails, it is considered a *contract failure* (C) obstacle.

Goal obstacle analysis facilitate the identification of exception cases guiding the identification of new, additional goals.

Once the goal obstacles are specified, we must consider the possible scenarios that are likely for each obstacle.

#### 3.2.1.2. Constructing Scenarios

The ways in which goals can fail are identified during goal obstacle analysis. The objective of this activity is to elaborate this information further via scenario analysis. Scenarios[4] offer a natural way to describe special, exceptional circumstances.

Scenarios are the most creative artifact of the analysis process and play a major role in discovering goals. Scenarios denote concrete circumstances under which a goal may fail, helping us uncover hidden goals or issues needing further resolution that might otherwise go unnoticed or be overlooked, thereby supporting the process of refining goals.

When goal priorities change, scenarios facilitate the evaluation of these new priorities.

Scenarios are identified by considering the goals and goal obstacles previously identified to determine the reasons why and the circumstances under which a goal may be completed or can fail. By asking "Why?" and "What happens if this goal isn´t achieved?" scenarios can be identified that address why a goal failed or what the consequences are if the goal should fail. Initially, the normal first case obstacles are considered and possible scenarios are defined for each one. This is done for each obstacle by asking "What are the circumstances under which this obstacle can occur?" "Why did this obstacle occur?" and "Why was this goal not achieved?" Answers to these questions facilitate the identification of scenarios.

It is beneficial to have domain experts construct scenarios, given that they are likely to think of more possible ways in which goals can fail than is an analyst who may be unfamiliar with the domain

[4] Scenarios are behavioral descriptions of a system and its environment arising from restricted situations.

Scenario identification and construction provides a systematic way to find abnormal cases. Scenarios also facilitate the consideration of assumptions and issues pertaining to the goals themselves.

#### 3.2.1.3. Identifying Constraints

The objective of this step is to identify any constraints that exist for goal completion. Constraints provide information regarding circumstances that must exist or conditions that must be met for a given goal to be completed. Constraints are identified by analyzing each textual description fragment.

As a general rule, constraints may be identified by looking for dependency relations and by searching for temporal connectives, such as *during*, *before* an *after*, or variants thereof.

In our approach we take into account these activities of elaboration of goals, however is not necessary to be so exhaustive (like original GBRAM); in fact this step depend on our rationale because we must identify and construct goal obstacles by inquiry from the available information sources when we think is pertinent or the context is not clear.

Once the goals have been specified and elaborated to the greatest possible, this information must be operationalized and translated into the natural language expression for characterization attributes.

### 3.2.2. Operationalization of Goals Into Characterization Attributes

During operationalization, the actions described stakeholders and the extracted information are related back to the goals.

Goals are a logical organizing mechanism for the incorporation of information.

As mentioned above the actions are mapped into goals. We used tables to represent this mapping (see Table 3.5).

In the operationalization activities we also used tables as a resulting artifact. While no formal in the strict sense, provides a textual representation of goals and attributes organized according to system goals as prescribed by the topography.

For constructing that table, we analize the previous information from Goal Analysis and Refinement activities (tables 3.4 and 3.5), then apply the inquiry cycle (explained in the next section) for getting the question-answer pairs attained to each characterization attribute.

Here, we present in table 3.8 the resulting table of the actvitivies of adapted GBRAM to our approach:

| ***Root Level*** | ***Question*** | ***Value*** | ***Cathegory*** | ***Question*** | ***Value*** | ***Cathegory*** | ***Question*** | ***Value*** | ***Cathegory*** |
|---|---|---|---|---|---|---|---|---|---|
| **1.Requirements Engineering Tools** | When do you apply? | | | | | | | | |
| | | Defining Process | **1.1 Defining Process** | Standard Tools or Methodology Specific? | | | | | |
| | | | | | Standard Desk Tools | **1.1.1 DeskTools** | | | |
| | | | | | | | | | |
| | | | | | Methodology Specific | **1.1.2 Rapid Application Development** | | | |
| | | | | | Estimation and Cost Tools | **1.1.3 Estimation and Cost Tools** | | | |
| | | Requirements Elicitation | **1.2 Requirements Elicitation** | Extracting, Generation or Capturing Requirements? | | | | | |
| | | | | | Capturing | **1.2.1 Capture Tools** | | | |
| | | | | | Extracting | **1.2.2 Extracting Tools** | | | |
| | | | | | | | Are you Using Interviews? | Yes | **1.2.2.1 Interviews Tools** |
| | | | | | | | Are you Using Scenarios? | Yes | **1.2.2.2.Scenario Tools** |
| | | | | | | | Are you Using Prototypes? | Yes | **1.2.2.3 Prototypes Tools** |
| | | | | | | | Facilitated Meetings? | Yes | **1.2.2.4 Facilitates Meetings Tools** |
| | | | | | | | Observation? | Yes | **1.2.2.5 Observation Tools** |
| | | | | | | | Other Techniques? | Yes | **1.2.2.6 Other Extraction Techniques** |
| | | | | | Generation | **1.2.3 Generation Requirements Tools** | | | |
| | | Requirements Analysis | **1.3 Requirements Analysis** | Classification or Negotiation? | | | | | |
| | | | | | Classification | **1.3.1 Requirements Classification Tools** | | | |
| | | | | | Negotiation | **1.3.2 Requirements Negotiation Tools** | | | |
| | | Requirements Specification | **1.4 Requirements Specification** | | | | | | |
| | | | | Do you need to model? | Conceptual Model | **1.4.1 Modeling Tools** | | | |
| | | | | Do you want Generate a Document? | Automatic Generation of Requirements Document | **1.4.2 Generation of Requirements Document Tools** | | | |
| | | Requirements Validation | **1.5 Requirements Validation** | | | | | | |
| | | | | Automatic Reviews of Requirements? (Detecting inconsistencies?) | Automatic Reviews | **1.5.1 Reviews Tools** | | | |
| | | | | Generating Test Acceptance? | Automatic Test Acceptance | **1.5.2 Acceptance Test Tools** | | | |
| | | Requirements Management | **1.6 Requirements Management** | Do you need mantain a Change Management of Requirements? | Change Management Requirements | **1.6.1 Change Management Requirements Tools** | | | |
| | | | | | | | Explicit Traceability? | Traceability | **1.6.1.1. Requirements Traceability Tools** |

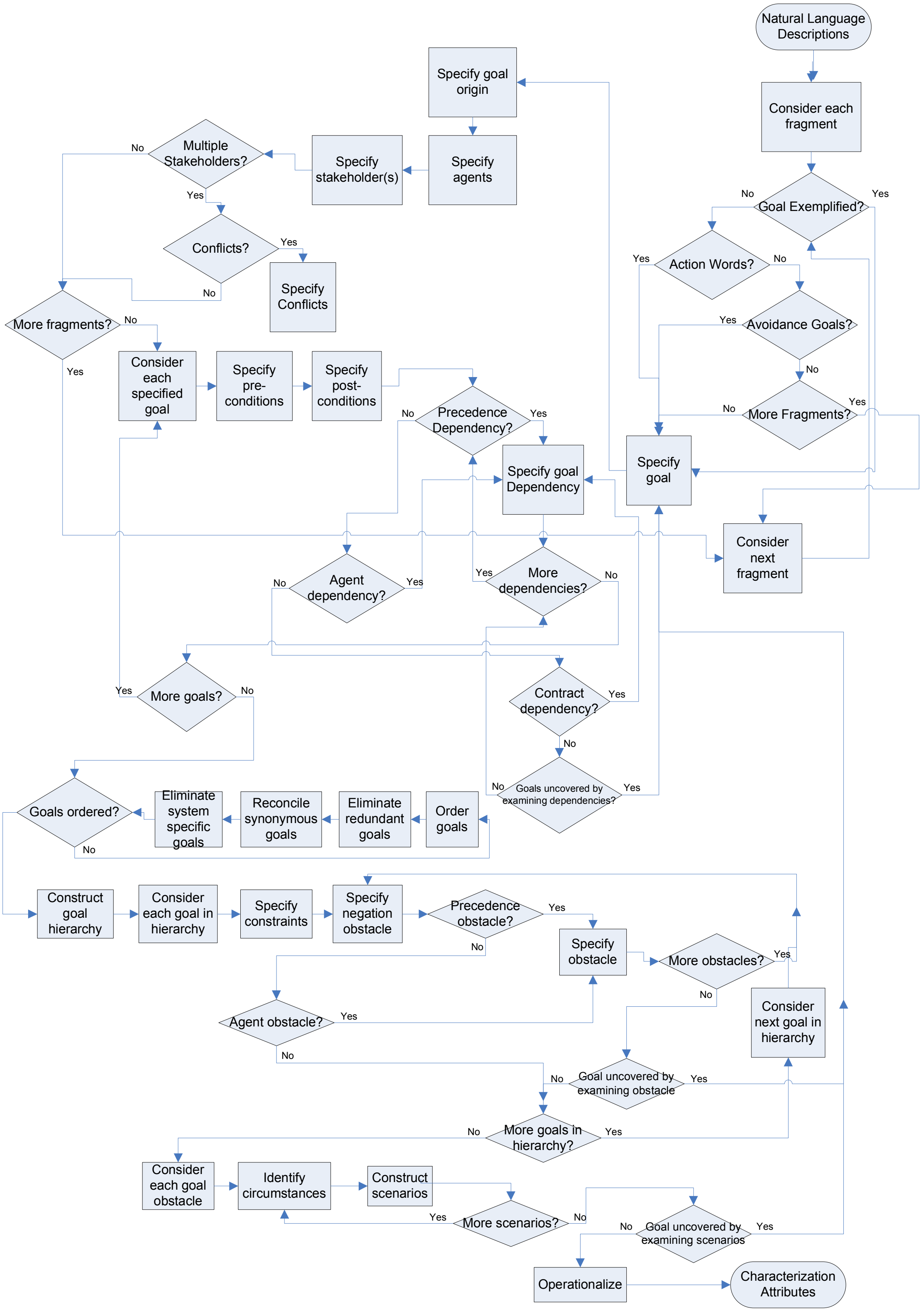


Figure 3.4 Adapted GBRAM Control Flor Chart

Figure 3.4 provides a reduced overview of the adapted activities of GBRAM, represented in a control flow chart. The flow chart is an approximation of the method. The boxes in the figure represent the activities that we must perform. The diamonds represent the various inquiry points for an inquiry-driven approach throughout the adapted GBRAM process, while the ellipses represent the input (natural language descriptions) and output (characterization attributes) of the method. The arrows denote the flow throughout the process as well as the iterative nature of goal-based analysis. Boxes and ellipses have only one output edge; however, since the diamonds represent inquiry points, a decision is required of us. Thus, diamonds have two outgoing edges; one edge represents an answer of 'yes', and the other edge indicates an answer of 'no'.

Since many of the activities may be performed concurrently, the flow chart simply serve to demonstrate how GBRAM is performed.

## 3.3. Summary

This section presented the Goal-Based process and its associated heuristics within the context of our analysis of Software Requirements Area.

First the process of identifying high-level goals from the sources by the application of GBRAM is fundamental.

The extracted goals are expressed in structured natural language. We begin the goal mining process exploring any available information sources such as :

- The information related with the domain and standards (i.e. IEEE, EIA) about Application Development in Software Engineering [14], [15], [16], [18], [19], [20], [23], [24] (see appendix A and B).
- The diverse taxonomies that many consultant and organizations provide in that field [1],[2],[3],[12] (see appendix B).
- The availability of products in the domain (e.g. vendors like Rational). [13],[21],[22].

Once goals are identified, they are elaborated; goal elaboration entails analyzing each goal for the purpose of documenting goal obstacles, scenarios, constraints, pre-conditions, post-conditions, questions and rationale. Goal refinement consists of removing synonymous and redundant goals, resolving any inconsistencies that exist within the goal set.

In practice, the goals are documented and annotated with auxiliary information including the stakeholders and responsible agents (see Table 3.4). Goals are then organized according to precedence relations, the next step involves the matching of goals of the main information sources (see Table 3.5).

Detailed techniques and heuristics for each of these operations are described in the thesis [11]

Last, in the activity of operationalization the goals into characterization attributes, we construct a table analizing the previous information from Goal Analysis and Refinement activities (tables 3.4 and 3.5), then apply the inquiry cycle for getting the question-answer pairs attained to each characterization attribute.

# 4. Heuristics and Guidelines

This section provides a brief overview of some of the most useful heuristics of GBRAM and how we can apply them to our approach.

In the GBRAM, heuristics are rules which guide us towards a high probability of success while avoiding wasted efforts. GBRAM provides sets of heuristics and guidelines for the identification of goals. One of our future objectives is to propose heuristics for the specific domain that helps to adapt the taxonomy to the changes, evolution of goals and needs of the users.

The selection of heuristics depends upon the domain and on the information available.

There are four types of heuristics employed using GBRAM:

- Identification heuristics
- Classification heuristics
- Refinement heuristics
- Elaboration heuristics

Identification heuristics assist us in identifying goals, stakeholders, agents, and constraints from multiple sources. The objective of goal classification heuristics is to aid us in determining the type of each goal identified. Refinement heuristics employ a series of question and techniques to reduce the size of the goal set. Elaboration heuristics address the need to acquire more detailed information by considering goal dependency relations, suggesting the goal obstacles for which scenarios should be constructed and which scenarios to elaborate. We have to mention the concurrent and overlapping nature of these activities.

## 4.1. Goal-Based Instantiation of Inquiry Cycle

The GBRAM heuristics are a set of rules. Some of these heuristics are straightforward, not requiring that we employ any specific inquiry process for adapting it to our domain. In contrast, other heuristics are meaningless without the accompanying questions to guide us in developing a deeper understanding and uncovering hidden goals and requirements. Often a simple answer to a question is insufficient; for example, justification or rationale for a particular response may be needed to understand the system. Thus, GBRAM offers a set of recurring question types which follows the inquiry cycle approach instantiated for goal-based analysis, and guide us in applying an inquiry-driven approach as they comprehensively investigate the available artifacts for descriptive answers.

The Inquiry Cycle model is a formal structure for describing discussions about requirements. That model was useful in the construction of question-answer pairs attained to characterization attributes (showed before in table .

These recurring question types are summarized below:

- *What-is*: These question request specific information regarding terminology which is unclear to someone with no knowledge of the application domain. For example, we may ask a stakeholder, "What is the process for deciding what elicitation technique to use?". In any analysis effort, participants in the process have to develop a common understanding of the concepts and terms. What-is questions enable us to do precisely this. What-is question clarify a situation or scenario enabling us to ensure that something is understood correctly. This type of question is most useful when we interact directly with stakeholders.

- *Who-is*: These questions request specification of the agent responsible for the given task, process or goal. It is helpful to ask questions such as, "Who is responsible for the ultimate decision?" and "Who is responsible for this task?" By asking who-is questions, we can acquire information about the various agents and tie the information to the corresponding goal-based on implied or explicit responsibility.

- *Why*: These questions request reasons which underlie work activities. We must ask questions such as "Why is this information routed?" While it is relatively simple for us to determine what information is required to route, the reason why the information is routed can be very difficult (if not impossible) to ascertain without direct stakeholder inquiry.

- *What-if*: It is beneficial to ask questions that enable us to further examine cases in which an unexpected action occurred in order to explore how other system features may be affected. These questions mandate the consideration of the other agents and processes that would be affected in the event of such an unexpected cancellation.

- *When*: These questions request timing constraints for a given event or events. To ascertain this type of information, we ask questions that may be followed up with a clarification question.

- *Relationship*: These questions ask how one agent is related to another or how one goal is related to another goal so that the dependency relations may be established. For example, we may consider each goal and ask: "What goals are prerequisites for this goal?", "What goals must follow this goal?", and "What agents depends on this goal for completion of their responsibilities?"

These question types assist us in knowing when and how to apply GBRAM heuristics by providing a guide as to how much detail is needed one can be reasonably confident that the goals have been fully elaborated and that any hidden goals or requirements have been uncovered.

| Code | Definition |
|---|---|
| HIG | Heuristics for Identifying Goals |
| HIS | Heuristics for Identifying Stakeholders |
| HIA | Heuristics for Identifying Agents |
| HIC | Heuristics for Identifying Constraints |
| HRR | Heuristics for Refining Redundancies |
| HRS | Heuristics for Refining Synonymous |
| HRSS | Heuristics for Refining System-Specific |
| HED | Heuristics for Elaborating Dependencies |
| HEO | Heuristics for Elaborating Obstacles |
| HES | Heuristics for Elaborating Scenarios |

Table 4.1 Glossary of Heuristics Identifier Codes in GBRAM

Each heuristic set addresses a problem space. The remainder of this section shows some examples and guidelines for the appropriate time and manner of application.

Table 4.1 provides a glossary for the labels which serve as tags for the identification of each of the heuristic sets.

### 4.1.1. Heuristics for Goal Identification

The goal identification heuristics address the problem of how to extract goals from the documentation and resources available to us.

Goal identification is not limited to the initial activities from GBRAM; goals may be identified throughout the entire analysis process.

The general heuristics for identifying goals and objectives should be considered by us when exploring existing documentation. Those heuristics which are not explicit are accompanied by a series of questions to guide us.

Figure 4.1 illustrates a possible series of inquiry points for applying the goal identification heuristics to extract goals from available documentation.

One example of this heuristics set is:

**HG2. Abstraction mechanism may be employed to extract goals from available documentation by asking:**

- What goal(s) does this statement exemplify?

- What goal(s) does this statement block or obstruct?

**If the answer to either of these questions is yes, then express the statement as a goal which represents a state that is desired or achieved within the system.** (See Example 1 in section 3.1.2)

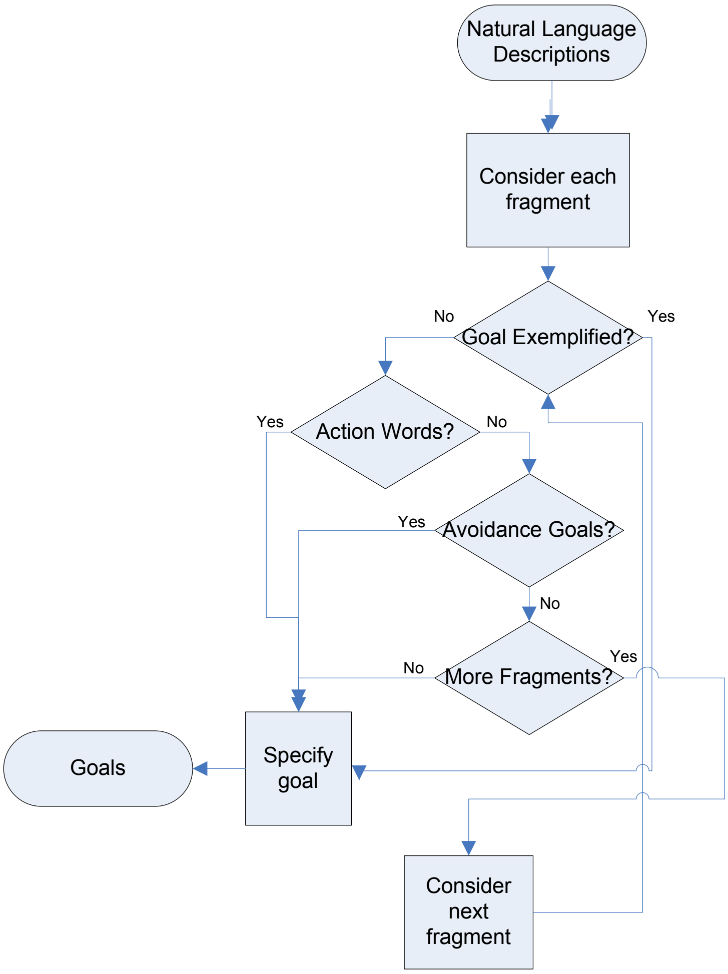


Figure 4.1 Control Flow Chart for Goal Identification

## 4.1.2. Heuristics for Stakeholder and Agent Identification

The heuristics for stakeholder identification address the problem of determining who and what parties claim an interest in the system.

We should consider the role that stakeholders play in prioritizing goals during negotiation.

**HIS3. Any representative affected by the completion or prevention of a goal is a stakeholder. Stakeholders are thus identified by asking:**

- Who or what claims a stake in this goal?
- Who or what stands to gain or lose by the completion or prevention of this goal?
- Who will use the system or component?

**HIA2. Responsible agents may be identified by considering each goal and asking:**

- Who or what agent is, could be, or should be responsible for this goal?

**The answer to this question will be the name of the responsible agent. The agent´s name should be 'attached' to the goal for which it is responsible.**

Table 3.3 illustrates how agents can be attached to goals using a tabular notation.

The example 1 of the section 3.1.3 illustrate these heuristics.

A summary of heuristics defined in GBRAM (only the suitable to our approach) is presented in Appendix D

# 5. Conclusions and Future Work

In this report we have proposed a methodological process (adapted GBRAM) to obtain the "characterization attributes" that are used to browse the taxonomy for classifying COTS components.

We have presented the description of the process of customizing GBRAM as a methodological approach to find "characterization attributes" to the case study of Software Requirements Area. That demonstrate this approach is workable.

The major contributions are:

a) It provides guidelines and heuristics for exploring, identifying, and organizing goals (potential characterization attributes) which guide us towards a high probability of success while avoiding wasted efforts.

b) It offers a guide for applying an inquiry-driven approach to goal-based analysis, that is useful for enhancing our questions-answers mechanism linked to mostly characterization attributes.

We recognize that the only valid test of a practical method is in its use on real projects. While the work reported here does not yet provide that level of validation (we think we have to apply it in other domains) it is developed in that way and does provide key insights in that sense.

The main adaptations of GBRAM to our approach are related in the corresponding section.

Our overall idea is to further develop, validate and adequate heuristics and strategies that can help the extraction of characterization attributes from goals in a specific domain. Appart from that we also want to develop a "goal-mining"[9] and a knowledge base of the domain. This allow not only an easy evolution of the taxonomy, but also the suitability and its dynamic use (i.e. the taxonomy should be constructed depending on the needs of the user). See figure 5.1

[9] Goal mining refers to the extraction of goals from the sources by the application of GBRAM.

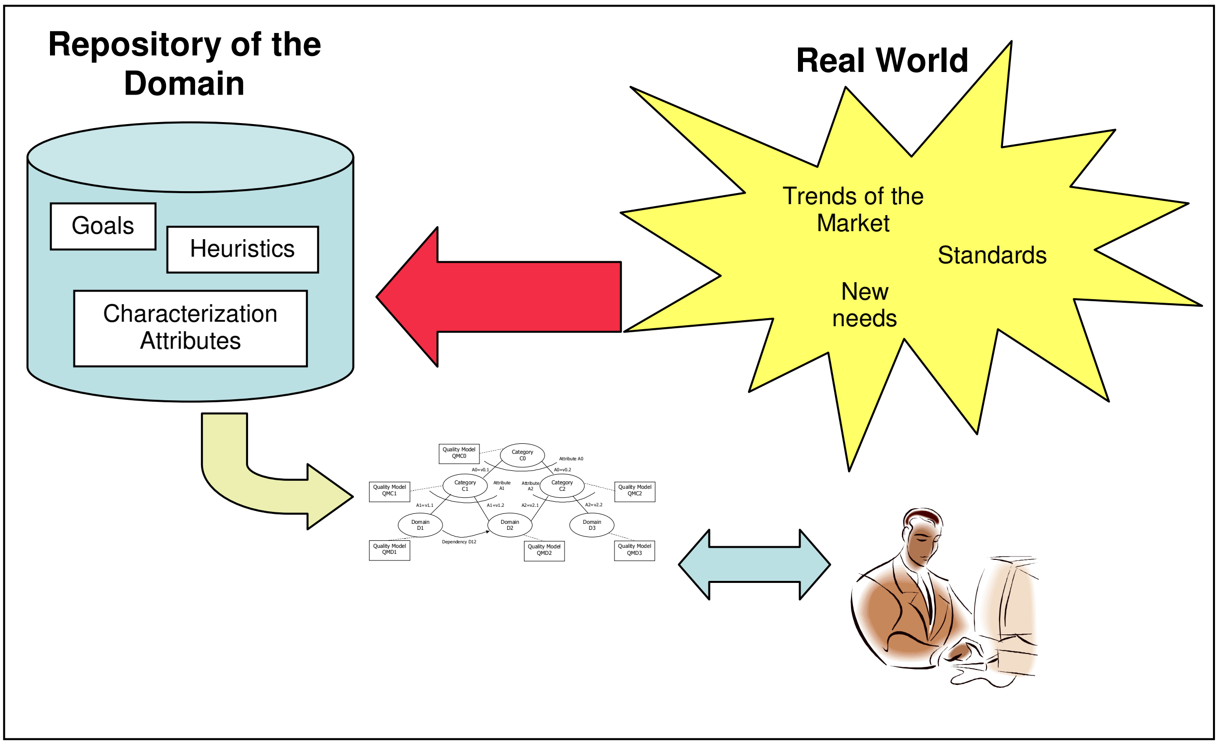


Figure 5.1 Overall motivation of the GBRAM use

Thus, there are a number of important aspects of research to investigate. The particular activities prospected for achieving our idea are the following:

- to improve the approach described in this report codifying the domain specific heuristics for applying our adapted GBRAM, this is, develop a set of reusable goals for each domain.

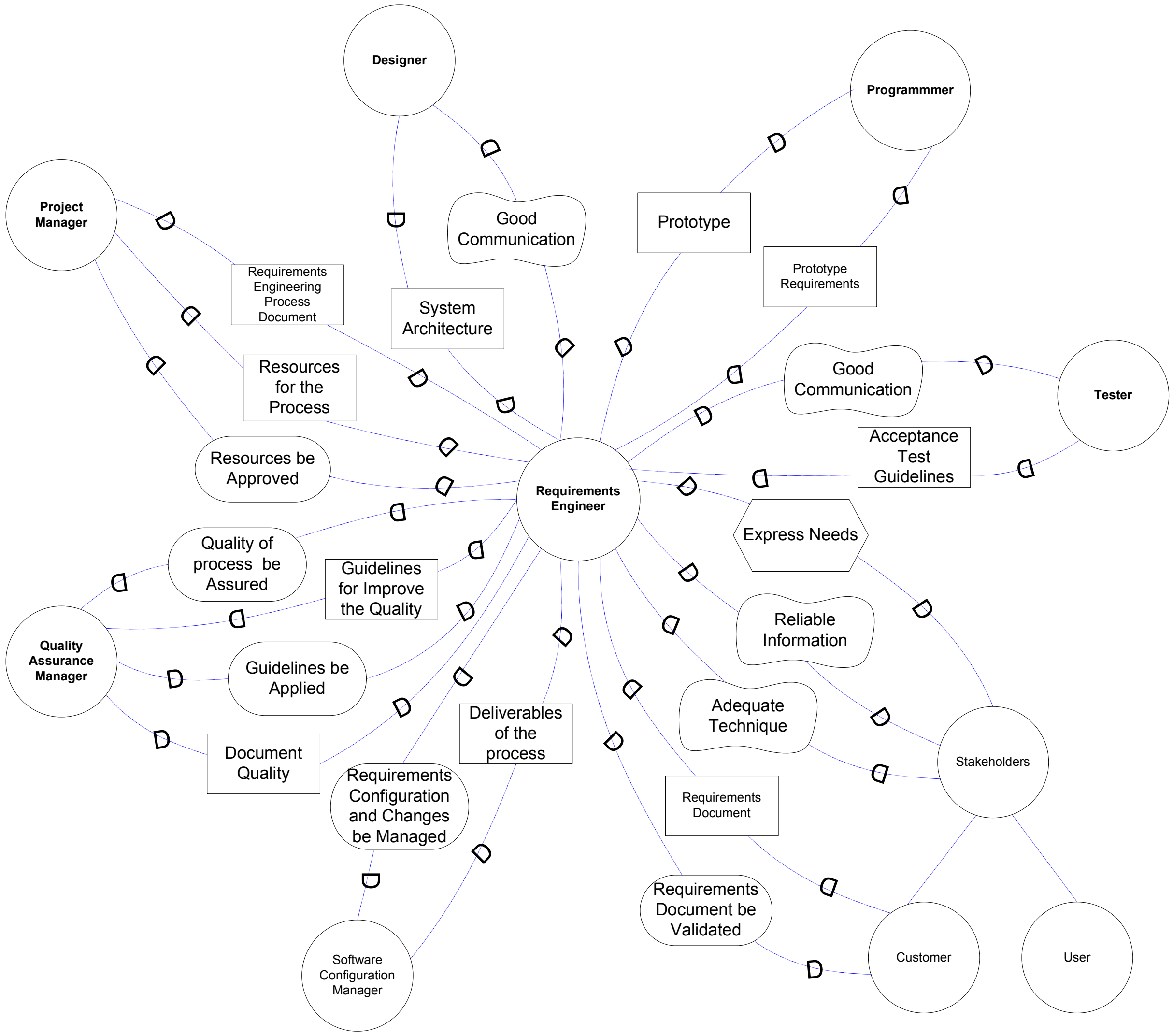
- to define a systematic mechanism to maintain and update the goal-mining of that specific domain

- to assure that such mechanism enhance in one hand the evolution of the taxonomy (it should be easy to include the new goals or reorganize the existent goals) and on the other hand the dynamic construction of the taxonomy depending on the specific needs of the user.

# Appendix 1

## I* Diagrams

Here we present the i* diagrams resulting of our case study. All them are based in SWEBOK information.



Designer
Programmmer
Project Manager
Good Communication
Prototype
Prototype Requirements
Requirements Engineering Process Document
System Architecture
Resources for the Process
Good Communication
Tester
Resources be Approved
Acceptance Test Guidelines
Requirements Engineer
Express Needs
Quality of process be Assured
Guidelines for Improve the Quality
Reliable Information
Quality Assurance Manager
Guidelines be Applied
Adequate Technique
Deliverables of the process
Stakeholders
Document Quality
Requirements Configuration and Changes be Managed
Requirements Document
Requirements Document be Validated
Software Configuration Manager
Customer
User
D

Figure A1. SD Model for Software Requirements Activity

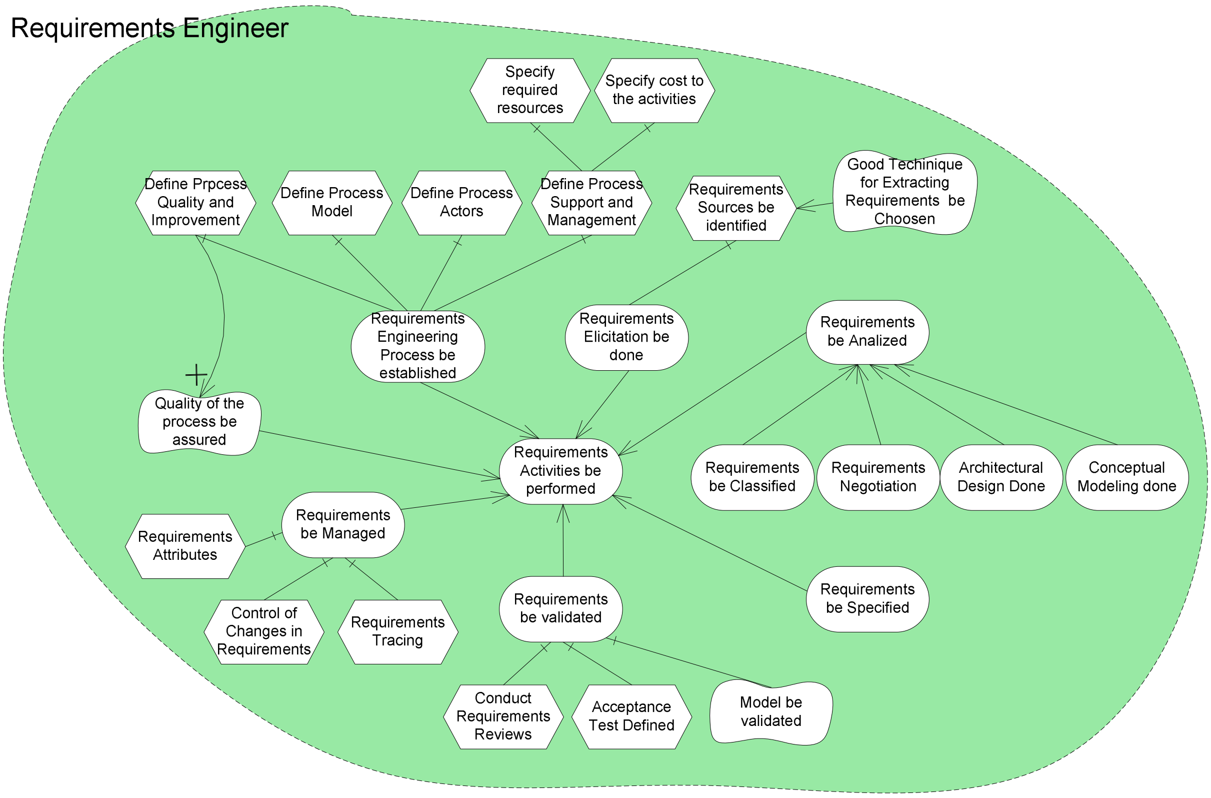

Requirements Engineer
Specify required resources
Specify cost to the activities
Define Prpcess Quality and Improvement
Define Process Model
Define Process Actors
Define Process Support and Management
Requirements Sources be identified
Good Techinique for Extracting Requirements be Choosen
Requirements Engineering Process be established
Requirements Elicitation be done
Requirements be Analized
Quality of the process be assured
Requirements Activities be performed
Requirements be Classified
Requirements Negotiation
Architectural Design Done
Conceptual Modeling done
Requirements Attributes
Requirements be Managed
Control of Changes in Requirements
Requirements Tracing
Requirements be validated
Requirements be Specified
Conduct Requirements Reviews
Acceptance Test Defined
Model be validated

Figure A2. SR Model for Requirements Engineer

# Appendix A

## Application Development Software Information

The remainder of this section present most of the information gathered in our context of interest. It is not the purpose of this section to analyse deeply either the software engineering or the software development process. Our purpose in this section is to provide the reader with a general overview of some of the main information we gather and key concepts involved in the software application development process to understand the domain and to exemplify the goals extraction. Issues such as software engineering process, methods, and specific activities related with it will only be addressed in an informative way, as parts of a bigger picture that will be later used as a framework reference.

Firstly, we present  the concept of Application in this context [14] :

- The set of business processes which capture and maintain business data and provide business information.  These processes may be either manual or automated.

- In information Technology, an application is the use of a technology, system or product.

- The term application is a shorter from application program.  An application program is a program designed to perform a specific function directly for the user or, in some cases, for another application program. Examples of applications include word processors, database programs, web browsers, development tools, drawing, paint, image editing programs, and communication programs. Applications use the services of the computer´s operating system and other supporting applications.  The formal request and means of communicating with other programs that an application program uses is called the application program interface (API).

In computer programs and software product development, the development environment is the set of processes and programming tools used to create the program or software product.  The term may sometimes also imply the physical environment.  An integrated development environment is one in which the processes and tools are coordinated to provide developers and orderly interface to and convenient view of the development process (or at least the processes of writing code, testing it, and packaging it for use).  An example of an IDE products is Microsoft Visual Studio .NET.  The term computer-assisted software environment (CASE) is generally used to describe a set of tools and practices that facilitate management of a software development project.

As a result, Software Application Development are all activities related in the construction of software programs, designed for performance a specific function required for the user(s), in some cases may be another program.

## Software Process [16]

The software process is a framework for the tasks that are required to build high quality software.

Software engineering is a layered technology (Fig. A1.1). Any engineering approach (including software engineering) must rest on an organizational commitment to quality.

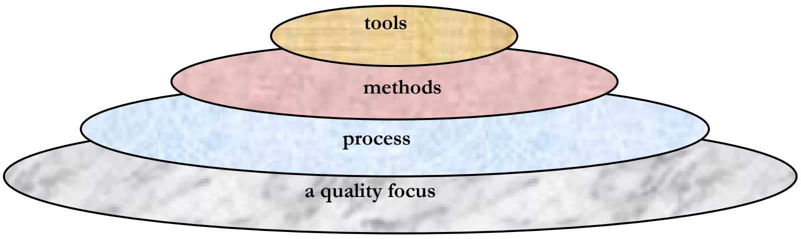


Figure A1.1 Software Engineering Layers

The foundation for software engineering is the process layer. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology. The key process areas form the basis for management control of software projects, and establish the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

Software engineering methods provide the technical "how to´s" for building software. Methods encompass a broad array of task that include: requirements analysis, design, program construction, testing and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

Software engineering tools provide automated or semi-automated support for the process and methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer aided software engineering (CASE), is established.

Thus, commonly we find that the software development process topic covers life cycles, best practices, and associated standards.

## Software Application Development

Our main source of information in the sense Standard Information related with Application Development Software is: SWEBOK (Guide to the Software Engineering Body of Knowledge)[15], that provide a consensuated characterization of the bounds of the Software Engineering discipline. The Body of Knowledge is subdivided into ten Knowledge Areas (KA). See Figure A1.2.

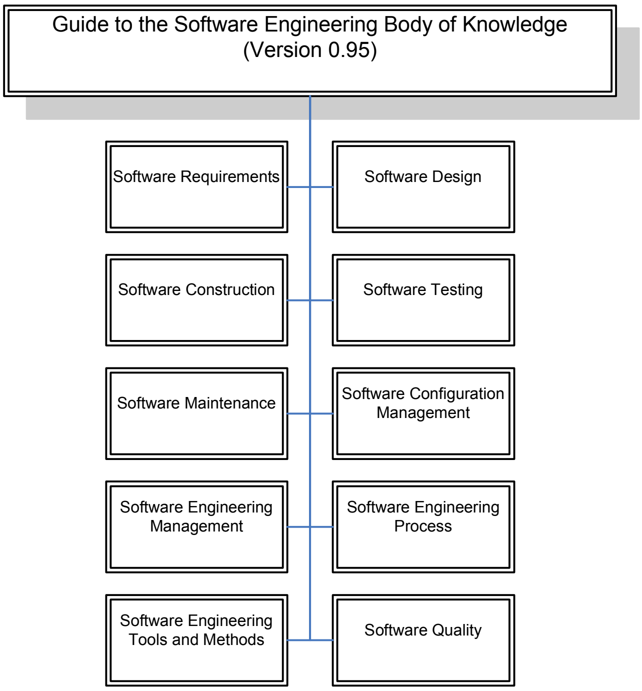

Guide to the Software Engineering Body of Knowledge (Version 0.95)
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Tools and Methods
Software Quality

Figure A1.2 SWEBOK Knowledge Areas

Our purpose in this section is to provide the reader with a general introduction and overview of some of the key concepts involved in the software application development process, specifically the Software Requirements area that is showed as a case study to exemplify the applicability of GBRAM in the construction of taxonomies. Issues such as software engineering process, methods, and specific activities related with it will only be addressed in an informative way, as parts of a bigger picture that will be later used as a framework reference, but all information gathered is in the references.

## **Software Requirements Area** [15]

The knowledge area breakdown that is chosen by SWEBOK (presented in Figure 3) is broadly compatible with the sections of ISO/IEC 12207-1995 [18] that refers to requirements engineering activities.

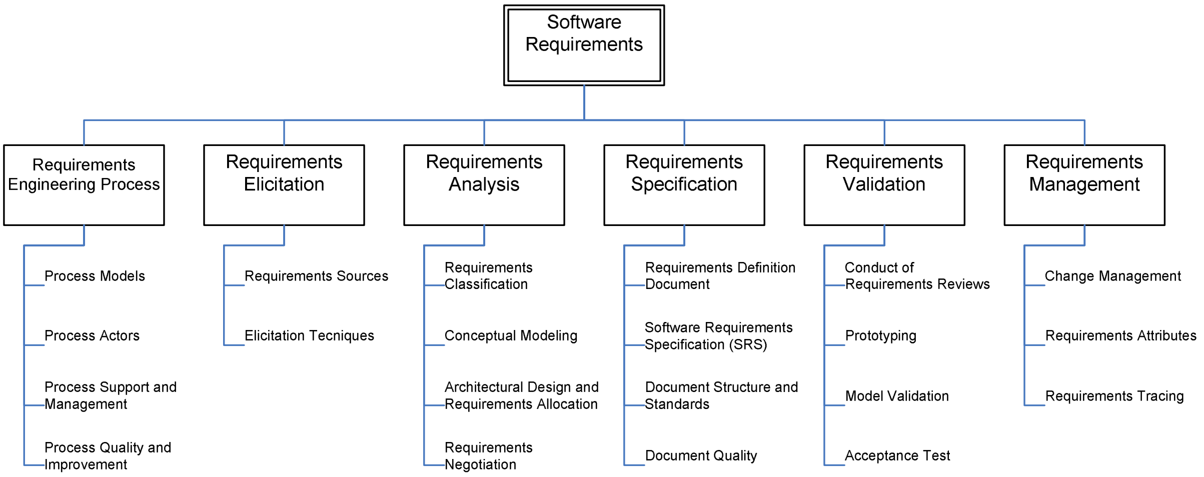


Figure 3. Software Requirements Area breakdown

A requirement is defined as a property that must be exhibited in order to solve some problem of the real world.

The first knowledge sub-area is the requirement engineering process, which introduces the requirements engineering process, orienting the remaining five topics and showing how requirements engineering is related with the overall software engineering process. It describes process models, process actors, process support and management and process quality improvement.

The second sub-area is requirements elicitation, which is concerned with where requirements come from and how they can be collected by the requirements engineer. It includes requirement sources and techniques for elicitation.

The third sub-area, requirements analysis, is concerned with the process of analyzing requirements to:

- detect and resolve conflicts between requirements;
- discover the bounds of the system and how it must interact with its environment;
- elaborate system requirements to software requirements.

Requirements analysis includes requirements classification, conceptual modeling, architectural design and requirements allocation and requirements negotiation.

The fourth sub-area is software requirements specification. It describes the structure, quality and verifiability of the requirements document. This may take the form of two documents, or two parts of the same document with different readership and purposes. The first document is the system requirements definition document, and the second is the software requirements specification. The sub-area also describes the document structure and standards and document quality.

The fifth sub-area is requirements validation whose aim is to pick up any problems before resources are committed to addressing the requirements. Requirements validation is concerned with the process of examining the requirements document to ensure that it defines the right system (i.e. the system that the user expects). It is subdivided into descriptions of the conduct of requirements reviews, prototyping, model validation and acceptance tests.

The last sub-area is requirements management, which is an activity that spans the whole software life -cycle. It is fundamentally about change management and the maintenance of the requirements in a state that accurately mirrors the software to be, or that has been, built. It includes change management, requirements attributes and requirements tracing.

Appendix B

# Trends and Terminology of the Actual Market

Here, we present many definitions and terminology used in the actual market of Software Application Development. The principal sources of information were [13], [17] and [19].

## Planning

The following is a list of planning-related terms and a description of each:

- **Methodware.** (Sometimes called software process management tools). This features knowledge "libraries," workflow, defined development routes, estimation support, multiple alternative processes, planning, tracking and related functionality.

- **Process management.** This is a management concept that describes the goal of increasing intra-enterprise coordination of separate business functions. The growth in the demand for enterprise software reflects the need for increased integration and sharing of business information throughout an enterprise.

    - **Project portfolio management (PPM).** PPM applications provide a set of integrated functions designed to streamline outward functions and inward processes of project-intensive departments, industries and organizations. They address a majority of the nine processes defined by the Project Management Institute's Project Management Book of Knowledge (PMBoK): project scope management, time management (including scheduling), cost management, resource management, quality management, project communications management, project risk management and project procurement management. The ninth area provides for integration of these processes. PPM applications optimize project delivery by providing, at a minimum, integrated functionality to support project planning, tracking, resource assignment and portfolio reporting.

- **Requirements management**. Requirements management tools streamline development teams' analysis of requirements, capture requirements in a database-based tool to enable collaborative review for completeness, ease use-case or test-case creation, provide traceability and facilitates documentation and versioning/change control. The database approach uses special purpose repositories that are part of the requirements management solution or ship with a general-purpose commercial database integrated with the tool.

## Design

The following is a list of design-related terms and a description of each:

- **Business process analysis (BPA).** This is the fundamental analysis of business processes and management systems to improve them for cost reductions, faster time to market, lowered risk or higher business value. BPA can point out opportunities for optimization, automating manual processes, reducing error cycles and identifying revenue leakage points. It uses objective, quantitative methods and tools to analyze, redesign and transform business processes, including supporting organization structures, information systems, job responsibilities and performance standards. In some cases, BPA could point out the need for wholesale changes implied in a full business process re-engineering (BPR) effort.

- **BPA analytical techniques**. These are mathematical, graphical, logical and managerial algorithms for describing and modeling business processes, information systems or management decision-making systems.

- **BPA methodology.** This is an integrated set of management policies, project management procedures, and modeling, analysis, design and testing techniques for analyzing established business processes and systems; designing new processes and systems; testing, simulating and prototyping new designs before implementation; and managing the implementation process.

- **BPA tools.** These are combinations of techniques and software products that allow electronic capture, analysis, testing, simulation, reconfiguration and persistent memory of business and systems models.

- **BPR.** This is the fundamental analysis and radical redesign of business processes and management systems to accomplish change or performance improvement.

- **Database design**. This includes logical (entity relationship) and physical (table, column and key) design tools for data. Physical data modeling is becoming almost mandatory for applications using relational database management systems (RDBMS). Strong support for physical modeling is paired with facilities to manage multiple models, to submodel or extract from larger models and to reverse-engineer a database design from established tables. Developers are a secondary market often targeted with a subset of the complete functionality.

- **Object-oriented analysis and design (OOA&D)**. These are tools that support object analysis and design technologies and they commonly use Unified Modeling Language (UML) notation with a variety of methodologies to assist in the creation of highly modular and reusable software. Applications, data, networks and computing systems are treated as objects that can be mixed and matched flexibly rather than as components of a system with built-in relationships. As a result, an application need not be tied to a specific system or data to a specific application. Note: The UML standard from Object Management Group has become the defacto standard for OO A&D tools.

## Construction (Development Environments and Suites)

The following is a list of construction-related terms and a description of each:

- **Language-oriented development environments**. Typically, these are development environments built around a compiler and a language, such as COBOL, C/C++, Fortran, ADA and PASCAL, among others. Language-oriented development environments generally include graphical user interface (GUI) builders, debuggers, editors and other utilities that are integrated into the environment.

- **Integrated services environments (ISEs)**. An ISE is a suite of integrated development tools, frameworks and technologies used for building service oriented and composite applications. Most often, these applications will implement a service-oriented architecture (SOA) and will use the techniques of services-oriented development of applications (SODA). ISEs are producer platforms (for creating services), much as application servers are provider platforms (for hosting services) and portal servers are consumer platforms (for using services). Seven basic characteristics of an ISE define the completeness of its support for SODA:

    - Design. Specification of application requirements
    - Modeling. Definition of application structure
    - Fabrication. Writing code, creating components and wrapping legacy resources. Nominally, this is the function of an integrated development environment (IDE).
    - Assembly. Aggregation of components, alignment of inputs to outputs, translation of input or output data
    - Orchestration. Flow control and process management
    - Automation. Hiding complexity and removing the need to write code
    - Variability. Rapid change management. The variability of a system may be inversely proportional to its automation.

- **Traditional (client/server) applications development (AD) tools.** This includes all traditional client/server development environments. They may be client/server fourth-generation languages (4GLs) targeting older technologies (for example, VisualBasic targeting the Microsoft COM/DCOM environment) and third-generation language (3GL) generators for Cobol, C or C++ targeting MVS or Unix.

- **Wireless and mobile AD**. This is the function of creating applications for wireless and mobile devices. The term refers not simply to programming, but to the larger overall process of defining application requirements, planning the application structure, developing the code, monitoring development progress and testing results. In summary, it covers the entire software development life cycle as it relates to a mobile or wireless application.

- **Business rule engines.** Business change has been a constant companion of systems development since the inception of IT, but a growing number of factors has led to the increase of change necessary to remain competitive in business. This increase in the frequency of change is leading to new approaches to alter the business rules embedded in business process flows, applications and even in the enterprise architecture. Enterprises are more pressed to become adaptable and apply the knowledge captured in rule sets to outflank competitors and respond to changing business environments. They can no longer wait for professional programmers to change applications written in traditional programming languages. Business users want to change rules without going through a long-running change process that is, at best, measured in days and, at worst, measured in weeks and months. Therefore, rules engines allow even end-users to make dynamic "real time" changes to their applications in an abstracted level of language.

## Automated Software Quality

Automated software quality (ASQ) includes all of the tools that are used to ensure the design and development of zero-defect software. The ASQ category consists of automated testing tools, software test management tools and hosted testing services.

- **Automated testing**. Automated testing applies commercially or internally developed software or services to assist in the testing process. Automated tests provide consistent results and data points. The benefits are repeatability, ease of maintenance, the ability to efficiently use resources in off-peak hours and the capability to create reports based on the executed tests.

- **Software test management.** This involves the planning and monitoring of the testing effort and includes control of the testing resources. From a tools perspective, test management is the collection, display, and control of tests and resources from one central repository. From a methodology point of view, test management is essential for developing repeatable processes.

- **Hosted testing services.** A subset of hosted development services, hosted testing services offer a secure, scalable, reliable and portable environment for testing applications. The hosted testing service may also offer test execution, in addition to providing the infrastructure and the tool. The most common use for hosted testing services is load testing.

## Operational Life Cycle

The following is a list of operational life cycle-related terms and a description of each:

- **Component-based development (CBD).** This is a set of reuse-enabling technologies, tools and techniques that allow AD organizations to go through the entire AD process (for example, analysis design, construction and assembly) or through any particular stage via the use of predefined component-enabling technologies (such as AD patterns, frameworks and design templates), tools and application building blocks.

- **Legacy understanding**. The legacy understanding market includes tools that may operate on a mainframe or a workstation. These tools provide system inventory, program understanding, code modification, relationship mapping of application artifacts, in-depth data and control flow, and code modification and testing.

- **Legacy extension**. The broader legacy extension market consists of enterprises looking for a noninvasive approach to accessing established systems data and logic. The capabilities offered by these tools are growing beyond the simple one-to-one presentation translation more typically associated with screen scrapers. The operational requirements of these products are growing, reflecting the increased need for performance and reliability. They include high volume capability, multiple server options, multiple threading, traffic monitoring, logging capability, visual tracing, integration with workflow and multiple emulator options. Development requirements include normal AD capabilities, semi-automated screen capture, rule/template screen capture, semi-automated maintenance, open interfaces, shared development, many-to-many screen mapping, dynamic value-added graphics, and key and cursor control. Technologies included in the legacy extension segment include:

    - Presentation integration servers, which provide noninvasive processes to transform character-based presentation formats into graphic presentations through Windows or Java GUI- and browser-based clients.

- Programmatic integration servers, which provide a mechanism to package defined sequences of character-based data into programmatic representations, such as COM, Java or Extensible Markup Language (XML), to supply legacy information to new composite applications.

- **Legacy transformation**. The legacy transformation market consists of tools that provide business rule identification, code slicing, code modification or transformation from one language to another. These products are generally provided as add-ons to legacy-understanding tools and may operate on a mainframe or a workstation. Other sophisticated tools can also support language wrapping for creating components out of legacy systems and support the porting legacy business logic to new architectures and languages.

- **Legacy application or system**. This is an information system that may be based on older technologies, but is critical to day-to-day operations. Replacing legacy applications and systems with systems based on new and different technologies is one of the IS professional's most significant challenges. As enterprises upgrade or change their technologies, they must ensure compatibility with old systems and data formats that are still in use.

- **IT metadata repository.** This may be thought of as the "inventory management system for the IT department." It tracks "artifacts" of importance to the IS organization. It is capable of storing descriptions and behaviors of objects in an enterprise, including requirements, policies, processes, data, software libraries, projects, platforms and personnel, with the potential of supporting software development and operations management. As a single point of definition for all system resources, it should stimulate program and installation management productivity. It can be a "hub" to consolidate "metadata" from a variety of other tools —modeling tools, IDEs, legacy environments and packages, data warehousing tools and others.

- **Software change and configuration management (SCCM)**. SCCM is a set of disciplines to stabilize, track and control an agreed upon set of software items. This includes version management, change management, defect tracking, change automation and other related processes.

## Appendix C

# Analyzing Existent Taxonomies

The software market segments covered contain a wide variety of products and technologies. To better understand the range of vendors and products included, many companies of consulting provide taxonomies for the market that helps their customers to make right decision.

These companies, focusing on the market of tools from various perspectives, sometimes do not have the purpose of creating a taxonomy that may be helpful for selection of tools, but also more oriented to business aspects or financial profits. We analyze in this section, some of these taxonomies that are oriented to Software Application Development built by well-known companies, with three main objectives:

- to know the needs of the actual market,
- to identify and to understand the attributes that were taken into account for characterizing each segment, and
- to recognize if it is feasible that we can use any of them like a basis for the construction of another new by means of its reuse.

The segmentation of software infrastructure products and application packages represent significant challenges, including:

- Products are very often used in ways that differ from their intended purpose.
- The positioning of a product by a vendor may not match the actual functionality of the product.
- Products sets and suites are evolving and devolving. Products are, therefore, moving across segment boundaries, and new segments must be created and old segments must be revised.
- In many markets, the lines between segments are blurring.
- Important attributes, which may be of interest in their own right, do not necessarily, constitute a unique market.

We present some existing taxonomies proposed by many consultants companies or standard organizations (in our case, we want to achieve a proposal taxonomy with a different purpose, more focused to arrange the knowledge, not practical business). Thus, we will evaluate the utility of each of them in practical selection of COTS.

## **GARTNER** [1], [19],[20]

Gartner´s classification of a product takes into account a variety of factors, including the technical features of the product, target audience, competitive positioning and perceived usage of the product by customers.

Gartner Dataquest breaks the software industry into logical segments, which allows for in-depth and segment-specific research. These segments are:

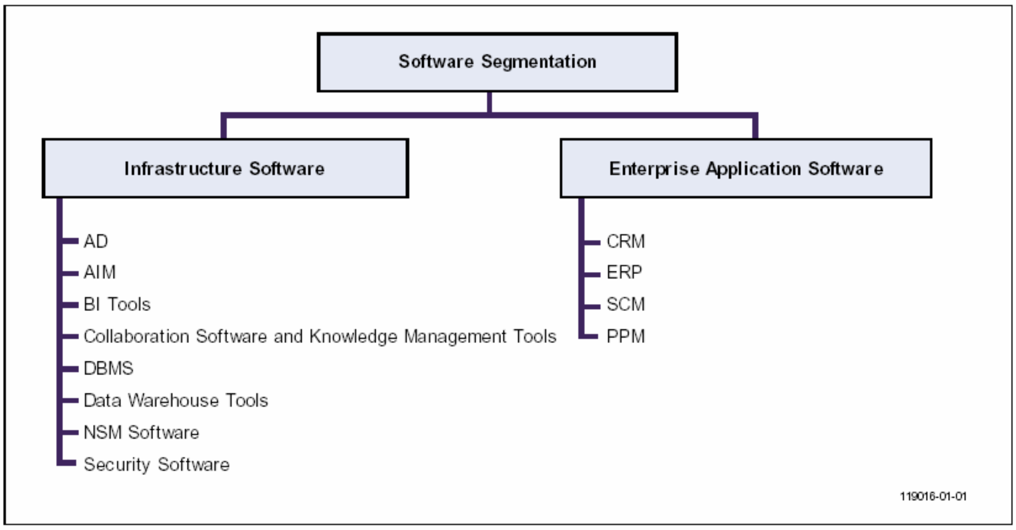
- Infrastructure Software
- Enterprise Application Software
- Emerging and Merging Markets

### Infrastructure Software

The focus of infrastructure software is to increase the performance of IT resources. In this category, they gather software primarily for use by IT professionals. This include: AD Software, Application Integration and Middleware, BI Tools, Collaboration Software and Knowledge Management Tools, Data Warehouse Tools, Database Management Systems, Embedded Software Tools, Network and Systems Management Software, Security Software, and Storage Management. (Fig. 3-3)

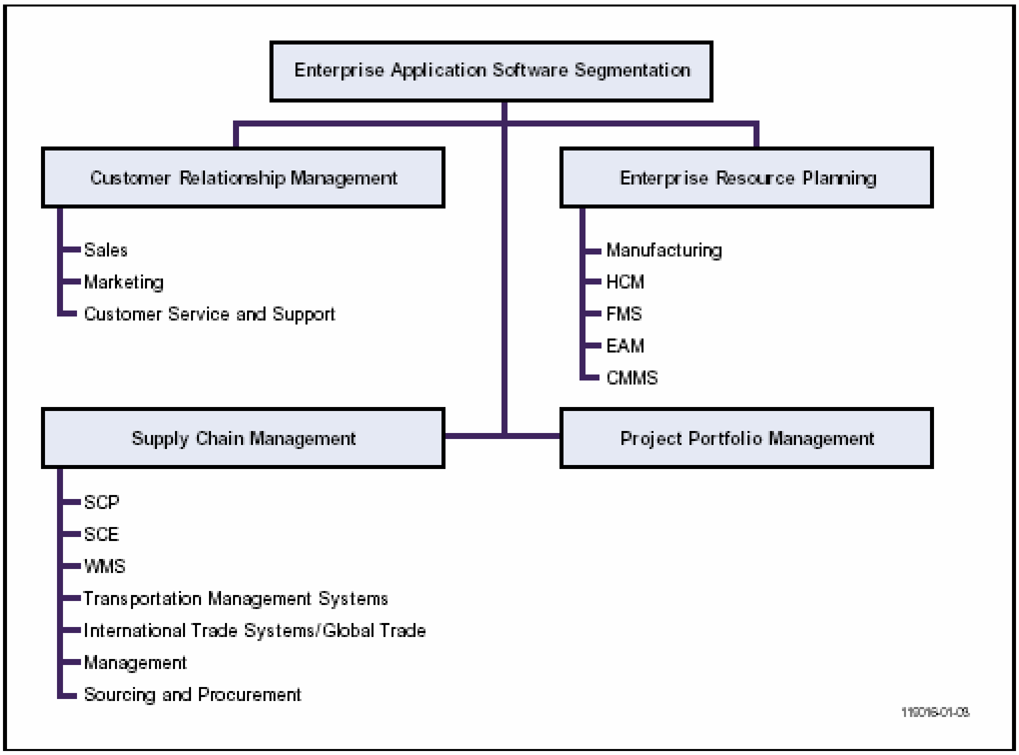
### Enterprise Application Software

The focus for application software is to increase the performance of business or personal resources. It enables users to leverage the power of computers towards achievement of their business, professional or personal objectives or goals. This include: Customer Relationship Management, Enterprise Resource Planning, Supply Chain Management, Project Portfolio Management, and Design and Engineering. (Fig. 3-2)



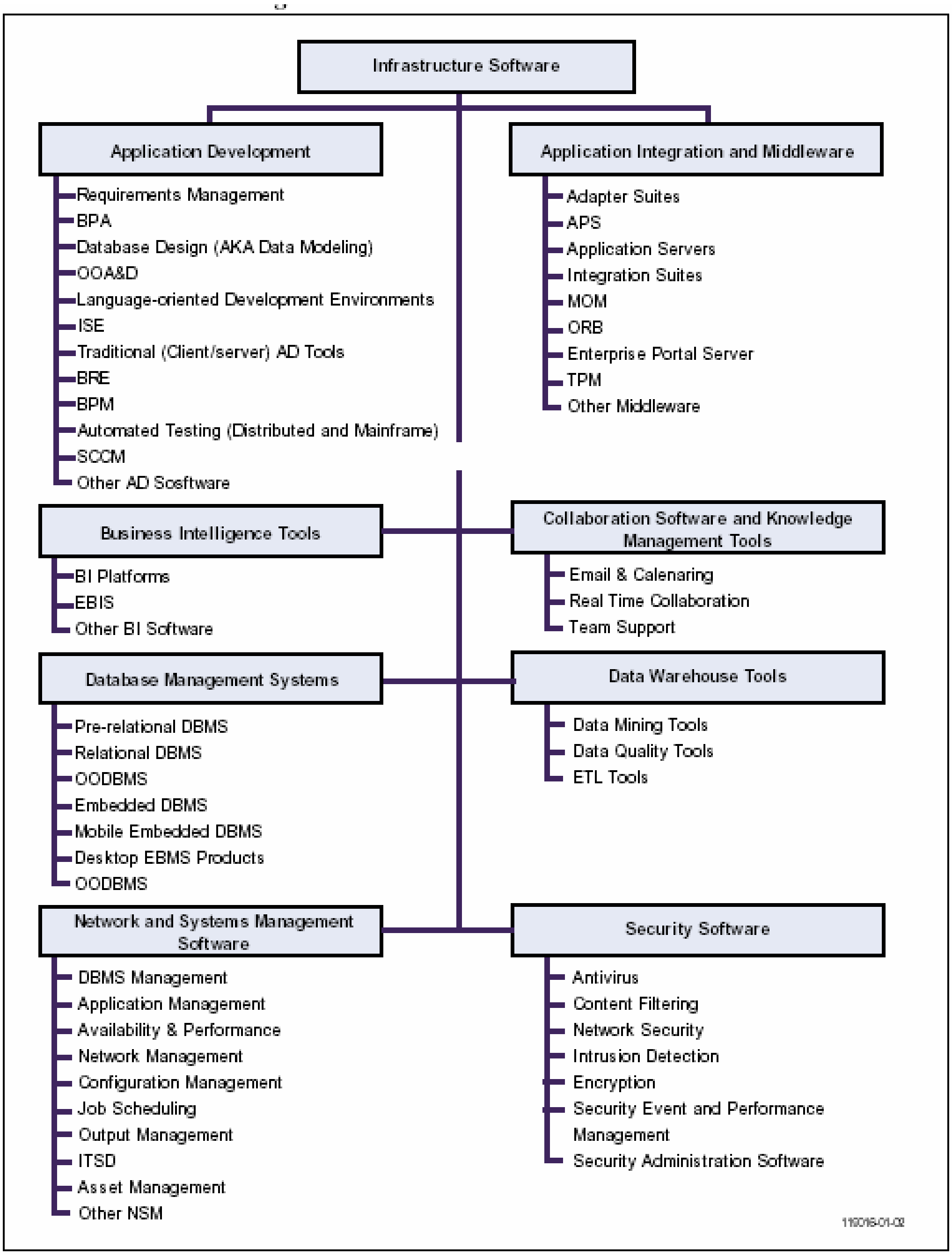
Source: Gartner Dataquest (December 2003)

Fig.3-1 Software Segmentation Structure


Enterprise Application Software Segmentation
Customer Relationship Management
Sales
Marketing
Customer Service and Support
Enterprise Resource Planning
Manufacturing
HCM
FMS
EAM
CMMS
Supply Chain Management
SCP
SCE
WMS
Transportation Management Systems
International Trade Systems/Global Trade
Management
Sourcing and Procurement
Project Portfolio Management
119016-01-03

Source: Gartner Dataquest (December 2003)

Fig. 3-2 Software Segmentation Structure



Source: Gartner Dataquest (December 2003)

Fig. 3-3 Infrastructure Software Segmentation Structure

## Emerging and Merging Markets.

The previous two subjects defined mutually exclusive product markets, which together provide insight into how the money flows through the software industry. However, many of the most interesting and challenging opportunities for vendors have other dimensions. Some are a composite mix of baseline sectors; others are virtual concepts or new technology standards, not real markets. Still others take less conventional forms, such as the application service provider (ASP) and business service provider (BSP) markets. All of them overlap in some way with other markets.

## Composite Markets

A composite market is one not necessarily tracked and measured on an ongoing basis, but that offers an alternative view or cut of our baseline market data (defined in Infrastructure Software or Enterprise Application Software). A composite market definition can be useful for tracking a temporary phenomenon. Occasionally, a composite market definition foretells the emergence of a long-lasting baseline Gartner Dataquest market definition at a later date.

New composite markets are always emerging. Rather than continually reorganize the component segmentation, Gartner will address new markets as exceptional or fringe until they are clearly an established and well-defined market segment in their own right.

### Business Activity Monitoring

BAM is neither a market nor a product. It is a concept, such as quality or knowledge management, and it is not new. BAM solutions focus on crossbusiness processes rather than divisional-, departmental- or technology-specific processes. The scope of integration in BAM solutions expands far beyond the four walls of a plant or a division, and real time is not necessarily nanoseconds but rather is determined by the requirements of the business process. It brings the near real-time world of the BI operational data store together with NSM monitoring and BPM through integration brokers and shared messaging.

### Collaborative Commerce

(C-commerce) involves the collaborative, electronically enabled business interactions among an enterprise's internal personnel, business partners and customers throughout a trading community. The trading community can be an industry, industry segment, supply chain or supply chain segment. For some enterprises, c-commerce is already a fact of business life, but how can they measure it? It is not yet quantifiable with any consistency. It is not a class of software and it is difficult even to define.

### Corporate Performance Management

(CPM) is a term that describes the methodologies, metrics, processes and systems used to monitor and manage the business performance of an enterprise. Applications that enable CPM translate strategically focused information to operational plans and send aggregated results. These applications are also integrated into many elements of the planning and control cycle, or address BAM or customer relationship optimization (CRO) needs. CPM applications enable sharing of information across and even beyond the borders of the enterprise, to all employees, business partners shareholders and most importantly, customers.

### Contract Management

Is the identification and efficient management of all contracts with suppliers. Applications that manage companies' contracts weave themselves through sourcing and procurement applications. Although contract management is an integral part of sourcing (creation) and procurement (enforcement), Gartner has separated it out as a segment but acknowledge the interdependencies. Contract management aides sourcing and procurement through negotiation, generation and non-repudiation, abstraction, versioning and archiving, compliance management and exception alert generation, and dispute resolution.

### DBA Software Tools

Database administration (DBA) tools encompass software products including systems management extensions for monitoring the health of databases, DBA tools for SQL tuning, space optimization, schema change management, back up and recovery products for files as well as database-aware recovery products, data warehouse administration tools, and certain database development tools often used by database administrators. This market could also be called the DBMS aftermarket.

### ERP II and Enterprise Application Suite

The emerging next generation of ERP strategy is called ERP II by Gartner Dataquest. ERP II is an application and deployment strategy that expands out from ERP functions to achieve integration of an enterprise's key domain-specific, internal and external collaborative, operational and financial processes.

### Mobile and Wireless Packaged Applications Software

The terms mobile application and wireless application are bandied about in the trade press, often used interchangeably, and can refer to simple stand-alone software or to internet-worked processes of great complexity. Mobile and wireless applications can mean anything from a system on a smart phone, a calendar or a tic-tac-toe game on a PDA to Internet/corporate e-mail connectivity up to a sales force automation order-entry system that updates back-end databases over a wireless link.

### Mobile and Wireless Infrastructure Software Platforms

Here are development tools and deployment servers for creating brand new customer mobile applications or for "mobilizing" established conventional enterprise applications, e-mail and enterprise data stores.

### Order Management

It is a business process and not a specific market. Much of the functionality attributed to order management is embedded within and touches components within the CRM (Customer Relationship Management), ERP (Enterprise Resource Planning) and SCM (Supply Chain Management) markets as it guides products and services through order entry, processing and tracking.

- **Product Life Cycle Management**: At its highest definitional level, product life cycle management (PLM) is a process for guiding products from idea through retirement to deliver the most business value to an enterprise and its trading partners. PLM employs product information and business analysis to support product portfolio strategy, product life cycle planning, management of activities, and execution of those activities through each phase in a product's life. The applications that support the business activities enabled through PLM includes product ideation, design, engineering, manufacturing process management, product data management, and product portfolio management.

### Smart Enterprise Suites

Include a significant combination of technologies, such as content management, team support, a portal framework and information retrieval. They may also include expertise location and management, community technology, process management, and multichannel access.

### Supplier Relationship Management

There has been much ado for nearly two years about the user benefits of supplier relationship management (SRM). With much fanfare and accolades from the software industry, there is a still confusion and varying opinion on what exactly SRM is and what it should encompass. So, is there an SRM software market? No, not yet. There are a plethora of software applications that support SRM initiatives but, to date, no true SRM applications. SRM is a logical extension of SCM. SRM is the latest of a series of innovations within SCM and represents an incremental development. As such, SRM is a subset of SCM. SRM represents an evolutionary extension of SCM, driven by the need for enterprises to create a more comprehensive life cycle view of suppliers' operational contribution to the top and bottom lines. Consequently, SRM is an extension of, and builds on, SCM technologies and practices.

### Web Services Software

Deploying Web services-enabled software will be an evolutionary process, not a revolutionary one. While the majority of software vendors have committed to supporting Web services software standards within their established product lines, it will take more than four years to evolve these immature standards, build up skills, and plan, build and test for new versions of software that gradually incorporate these standards. Web services standards will be deployed through multiple markets, such as integration suites, AD tools and some enterprise application segments.

### Other Markets

- **ASPs and Application Hosting**: This is a service addressing the life cycle needs of the application from the initial IT infrastructure development to maintenance of a complete set of IT business applications. The provider offers software maintenance, conversion, enhancement and support in a hosted environment. This is considered an indirect channel for software delivery. This sector is also researched within Gartner Dataquest's IT services practices.

- **Open Source Software:** The focus of Gartner is to measure the direct value of software markets. However, free software can and does have a significant influence on the dynamics of demand and supply, and it influences the Gartner´s research agendas.

- **Software Solutions Markets:** Software products are often bought and sold as part of a larger package containing some mix of hardware, software, services, expertise, information content and financing service. To implement, operate and use the solutions requires investment in internal resources. The methodology behind of the market data analysis allows to Gartner to unbundle these elements so that they are counting or comparing similar things. Most of their solutions market research is clustered into analyst teams for global industry vertical market analysis and enterprise solutions.

- **New Categories:** New markets are always emerging, which are either the result of the bundling of several component markets or the arrival of a new layer of tools on top of established technology. Rather than continually reorganize the component segmentation, Gartner will address new markets as exceptional or fringe until they are clearly an established and well-defined market segment in their own right.

The Gartner research covers software vendors worldwide by selected software categories. Based in this research, they develop and maintain a database of information on software supply by vendor, revenue, region and software segment. In addition they also analyze segment and vendor revenue by platform, vertical industry, enterprise size and sales channels (direct, indirect and others) for most of the markets.

The definitions and factors for segmentation are revised, altered or expanded each year to reflect changes in software technologies and the software marketplace.

As a result they have a very complete taxonomy of the entire market. It is very easy the navigability and find any related topic. In addition, we can find the dependencies or lapping of the segment with other segments.

Next, we analize the taxonomy proposed by Gartner for Application Development:

## Gartner Application Development Taxonomy

The categories for which application development software is analyzed are comprehensively defined here for the purpose of providing clarity and guidance.

This taxonomy is available on internet, and we can search the actual research and products depend on our requirements of information. In other words, you can find very easy many products and papers related with your subject.

## GARTNER

### Application Development (2580)

| L1 | L2 | L3 | L4 | L5 | Subcathegory |
|---|---|---|---|---|---|
| 1 | | | | | Application Development Tools (870) |
| 1 | 1 | | | | Development Life Cycle Software (107) |
| 1 | 1 | 1 | | | Debugging and Defect Tracking Tools (6) |
| 1 | 1 | 2 | | | Software Configuration Management Tools (9) |
| 1 | 1 | 3 | | | Migration and Porting Tools (7) |
| 1 | 1 | 4 | | | Software Process Management (5) |
| 1 | 1 | 5 | | | Methodware(2) |
| 1 | 1 | 6 | | | Test Software (32) |
| 1 | 1 | 6 | 1 | | Software Testing (47) |
| 1 | 2 | | | | Development Enviroment and Suites Software (154) |
| 1 | 2 | 1 | | | Integrated Development Environments (20) |
| 1 | 2 | 2 | | | Modeling and Code Generation (18) |
| 1 | 2 | 3 | | | CASE Tools(3) |
| 1 | 3 | | | | Programming Language Software (85) |
| 1 | 4 | | | | Development Infraestructure Software (16) |
| 1 | 5 | | | | Embedded Software Tools (72) |
| 1 | 6 | | | | Rapid Application Development Tools (13) |
| 1 | 7 | | | | Web Site Development Tools (135) |
| 1 | 8 | | | | Web Services Production Platforms |
| 2 | | | | | Application Development Governance (104) |
| 2 | 1 | | | | Application Development Metrics (16) |
| 2 | 2 | | | | Capability Maturity Model (29) |
| 2 | 3 | | | | Software Process Improvement (20) |
| 3 | | | | | Application Development Methodologies (482) |
| 3 | 1 | | | | Legacy Application Extension Methodologies (135) |
| 3 | 2 | | | | Object-Oriented Development Methodologies (12) |
| 3 | 3 | | | | Reuse and Component-Based Methodologies (43) |
| 3 | 4 | | | | Agile Development Methodologies (29) |
| 3 | 5 | | | | Application-Specific Methodologies (215) |
| 4 | | | | | Application Deployment (51) |
| 4 | 1 | | | | Application Deployment for Wireless and Voice (10) |
| 4 | 2 | | | | Application Deployment for E-commerce and Internet (9) |
| 4 | 3 | | | | Application Deployment for Legacy and Client/Server |
| 4 | 4 | | | | Deployment and Distribution (94) |
| 5 | | | | | Application Development Project Management (261) |
| 5 | 1 | | | | Projetc Management Office (16) |
| 5 | 2 | | | | Service Level Agreements (7) |
| 5 | 3 | | | | Software Configuration Management Tools (6) |
| 5 | 4 | | | | Configuration Management Software (71) |
| 5 | 5 | | | | Project Portfolio Management (171) |
| 5 | 5 | 1 | | | Professional Service Automation (45) |
| 5 | 5 | 2 | | | Service Optimization Management (38) |
| 5 | 5 | 3 | | | Portfolio Management Software (16) |
| 5 | 5 | 4 | | | Project Management Software (17) |
| 5 | 5 | 5 | | | Services Process Analytics and Forecasting (3) |
| 6 | | | | | Application Development Quality (132) |
| 6 | 1 | | | | Software Testing (47) |
| 6 | 2 | | | | Debugging and Defect Tracking Tools (6) |
| 6 | 3 | | | | Test Software (32) |
| 7 | | | | | Application Architecture (582) |
| 7 | 1 | | | | J2EE (33) |
| 7 | 2 | | | | .Net (88) |
| 7 | 3 | | | | Service-Oriented Architecture (85) |
| 7 | 4 | | | | Service-Oriented Development of Applications (64) |
| 7 | 4 | 1 | | | Composite Applications (27) |
| 7 | 5 | | | | Open Source Applications (85) |
| 7 | 6 | | | | Enterprise IT Architecture (216) |
| 7 | 7 | | | | Web Services (505) |
| 7 | 7 | 1 | | | Web Services Security (26) |
| 7 | 7 | 2 | | | Web Services Production Platforms (19) |
| 7 | 7 | 3 | | | Web Services Management Platforms (16) |
| 8 | | | | | Custom Application Development Services (131) |
| 9 | | | | | Software Configuration Management Tools (9) |
| 10 | | | | | Portfolio Management (60) |

## **INCOSE** [2]

INCOSE is an international organization formed to develop, nurture, and enhance the interdisciplinary approach and means to enable the realization of successful systems. INCOSE works with industry, academia, and government to:

- provide a focal point for dissemination of systems engineering knowledge
- promote collaboration in systems engineering education and research
- assure the establishment of professional standards for integrity and in the practice of systems engineering
- encourage governmental and industrial support for research and educational programs that will improve the systems engineering process and its practice

They work in many areas, the most important for this report, is concerned with COTS tools. In that sense, they maintain a database with information related with COTS. They provide by means the Modeling and Tools Technical Committee MTTC the constituency with information regarding COTS systems engineering SE tools, as well as guidance regarding the development and use of models constructed with such tools. In addition, they are the resource center for tools standards--including tool exchange and emerging modeling standards.

The MTTC consists of some seventy Software Engineers that are working within three technical areas supported by the following structure:

- SE Tools Database Working Group: Deliver a tools comparison/information database for requirements management tools.
- Tools Integration Working Group: Promote tools interoperability and provide concepts for integration of COTS SE tools, develop tool information exchange standards.
- Model-Driven System Working Group: Develop and disseminate executable representations of the systems engineering process, including schemas for tools integration.
- Soft Systems Engineering Working

The Tools Database Working Group's is bringing online an automated tools survey that will allow the tool vendors to directly update their tool information (rather than wait for the TDWG to get together and post the information). The user is notified if the information was verified or not, because the valuation by the vendors can be "partial")

INCOSE shows (as a free access) a taxonomy based on a database of products.

The taxonomies that they present (in fact, they arrange the information from his database by means standards like IEEE-1220) are so general.

| INCOSE (IMPIG TAXONOMY) | | | | | |
|---|---|---|---|---|---|
| Systems Engineering | | | | | |
| L1 | L2 | L3 | L4 | L5 | Subcathegory |
| 1 | | | | | Management |
| 1 | 1 | | | | Configuration Management Tools |
| 1 | 2 | | | | Work Flow Management Tools |
| 1 | 3 | | | | Risk Management Tools |
| 1 | 4 | | | | Cost Estimating & Tracking Tools |
| 1 | 4 | 1 | | | Cost Estimating Tools |
| 1 | 4 | 2 | | | Cost Tracking Tools |
| 1 | 5 | | | | Defect Tracking Tools |
| 2 | | | | | Engineering Tools |
| 2 | 1 | | | | System Design Tools |
| 2 | 1 | 1 | | | System Modeling Tools |
| 2 | 1 | 1 | 1 | | Structural Modeling Tools |
| 2 | 1 | 1 | 2 | | Behavioral Modeling Tools |
| 2 | 1 | 1 | 2 | 1 | Static Behavioral Tools |
| 2 | 1 | 1 | 2 | 2 | Dynamic Behavioral Tools |
| 2 | 1 | 1 | 3 | | HMI Prototyping Tools |
| 2 | 1 | 2 | | | Design Support Tools |
| 2 | 1 | 2 | 1 | | Simulation Tools |
| 2 | 1 | 2 | 2 | | Numerical Analysis Tools |
| 2 | 1 | 2 | 3 | | Domain Specific Tools |
| 2 | 1 | 2 | 4 | | Measures of Effectiveness Tools |
| 2 | 2 | | | | Requirements Engineering Tools |
| 2 | 2 | 1 | | | Requirements Management Tools |
| 2 | 2 | 1 | 1 | | Requirements Classification Tools |
| 2 | 2 | 1 | 2 | | Requirements Capture & Identification Tools |
| 2 | 2 | 1 | 2 | 1 | Textual Requeriments Capture Tools |
| 2 | 2 | 1 | 2 | 2 | Tools for Elicitation of Requirements |
| 2 | 2 | 1 | 3 | | Requirements Traceability Tools |
| 2 | 2 | 2 | | | Requirements Generation Tools |
| 2 | 3 | | | | Design Validation Tools |
| 2 | 3 | 1 | | | Thread Analysis Tool |
| 2 | 3 | 2 | | | Test Validation Planning Tools |
| 2 | 3 | 3 | | | Scenario Validation Tools |
| 2 | 3 | 4 | | | Tools to Validate System Compliance/Requirements |
| 2 | 3 | 4 | 1 | | Measurement Tools |
| 2 | 3 | 4 | 2 | | Performance Analysis Tools |
| 2 | 4 | | | | Specialty Engineering Tools |
| 3 | | | | | Information Sharing (and Process Tools) |
| 3 | 1 | | | | Communication Tools |
| 3 | 1 | 1 | | | Interpersonal Communications Tools |
| 3 | 1 | 2 | | | Network Information Retrieval Tools |
| 3 | 1 | 2 | 1 | | Directory Services |
| 3 | 1 | 2 | 2 | | File Transfer |
| 3 | 1 | 2 | 3 | | Information Services |
| 3 | 1 | 2 | 4 | | World Wide Web |
| 3 | 2 | | | | Data Analysis Tools |
| 3 | 2 | 1 | | | Spreadsheet Tools |
| 3 | 2 | 2 | | | Data Reduction Tools |
| 3 | 2 | 3 | | | Data Visualization Tools |
| 3 | 3 | | | | Electronic Publishing Tools |
| 3 | 3 | 1 | | | Automatic Document Generation Tools |
| 3 | 3 | 2 | | | Technical Publishing System |
| 3 | 3 | 3 | | | Technical Ilustration Tools |
| 3 | 3 | 4 | | | File Format Converters |
| 3 | 4 | | | | Electronic Viewing Tools |
| 3 | 4 | 1 | | | Database End User Viewers |
| 3 | 4 | 2 | | | Formatted Document Viewers |
| 3 | 5 | | | | Tool Integration Facilities |
| 4 | | | | | Infrastructure Support |
| 4 | 1 | | | | System Administration |
| 4 | 2 | | | | Network Support |
| 4 | 3 | | | | Product Data Management |
| 4 | 4 | | | | |

Through a collaborative effort between the INCOSE Information Model & Process Interest Group (IMPIG) and the Tools Database Working Group they have merged the Tools Taxonomy graphics and descriptions with the Tools database of information related to Commercial off-the-shelf (COTS) and Government off-the-shelf (GOTS) software of interest to Systems Engineers.

## Definitions Related with the Construction the of IMPIG Taxonomy.

**Configuration Management Tools**: Configuration Management is responsible for managing system baselines, both products of the engineering process and the tools used to create those products. Configuration Management incorporates version control, baseline control, change history, check-in check-out control, change and archiving capabilities of both process and product.

**Work Flow Management Tools**: Work Flow Management tools are used to define and schedule tasks, and make work assignments. They help monitor progress, allow for prioritization of tasks, and support task review/approval. Workflow Management should also identify resource loads and critical paths. Automation can help enforce the Integrated Product Development Process through the use of pre-defined workflow models.

**Risk Management Tools**: Risk Management techniques support qualitative risk identification and quantitative risk analysis and visualization. It is used to manage technical, cost, and schedule risks, including the ability to meet performance requirements and measures of effectiveness. Risk Management supports management decisions, identifies risk prioritized by program impact, and generates risk mitigation plans.

**Cost Estimating & Tracking Tools**: Cost Estimation and Tracking techniques are used to estimate the cost of tasks and to roll those costs up to total project cost. They require input information that provides Basis of Cost parameters from prior experience. They are also used to track the cost of the project during development and must interface with the organization's financial recording and reporting system.

**Cost Estimating Tools**: Develop system pricing considering design complexity, domain knowledge, historical cost information, and cost objectives.

**Cost Tracking Tools**: Means of collecting costs, and estimating Budgeted Cost of Work Scheduled (BCWS), Budgeted Cost of Work Performed (BCWP), Actual Cost of Work Performed (ACWP), etc.

**Defect Tracking Tools**: Defect Tracking techniques are used to record system defects and change requests found in product testing and in the field. These techniques track the date the issue was identified, assigned, removed, tested, and verified. They also record a description of the defect, its priority and severity, information about its method of removal, potential work-arounds if the defect can not be removed, etc.

**Behavioral Modeling Tools**: Behavioral Modeling falls into two categories, static and dynamic. Behavioral Modeling helps model the functional capabilities of a system and its response to events.

**Static Behavioral Tools**: Static Behavior Modeling presents a static description of the behavior of the system. Static implies that the model may represent dynamic behavior, but it is not executable. Examples of static behavioral models are Functional Flow Block Diagrams, State Transition Diagrams, and Object Interaction Diagrams.

**Dynamic Behavioral Tools**: Dynamic Behavior Modeling provides a dynamic description of the behavior of a system. These models are executable through simulation and prototyping.

**HMI Prototyping Tools**: Human Machine Interface (HMI) Prototyping techniques are used to develop or verify HMI design by modeling the interaction of operators or users with the system. This work is particularly useful when performed early and reviewed with users to verify requirements and assumptions about the system. Automation provides the capability to describe input/output screens, to depict parts of the system and to animate these presentations.

**Simulation Tools**: Simulation techniques are used for detailed subsystem modeling. Since it is generally impractical to model the entire system to a detailed level, selective employment of these techniques for the purpose of minimizing risk is recommended.

**Domain Specific Tools**: Domain Specific includes a range of general purpose methods and techniques that are directed at a specific physical phenomenon. The subclass includes thermal/heat transfer, fluid flow, power dissipation, satellite performance/orbit injection models, sonar performance models, etc. that do not fall under the category of general purpose numerical analysis. Many of these methods are applied to complex boundary value problems and apply finite element or finite difference methods.

**Measures of Effectiveness Tools**: Measures of Effectiveness support the definition of a parts tree and the budgeting of performance attributes among the components in the tree. Evaluation of measures of effectiveness occurs during trade-off analysis. Agility for comparing various design options is helpful.

**Requirements Classification Tools**: Requirements Classification Tools help the engineer classify the requirements based on work to be done so that the requirement analysis activity can be scheduled and tracked. They help the engineer classify based on how the requirements will be used in modeling so that completeness of traceability can be monitored.

**Requirements Capture & Identification Tools**: Requirements Capture Tools accept text information from heritage sources, users, customer requirements and customer operations concepts. They assemble the information. They assist the engineer in finding relationships among entities in the information and in moving among the entities. Modern forms of these tools use natural language processing to interpret the text and hypertext technology to automatically build electronic hypertext versions of the text. These tools also accept requirements change documents and, assist in establishing where these changes impact earlier documents. Requirement Identification Tools aid the engineer in separating requirements in the information before him from extraneous information. Modern versions of these tools use natural language processing to identify statements containing imperatives of any kind in the information.

**Tools for Elicitation of Requirements**: These tools assist the requirements engineer in drawing out requirements from system stakeholders. They include survey and interview tools.

**Requirements Traceability Tools**: Requirement Traceability Tools enable the engineer to link requirements to their source, to changes in requirements, and to modeling elements that satisfy the requirements. They provide traceability among the successive documents that are used to review the system development.

**Requirements Generation Tools**: Requirements generation tools utilize system simulation results, performance allocations, mission scenarios, and design constraints to generate lower level requirements in an organized and traceable manner.

**Thread Analysis Tools**: Thread Analysis Tools track individual threads of stimulus/response behavior through the system so that validation of performance can be specified. They are also used for tracing response threads through all levels of system behavior to validate continuity, to identify overlapping use of resource by multiple threads, and to confirm interface specification.

**Measurement Tools**: Measurement tools provide users with the capability to collect analyze and report on engineering quality.

**Data Visualization Tools**: Data Visualization tools present complex data, sometimes with high dimensionality, to the evaluator in an understandable way.

**Automatic Document Generation Tools**: Automatic Document Generation tools build documents from a set of textual, tabular, and graphical objects in file systems or databases. The document generator usually follows a set of rules for building the document and receives style and formatting information from a document style sheet.

**Technical Publishing System**: Technical Publishing Systems support the creation, composition and assembly of large documents. They provide a "What

You See Is What You Get" (WYSIWYG) word processing environment making it easy to flow text around tables, figures, and equations. They also separate format and content editing, providing documents with a uniform look and feel even when multiple authors are working on a document. Some of the more sophisticated systems provide markup capabilities for reviews as well as revision management capabilities.

**Technical Illustration Tools**: Draw, Paint, and Technical Illustration tools are used for creating presentations, engineering drawings, and high quality artwork containing raster images, continuous-tone images, and vector line art. These packages allow users to export their illustrations to a variety of Technical Publishing Systems, printers, phototypesetters, and graphical file formats.

**File Format Converters**: File Format Converters convert files from one format to another. These files may be text, graphics, or multimedia.

**Database End User Viewers**: End-User Database Viewers provide users with form-based access to Management Information Systems. These systems generally provide access to a wide spectrum of information from engineering and finance to personnel information and benefits.

**Formatted Documents Viewers**: Formatted Document Viewers allow you to replace paper documents and drawings with "What You See Is What You Get" (WYSIWYG) electronic pages providing intelligent access to document archives. Query mechanisms, hyperlinks, and bookmarks are incorporated to reduce the cycle time for finding the required information. Some of these systems allow users to add notes to the database.

**Tool Integration Facilities**: Tool Integration Facilities are tools or environments that facilitate data interchange or formatting.

## INCOSE taxonomy using SE (Software Engineering) Tools Database

Other way to view the SE Tools database is via the EIA(Electronic Industries Alliance)-632 Process : "Processes for Engineering a System".

**INCOSE (EIA-632 Taxonomy)**

| L1 | L2 | L3 | Subcathegory |
|---|---|---|---|
| 1 | | | Technical Management Process Tools |
| 1 | 1 | | Planning Process Tools |
| 1 | 2 | | Assessment Process Tools |
| 1 | 3 | | Control Process |
| 1 | 3 | 1 | Outcomes Management |
| 1 | 3 | 2 | Information Dissemination |
| 2 | | | Acquisition and Suplí |
| 2 | 1 | | Supply Process |
| 2 | 2 | | Acquisition Process |
| 3 | | | System Design |
| 3 | 1 | | Requirements Definition |
| 3 | 2 | | Solution Definition |
| 4 | | | Product Realization |
| 4 | 1 | | Implementation Process |
| 4 | 2 | | Transition to use |
| 5 | | | Technical Evaluation |
| 5 | 1 | | Systems Analysis Process |
| 5 | 2 | | Requirements Validation Process |
| 5 | 3 | | System Verification Process |
| 5 | 4 | | End Products Validation Process |

Another way to view the SE Tools database is via the IEEE-1220 Process : "Standard for Application and Management of the Systems Engineering Process".

**INCOSE (IEEE-120)-Requirements Análisis**

| L1 | L2 | Subcathegory |
|---|---|---|
| 1 | | Requirements Análisis |
| 2 | | Requirements Validation |
| 3 | | Functional Análisis |
| 4 | | Systems Análisis |
| 4 | 1 | Requirements Trade Studies and Assessment |
| 4 | 2 | Functional Trade Studies and Assessment |
| 4 | 3 | Design Trade Studies and Assessment |
| 5 | | Functional Verification |
| 6 | | Síntesis |
| 7 | | Physical Verification |
| 8 | | Control Process |
| 8 | 1 | Configuration Management |
| 8 | 2 | Data Management |
| 8 | 3 | Interface Management |
| 8 | 4 | Performance Based Progress Measurement |
| 8 | 5 | Risk Management |

## **IDC (International Data Corporation)** [12]

IDC is a global market intelligence and advisory firm in the information technology and telecommunications industries. They analyze and predict technology trends so that their clients can make strategic, fact-based decisions on IT purchases and business strategy.

They do not have (at least with free access) a formal taxonomy (in the sense that we are analyzing) for arranged the tools in a domain specific.

Respect to Application Development, they arrange their research in the topics showed in the next table.

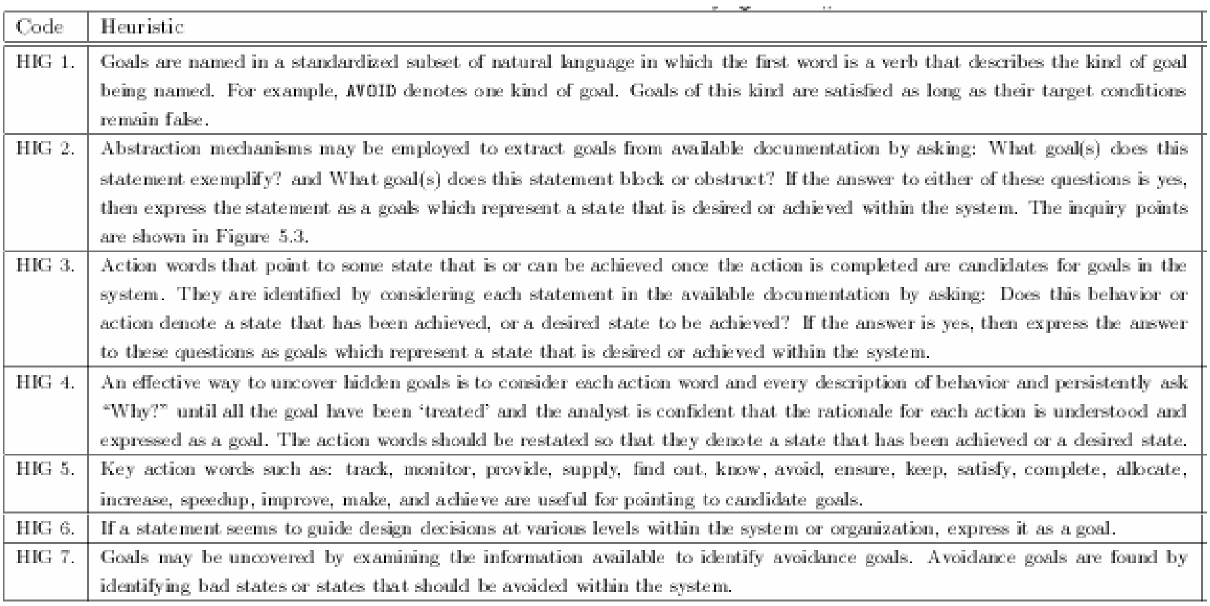
Analyzing the web site of IDC respect to our topic in this report (Application Development), we found that: navigability for to find information related with tools by this topics is not an easy task, because they categorize their research primarly in three main subjects: Business Areas, Consumer, and Enterprise, due to, many topics of research are lapping and is difficult distinguish the domain. In addition the focus of this taxonomy is more oriented to business, not tools in the market.

Any concept is documented or cleared (at least, with free access).

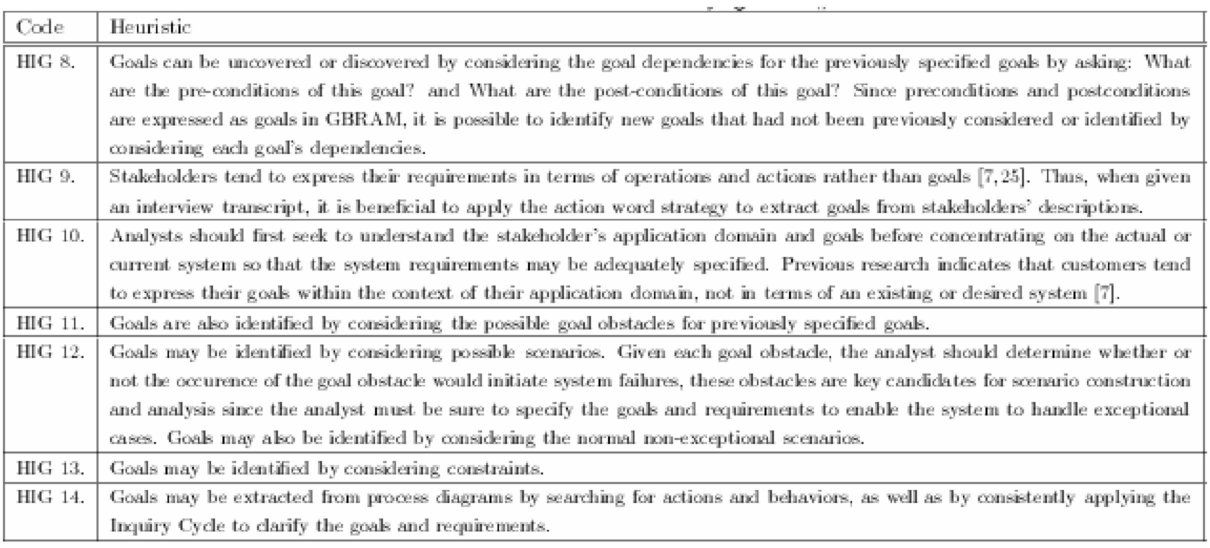
| **IDC** | |
|---|---|
| **Applications and Application Development** | |
| **L1** | **Subcathegory** |
| 1 | Application Design Tools (870) |
| 2 | Application Life-Cycle Management |
| 3 | Application Outsourcing and ASP Services |
| 4 | Application Strategies |
| 5 | Collaborative Computing |
| 6 | Data Warehousing, Information & Data Management Software |
| 7 | Document and Content Technologies |
| 8 | eComerce Applications |
| 9 | eCommunity Software Strategies |
| 10 | Enterprise Integration Software |
| 11 | Speech and Natural Language Software |
| 12 | Vertical Industry Applications |

# Appendix D

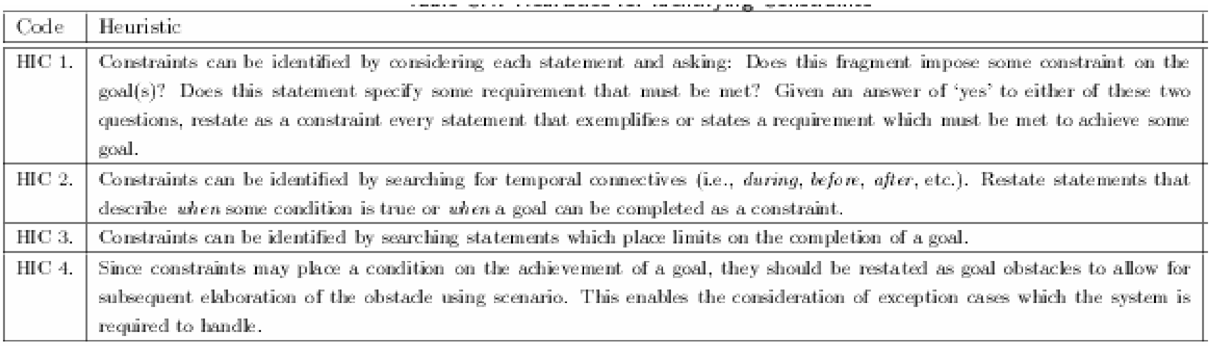
## Summary of Suitable GBRAM Heuristics



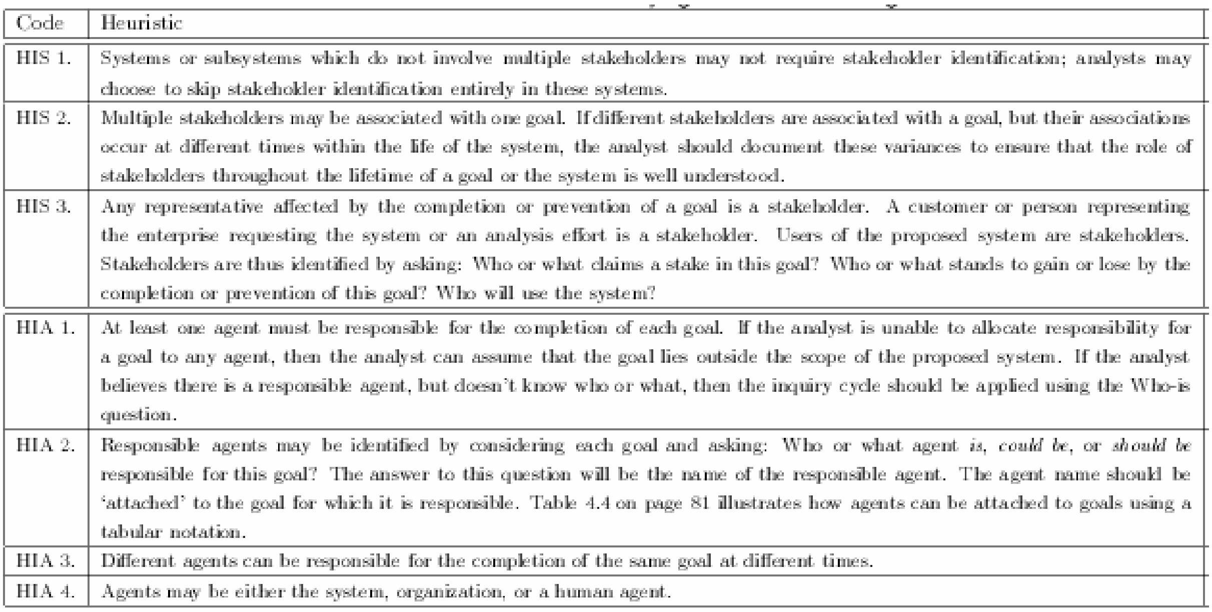
| Code | Heuristic |
|---|---|
| HIG 1. | Goals are named in a standardized subset of natural language in which the first word is a verb that describes the kind of goal being named. For example, AVOID denotes one kind of goal. Goals of this kind are satisfied as long as their target conditions remain false. |
| HIG 2. | Abstraction mechanisms may be employed to extract goals from available documentation by asking: What goal(s) does this statement exemplify? and What goal(s) does this statement block or obstruct? If the answer to either of these questions is yes, then express the statement as a goals which represent a state that is desired or achieved within the system. The inquiry points are shown in Figure 5.3. |
| HIG 3. | Action words that point to some state that is or can be achieved once the action is completed are candidates for goals in the system. They are identified by considering each statement in the available documentation by asking: Does this behavior or action denote a state that has been achieved, or a desired state to be achieved? If the answer is yes, then express the answer to these questions as goals which represent a state that is desired or achieved within the system. |
| HIG 4. | An effective way to uncover hidden goals is to consider each action word and every description of behavior and persistently ask "Why?" until all the goal have been 'treated' and the analyst is confident that the rationale for each action is understood and expressed as a goal. The action words should be restated so that they denote a state that has been achieved or a desired state. |
| HIG 5. | Key action words such as: track, monitor, provide, supply, find out, know, avoid, ensure, keep, satisfy, complete, allocate, increase, speedup, improve, make, and achieve are useful for pointing to candidate goals. |
| HIG 6. | If a statement seems to guide design decisions at various levels within the system or organization, express it as a goal. |
| HIG 7. | Goals may be uncovered by examining the information available to identify avoidance goals. Avoidance goals are found by identifying bad states or states that should be avoided within the system. |



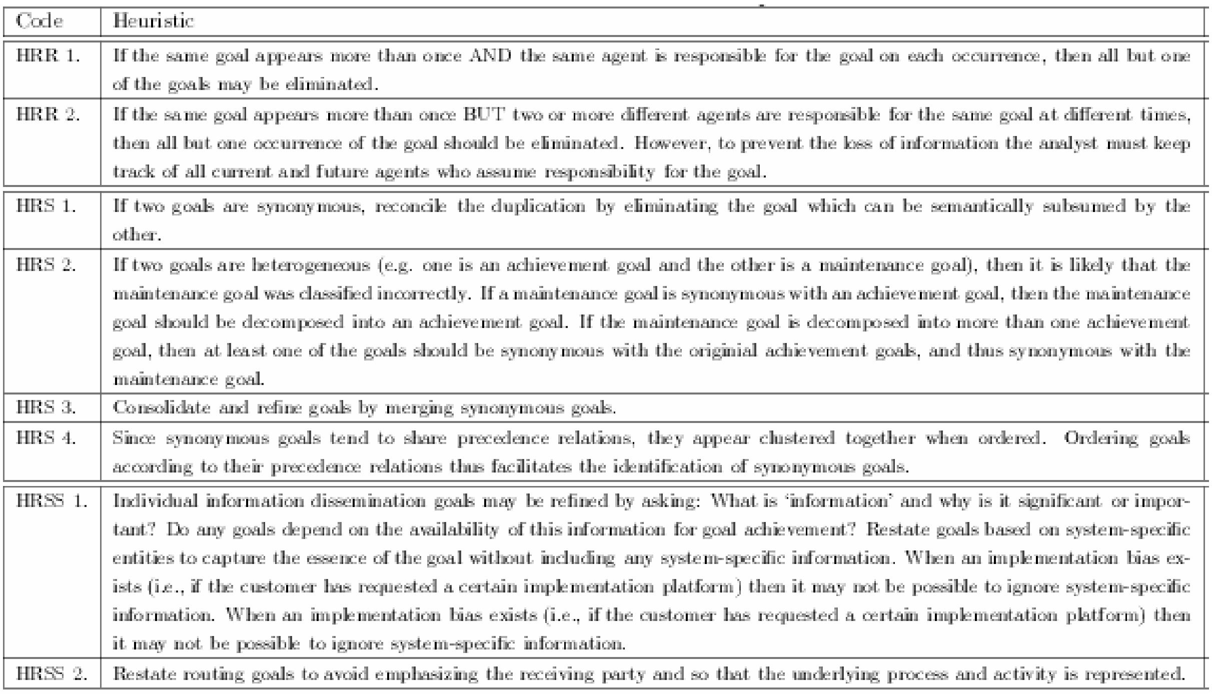
| Code | Heuristic |
|---|---|
| HIG 8. | Goals can be uncovered or discovered by considering the goal dependencies for the previously specified goals by asking: What are the pre-conditions of this goal? and What are the post-conditions of this goal? Since preconditions and postconditions are expressed as goals in GBRAM, it is possible to identify new goals that had not been previously considered or identified by considering each goal's dependencies. |
| HIG 9. | Stakeholders tend to express their requirements in terms of operations and actions rather than goals [7,25]. Thus, when given an interview transcript, it is beneficial to apply the action word strategy to extract goals from stakeholders' descriptions. |
| HIG 10. | Analysts should first seek to understand the stakeholder's application domain and goals before concentrating on the actual or current system so that the system requirements may be adequately specified. Previous research indicates that customers tend to express their goals within the context of their application domain, not in terms of an existing or desired system [7]. |
| HIG 11. | Goals are also identified by considering the possible goal obstacles for previously specified goals. |
| HIG 12. | Goals may be identified by considering possible scenarios. Given each goal obstacle, the analyst should determine whether or not the occurence of the goal obstacle would initiate system failures, these obstacles are key candidates for scenario construction and analysis since the analyst must be sure to specify the goals and requirements to enable the system to handle exceptional cases. Goals may also be identified by considering the normal non-exceptional scenarios. |
| HIG 13. | Goals may be identified by considering constraints. |
| HIG 14. | Goals may be extracted from process diagrams by searching for actions and behaviors, as well as by consistently applying the Inquiry Cycle to clarify the goals and requirements. |



| Code | Heuristic |
|---|---|
| HIC 1. | Constraints can be identified by considering each statement and asking: Does this fragment impose some constraint on the goal(s)? Does this statement specify some requirement that must be met? Given an answer of 'yes' to either of these two questions, restate as a constraint every statement that exemplifies or states a requirement which must be met to achieve some goal. |
| HIC 2. | Constraints can be identified by searching for temporal connectives (i.e., *during*, *before*, *after*, etc.). Restate statements that describe *when* some condition is true or *when* a goal can be completed as a constraint. |
| HIC 3. | Constraints can be identified by searching statements which place limits on the completion of a goal. |
| HIC 4. | Since constraints may place a condition on the achievement of a goal, they should be restated as goal obstacles to allow for subsequent elaboration of the obstacle using scenario. This enables the consideration of exception cases which the system is required to handle. |



| Code | Heuristic |
|---|---|
| HIS 1. | Systems or subsystems which do not involve multiple stakeholders may not require stakeholder identification; analysts may choose to skip stakeholder identification entirely in these systems. |
| HIS 2. | Multiple stakeholders may be associated with one goal. If different stakeholders are associated with a goal, but their associations occur at different times within the life of the system, the analyst should document these variances to ensure that the role of stakeholders throughout the lifetime of a goal or the system is well understood. |
| HIS 3. | Any representative affected by the completion or prevention of a goal is a stakeholder. A customer or person representing the enterprise requesting the system or an analysis effort is a stakeholder. Users of the proposed system are stakeholders. Stakeholders are thus identified by asking: Who or what claims a stake in this goal? Who or what stands to gain or lose by the completion or prevention of this goal? Who will use the system? |
| HIA 1. | At least one agent must be responsible for the completion of each goal. If the analyst is unable to allocate responsibility for a goal to any agent, then the analyst can assume that the goal lies outside the scope of the proposed system. If the analyst believes there is a responsible agent, but doesn't know who or what, then the inquiry cycle should be applied using the Who-is question. |
| HIA 2. | Responsible agents may be identified by considering each goal and asking: Who or what agent *is*, *could be*, or *should be* responsible for this goal? The answer to this question will be the name of the responsible agent. The agent name should be 'attached' to the goal for which it is responsible. Table 4.4 on page 81 illustrates how agents can be attached to goals using a tabular notation. |
| HIA 3. | Different agents can be responsible for the completion of the same goal at different times. |
| HIA 4. | Agents may be either the system, organization, or a human agent. |



| Code | Heuristic |
|---|---|
| HRR 1. | If the same goal appears more than once AND the same agent is responsible for the goal on each occurrence, then all but one of the goals may be eliminated. |
| HRR 2. | If the same goal appears more than once BUT two or more different agents are responsible for the same goal at different times, then all but one occurrence of the goal should be eliminated. However, to prevent the loss of information the analyst must keep track of all current and future agents who assume responsibility for the goal. |
| HRS 1. | If two goals are synonymous, reconcile the duplication by eliminating the goal which can be semantically subsumed by the other. |
| HRS 2. | If two goals are heterogeneous (e.g. one is an achievement goal and the other is a maintenance goal), then it is likely that the maintenance goal was classified incorrectly. If a maintenance goal is synonymous with an achievement goal, then the maintenance goal should be decomposed into an achievement goal. If the maintenance goal is decomposed into more than one achievement goal, then at least one of the goals should be synonymous with the originial achievement goals, and thus synonymous with the maintenance goal. |
| HRS 3. | Consolidate and refine goals by merging synonymous goals. |
| HRS 4. | Since synonymous goals tend to share precedence relations, they appear clustered together when ordered. Ordering goals according to their precedence relations thus facilitates the identification of synonymous goals. |
| HRSS 1. | Individual information dissemination goals may be refined by asking: What is 'information' and why is it significant or important? Do any goals depend on the availability of this information for goal achievement? Restate goals based on system-specific entities to capture the essence of the goal without including any system-specific information. When an implementation bias exists (i.e., if the customer has requested a certain implementation platform) then it may not be possible to ignore system-specific information. When an implementation bias exists (i.e., if the customer has requested a certain implementation platform) then it may not be possible to ignore system-specific information. |
| HRSS 2. | Restate routing goals to avoid emphasizing the receiving party and so that the underlying process and activity is represented. |

| Code | Heuristic |
|---|---|
| HED 1. | An effective way to discover precedence dependencies between goals is to consider each goal and ask: What goals are prerequisites for this goal? What goals must follow this goal? The answer to these questions indicate the given goal's precedence relations. They should be documented by the analyst so that the goal may be subsequently ordered according to these relationships. Chapter 4 on page 88 which analysts may employ for these annotations, or a tabular notation as shown in Table 5.3. |
| HED 2. | An effective way to identify contract dependencies is to consider each goal and ask: What goal(s) *must* be completed if this goal is achieved? |
| HED 3. | An effective way to identify agent dependencies between goals is to consider each goal and ask: What agent must complete the goal(s) they are responsible for before the agent responsible for this goal can achieve this goal? |
| HEO 1. | There is at least one goal obstacle for every goal. This is informally referred to as the trivial obstacle and formally referred to as the normal first case goal obstacles. These obstacles are worded by negating the verb in the goal name. |
| HEO 2. | A statement that illustrates a condition which prevents the completion of a goal or which illustrates an example of a goal being blocked by another goal is indicative of an obstacle and should be expressed as an obstacle. |
| HEO 3. | An effective way to identify goal obstacles is to consider each goal and ask: What other goal(s) or condition(s) does this goal depend on? What other goal(s) must be completed or achieved in order for this goal to be achieved? What goal(s) depend on this goal? What goal(s) must follow from this goal? Can the failure of another goal to complete cause this goal to be blocked? If this goal is blocked, what are the consequences? The answer to the questions above should be worded to emphasize the state that is true, thereby denoting a goal obstacle. |
| HEO 4. | A prerequisite failure obstacle occurs when a goal having a precedence relation is obstructed because the precedence goal fails. Prerequisite failures are identified by considering each goal and asking: What other goal(s) does this goal depend on? |
| HEO 5. | An agent failure obstacle occurs when a goal fails because the responsible agent fails to achieve the goal. Agent failures are identified by considering each goal and asking: Can the failure of an agent to fulfill their responsibilities cause this goal to fail? |
| HEO 6. | An contract failure obstacle occurs when a goal which holds a contract with another goal fails. Contract failure obstacles are identified by considering each goal and asking: Does this goal share a contractual relation with another goal? |

# References

1. The Gartner Group, available on-line at http://www4.gartner.com

2. INCOSE. "Software Engineering Tools Taxonomy". http://www.incose.org/tools/tooltax/se_tools_taxonomy.html

3. J.-C. Mielnik, B.Lang, S.Lauriére, J.-G. Schlosser, V. Bouthors. "eCots Plattaform: An Inter-industrial Initiative for COTS-Related Information Sharing". In Proceedings of 2nd International Conference on COTS-Based Software Systems (ICCBSS), LNCS 2580, 2003.

4. J.P. Carvallo, X. Franch, C. Quer, M. Torchiano. "Characterization of a Taxonomy for Business Applications and the Relationships Among Them". Proceedings 3rd International Conference on COTS-Based Software Systems, ICCBSS´04, Redondo Beach (California).

5. A. I. Anton. "Goal-Based Requirement Analysis". IEEE Proceedings of International Requirements Engineering Conference, 1996.

6. E. Yu. "Modelling Strategic Relationships for Process Reengineering". Ph.D. thesis, also Tech. Report DBKS-TR-94-6, Department of Computer Science,University of Toronto,1995.

7. X. Franch, J.P. Carvallo. "Using Quality Models in Software Package Selection". IEEE Software, January/February 2003.

8. J.P. Carvallo, X. Franch, C. Quer. "Defining a Quality Model for Mail Servers". 2nd International Conference on COTS-Based Software Systems (ICCBSS), Ottawa, Canadá, 2003.

9. Pere Botella, X. Burgués, Juan P. Carvallo, X. Franch, C. Quer. "Using Quality Models for Assessing COTS Selection". Proceedings V Workshop on Requirements Engineering (WER'02), Valencia, Spain. November 2002.

10. A. Vam Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto, August 2001. Pp 249-263.

11. A. I. Antón. "Goal Identification and Refinement in the Specification of Software-Based Information Systems". Ph.D. thesis, Georgia Institute of Technology, June 1997.

12. International Data Corporation http://www.idc.com

13. ComponenSource http://www.componentsource.com

14. http://searchvb.techtarget.com

15. Guide to the Software Engineering Body of Knowledge, SWEBOK, www.swebok..org

16. Roger S. Pressman. "Software Engineering: A Practitioner´s Approach" (European Adaptation, 5th Edition). McGraw-Hill.. ISBN:0077096771.

17. CBSE Net. Component Based Software Engineering Network. http://www.cbsnet.org

18. ISO/IEC 12207-1995

19. Infrastructure Software Market Definitions for Application Development. Gartner, Dataquest Guide. 4th June, 2003. http://www.gartner.com

20. Software Market Research Methodology and Definitions 2003-2004. Gartner Dataquest Guide. 16th January, 2004. http://www.gartner.com

21. Information of products and vendors. http://

22. http://www.rational.com

23. IEEE/EIA 12207.1-1997

24. IEEE Std 830-1998.

25. CBSE Net. "Application Domain Taxonomy ". Available on-line (previous registration) at: http://www.cbsenet.org/pls/CBSEnet/eco_ricerca_documenti.concept_search_frame.