Capturing and Structuring Goals: Analysis Patterns

Elizabeth A. Kendall
kendall@rmit.edu.au, kendall@info.bt.co.uk
Intelligent Business Systems Research, British Telecom Laboratories
Martlesham Heath, Ipswich, IP5 3RE ENGLAND

Uma Palanivelan, Satya Kalikivayi
Royal Melbourne Institute of Technology
Department of Computer Systems Engineering, Melbourne, Australia

ABSTRACT

Too often, analysis concentrates on *how* a system or business functions, through scenario descriptions and interaction models. In these four related patterns, analysis centers on objectives, with emphasis on *what* a business or system is trying to achieve. The purpose of each activity is captured and explicitly stated, hierarchical and other relationships between objectives are identified, and a goal hierarchy diagram is formed. This is utilized as the structural foundation for all requirements analysis. Due to the nature of goals, it is less variable than a structure based on tasks and activities.

INTRODUCTION

The patterns described here were mined from two sources: Alistair Cockburn's work on structuring use cases with goals [3, 4] and Anand Rao's Goal Event Method [16] (GEM) for modeling business processes and agent systems. This paper is a reformulation of their work in a pattern format.

These patterns are relevant to business process modeling and software requirements analysis. As stated in [3, 4], goals are a normal aspect of our analysis and software engineering work. The four patterns presented here represent steps toward a pattern language for capturing, structuring, and tracking a system's goals and objectives.

The four closely related patterns are Capturing Goals, Goal Cases, Structuring Goal Cases, and Goal Cases as Objects. The patterns are best employed in concert. The relationships between them are that the resulting context for one is the input or initial context for the next, as shown below in Figure 1.

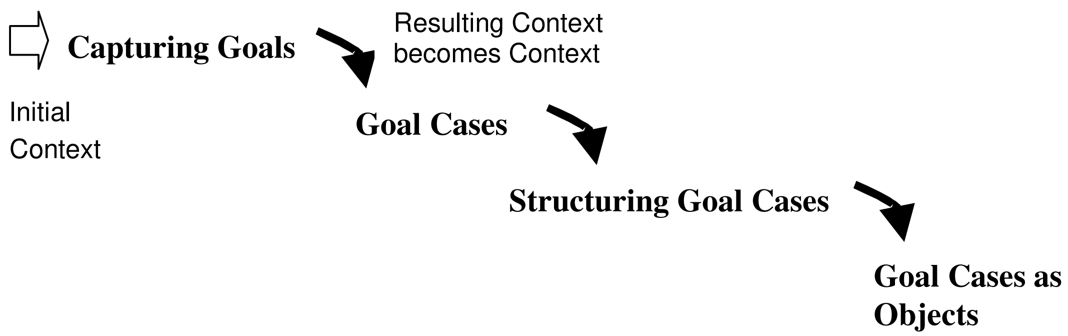

FIGURE 1: PATTERN MAP

As the patterns are closely related, the overall resulting context and known uses are discussed after all four patterns have been introduced. Additionally, all of the patterns share the following initial context, which is illustrated by an example.

CONTEXT

You are building a model of an organization or a system for eventual reengineering or development work. Your requirements are extensive and fairly informal, consisting of practices and activities, user stories, and detailed scenarios. Although you have many tools and techniques for analysis, it is difficult to know what is important and to determine what can provide the underpinning for your model. You want changes, which are bound to occur, to have as small an impact as possible. There is also a need for the model to be structured and modular, so it is easy to find things, to add new details, and to remove unwanted items. A multilevel model is also necessary, so details can be hidden during abstraction but are still readily accessible for review.

EXAMPLE

The example involves part of a system to detect and notify a system administrator of computer system intrusion. The small part of the system described here is responsible for dealing with host violations, in particular login violations and system file intrusions. Requirements are informally stated in a sequence of interactions.

REQUIREMENTS

- The system is responsible for dealing with host violations, in particular login violations and system file intrusions. The system administrator is notified of suspected or attempted intrusions.
- It is necessary to validate the date, time and existence of system files periodically, every few minutes. When a file is not found or a new version appears, the system administrator needs to be notified. When a user tries to modify or delete a system file, the system administrator needs to be notified.
- A user tries to login when he or she does not have a valid account. If this occurs once or twice in a short period of time, it is not a violation. Three or more attempts is a violation that needs to be reported.
- The system administrator may not be available to receive a notification. This can be due to a network failure or the fact that the administrator is performing another task. The report needs to be stored and resent after a delay.

PATTERN 1: CAPTURING GOALS

Problem

How can you capture and emphasize what is important in an extensive set of business or system requirements? How can you identify what is not likely to change so you can use it as the foundation for your model?

Context

You are trying to determine the essence of a detailed set of requirements that are of varying levels of detail and fairly informal, including user stories, processes, and activities that have many variations. You also need to identify a basis or foundation for your requirements model. Usually this occurs when you are just beginning to build an analysis model, but it may occur as you iteratively perform requirements analysis at varying levels of detail.

Forces

- Requirements involve a significant amount of detail.
- It is easy to be overwhelmed by the quantity of information.
- Many requirements are stated in task oriented scenarios, with a focus on activities and detailed steps.
- The purpose of each activity is not explicitly defined and objectives are obscured.
- The tasks and activities employed by a business or system are variable. New alternatives are uncovered, and approaches that do not really work or are not needed any more are discarded.
- Goals are less likely to change as they reflect aims and objectives, which are more stable.
- Scenarios and requirements have goals or objectives, and these should be captured.

Solution

A requirements model should concentrate on the value added work, which is the goal or objective, at any point [16]. In analysis, goals should be distinguished for each set of scenarios [3, 4]. You should base your model on *what* is to be achieved rather than *how* activities are to be carried out.

Capturing Goals begins by extracting scenarios from the requirements specification, user stories, or any available source. Goal statements for each scenario are ascertained by posing the question, "What is the objective of this scenario?"

Example

The goals are identified, by determining the purpose for each scenario.

GOALS

1. To detect and notify system administration of host violations.
2. To detect and notify system administration of system file violations.
3. To detect and notify system administration of login violations.
4. To ensure that system administration receives notification of a violation.

Resulting Context

The goals and objectives for each scenario have been captured and explicitly stated. These are less likely to change than the detailed steps and activities involved, and they can provide a foundation for the analysis model.

Related Patterns

Core Processes [1] abstracts an organization's problems and goals from the business scenarios by identifying the processes that are critical to the organization and its customers. Vision Statement [1] documents an organization's goals. These patterns are for business process modeling and do not address software system development.

The Sponsor Objectives [17] pattern can be used to capture and record a business' objectives and goals. This pattern is employed to get business objectives and system objectives in line, so that the system requirements can really satisfy a customer's needs. This pattern emphasizes meetings and any discrepancies in objectives are recorded. This pattern is complementary to the Capturing Goals pattern. It should be used when it is particularly difficult to determine goals and when customers are providing multiple and conflicting objectives.

PATTERN 2: GOAL CASES

Problem

Once you have identified goals and objectives, how can you keep track of their source?

Context

You have goals and scenarios, and you want to record and track the relationships between them. As a result of applying the Capturing Goals pattern, scenarios and goals are explicitly stated, but separate. This will make it difficult to trace the origin of a goal if it is questioned, or to add detail.

Forces

- All requirements, including goals and objectives, must be traceable to provide documentation and to address change.
- The source of a goal must be traceable to the relevant user stories or scenarios.
- Relationships between scenarios and goals must be preserved.

Solution

Each use case must be related to a particular goal. To emphasize this perspective, a use case and its related goal is termed a *goal case,* and they are stored together.

Goal + Use Case $\Longrightarrow$ Goal Case

Example

Each scenario and its related goal is linked together and termed a goal case. This can be done via hypertext in a CASE tool; it is done explicitly below for the example.

GOAL CASES

1. ***To detect and notify system administration of host violations.*** The system is responsible for dealing with host violations, in particular login violations and system file intrusions. The system administrator is notified of suspected intrusions or attempted intrusions.
2. ***To detect and notify system administration of system file violations.*** It is necessary to validate the date, time and existence of system files periodically, every few minutes. When a file is not found or a new version appears, the system administrator needs to be notified. When a user tries to modify or delete a system file, the system administrator needs to be notified.
3. ***To detect and notify system administration of login violations.*** A user tries to login when he or she does not have a valid account. If this occurs once or twice in a short period of time, it is not a violation. Three or more attempts is a violation that needs to be reported.
4. ***To ensure that system administration receives notification of a violation.*** The system administrator may not be available to receive a notification. This can be due to a network failure or the fact that the administrator is performing another task. The report needs to be stored and resent after a delay.

Resulting Context

You have preserved the source of each goal by storing goals and their related scenarios in goal cases.

Related Patterns

Scenarios Define Problem [5], and Scenarios Define Business Processes [1] document the requirements of a business process as a collection of business scenarios. Goals Cases explicitly states a goal for every scenario, directly building upon use cases [7, 8].

PATTERN 3: STRUCTURING GOAL CASES

Problem

There are different types of scenarios and goals, and there are hierarchical and other relationships between them. How can you capture these distinctions and preserve these relationships?

Context

You’re able to determine scenarios and their related goals, but the scenarios are of different sizes and at varying levels of detail. You aren't sure when to start and when to stop a given scenario. Scenarios and goals are of varying levels of importance. Some scenarios are central and always occur, while others are conditional exceptions that seldom occur. You want to be able to preserve these distinctions and relationships.

Forces

- It is difficult to determine when to start and when to end a given scenario.
- Goal cases are of varying size and importance.
- Scenarios may relate to the main discourse or sequence of interaction. They may also pertain to alternatives and extensions.
- Levels of detail in scenarios and goals must be preserved.

Solution

Scenarios and related goals should be structured so the main sequence of interaction, variations, conditional exceptions, and subordinate details can be distinguished from one another [3,4]. Structure your goal cases so that all interactions in a given goal case pertain to the same goal. The scenarios within a given case should continue until the goal is achieved or abandoned [3, 4].

At all levels, alternative and extension cases should be considered separately from the main discourse or main way the system or business operates. Establish placeholders for detail that is expected to appear later; these can be termed *variations*. When subordinate details are available indicate their level. Further, use cases and scenarios are recursive. That is, a use case can be (and often is) a collection of use cases, and a scenario is often comprised of a set of scenarios. It is just a question of the level of detail or abstraction.

Example

The goal cases are now structured into a main discourse, subordinates, and extensions. Subordinates within the main discourse can be indicated with a hierarchical outline format; extensions are marked with a letter suffix.

STRUCTURED GOAL CASES

1. ***To detect and notify system administration of host violations.*** The system is responsible for dealing with host violations.
1.1 ***To detect and notify system administration of system file violations.*** The system is responsible for dealing with system file intrusions.
1.1.1 ***To determine if system files have been deleted or modified.*** It is necessary to validate the date, time and existence of system files periodically, every few minutes. When a file is not found or a new version appears, this is a violation.
1.1.2 ***To detect an attempt at system file violation.*** When a user tries to modify or delete a system file, this is a violation.
1.1.3 ***To notify system administration of system file violations.*** The system administration needs to be notified of system file violations.
1.1.3a. ***To ensure that system administration receives notification of a violation.*** The system administrator may not be available to receive a notification. This can be due to a network failure or the fact that the administrator is performing another task. The report needs to be stored and resent after a delay.

1.2 ***To detect and notify system administration of login violations.*** The system is responsible for dealing with login violations.
1.2.1 ***To determine if an invalid user tries to login.*** A user tries to login when he or she does not have a valid account. If this occurs once or twice in a short period of time, it is not a violation. Three or more attempts is a violation.
1.2.2 ***To notify system administration of login violations.*** The system administration needs to be notified of login violations.
1.2.2a. ***To ensure that system administration receives notification of a violation.*** The system administrator may not be available to receive a notification. This can be due to a network failure or the fact that the administrator is performing another task. The report needs to be stored and resent after a delay.

The outline can also be depicted as a tree, as is shown below. In the diagram, subordinate relationships are shown with a solid arrow, and conditional extensions are shown with a dashed arrow. The arrows point from the subordinate or extension to the goal case of a higher level.

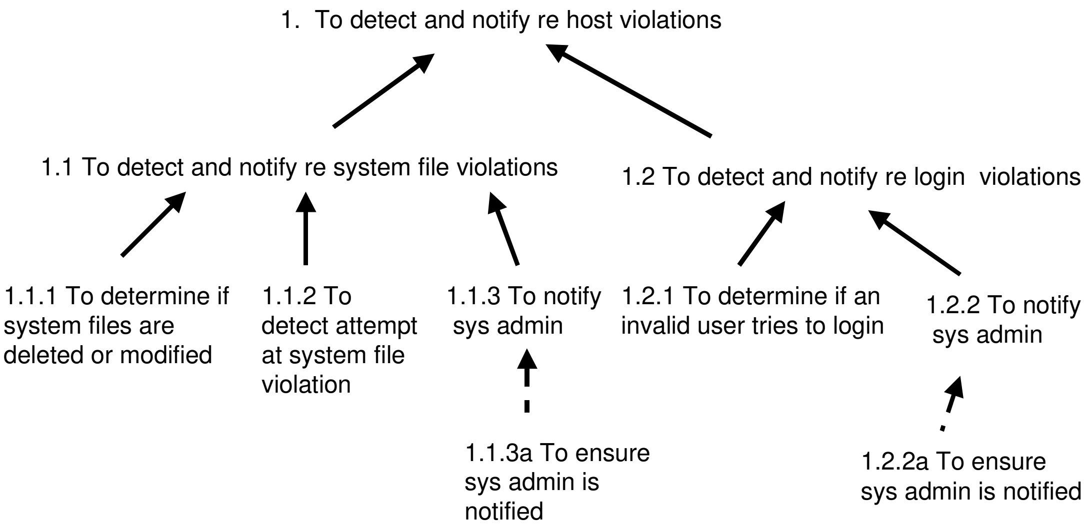

FIGURE 2: STRUCTURED GOAL CASES IN A TREE DIAGRAM

Resulting Context

The goal cases are all of similar size, with a clear beginning and a clear end. Subordinates and conditional extensions are readily recognizable. All hierarchical and conditional relationships between goal cases are preserved.

Related Patterns

Structuring Goal Cases clarifies the use case approach [7, 8].

PATTERN 4: GOAL CASES AS OBJECTS

Problem

How can you avoid redundancy and duplication in a goal case analysis model?

Context

The same goal case appears in more than one place in your model. This may happen frequently.

Forces

- Orthogonal goals, when arranged in a hierarchy, easily lead to redundant subgoals.
- Redundant goal cases will be hard to maintain.
- Duplicate goals and actions will bloat the analysis model.

Solution

Treat each goal case as an object. Promote redundant goals and actions to a high level object, and utilize inheritance to bring the common factors into subordinate objects. Additional goals and actions can be added to subordinates, as needed. Further, any identical goal cases can be viewed to be instances of the same class.

Example

Goal cases 1.1.3 and 1.2.2 are duplicates of the same goal "Notify System Administrator." The conditional extensions of 1.1.3a and 1.2.2a are also the same. These can be treated as instances of the same class, and hierarchical relationships can be represented with inheritance.

This leads to the goal case hierarchy diagram in Figure 3

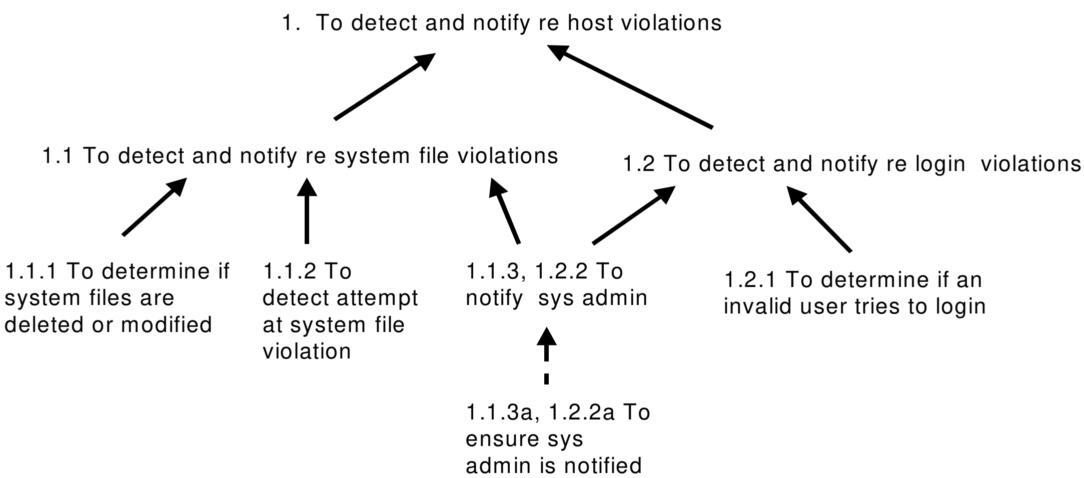

FIGURE 3: GOAL CASE HIERARCHY DIAGRAM

Resulting Context

Redundant goal cases have been removed.

Related Patterns

Goal Cases As Objects is taken directly from use case analysis.

OVERALL RESULTING CONTEXT

Advantages

- The solution obtained by applying all four of the patterns captures the goals of a system or business; this leads to a goal driven approach rather than one based on tasks and activities.
- The goal hierarchy diagram can be utilized as the foundation for all requirements analysis and modeling. All structure and relationships are preserved, but redundancy is avoided. Due to the nature of goals, it is less variable than a structure based on tasks and activities.
- Agent based systems feature proactive and autonomous behavior, where agents act to achieve goals without substantial human intervention [9]. Capturing and structuring goals is a good starting point for agent based system analysis [10].
- A goal centric approach to analysis leaves all options open for design and implementation. If an analyst focuses on interactions and processes within the system, they may prematurely limit their options. For example, a particular model of interactions and activities may preclude the use of mobile objects.

Potential Pitfalls

- Scenario explosion is a concern and can be addressed by employing subordinate use cases, extensions and variations [3, 4]. Variations are stated within a high level use case, as a place holder for subsequent detail.
- Goals may be abstract and not concrete, but, when scenarios, tasks and activities are attached in the goal cases, they can be made more specific.
- As in any requirements analysis, the scope of the system or business needs to be clearly defined.
- It may be difficult to ascertain a system's goals, and to get consensus regarding them. This is when the Sponsor Objectives [17] pattern should be employed.

Postconditions: Remaining Problems

- An application may result in a large number of goals. It may be useful to differentiate between commonly occurring types of goals, such as those involving recovery, alternative paths, strategic goals, summary goals, user goals, and sub-functions.
- Some goals may conflict with each other. This relationship should be captured and additional patterns are needed to address this consideration.

- The goal hierarchy diagram needs to be partitioned so goals can be assigned to objects or roles. This involves responsibility based design [18], with goals mapping to responsibilities. Patterns are needed for partitioning.

KNOWN USES

Structuring use cases with goals was proposed by Alistair Cockburn [3, 4], and these references are the major source of all four of the patterns presented here. Alistair cites many projects that have been addressed with this approach, including small, medium sized, and relatively large (50 work years, 200 use case) projects. Alistair describes the techniques as easy to use and scaleable. He states that the approach provides benefits outside of requirements gathering, including project management and project tracking.

The Goal Event Method (GEM) for business process modeling and system specification was developed by Anand Rao [16]. It has been widely used at the Australian Artificial Intelligence Institute for agent based system development [11]. GEM utilizes Capturing Goals, Goal Cases, and Structuring Goal Cases; it does not utilize Goal Cases as Objects. In GEM, modeling supported by a semi-automated tool.

M. Ould [13] developed the STRIM method for business process modeling and analysis; this method emphasizes goals and is widely used at Praxis [16]. Recent work in agent based system engineering at BT Research Laboratories is centering on a goal based approach [9, 10].

One of the basic questions that is asked during the development of a computer system and its user interface is "*What will the users want to do with the system?*" A task analysis of peoples' current activities is not sufficient to predict what they will do in the future. The ARPANET, for example, led to widespread use of electronic mail even though it had been primarily intended for remote logins and the sharing of computers and databases.

Nielsen [12] addresses this problem by considering goals that a user may have when using a system. First, primary user goals are identified; additional system features are then found through generalization and integration. Generalization involves supergoaling and reuse, which are known uses of Structuring Goal Cases and Goal Cases as Objects, respectively.

Functional modeling is an approach used to model a system by identifying the overall goal it must achieve and the functions it must perform. The approach tries to identify the overall goal of a system by posing the question *"Why was the system designed in the first place?"* Goal Tree- Success Tree Master Plant Logic Diagram (GTST_MPLD) is a method based on functional modeling which is used to build a computer based human support system to monitor complex manufacturing plants [14]. This approach models a complex system in five different types of hierarchies: functional, structural, goal, event and behavioral.

In GTST_MPLD, the goal tree for a system is constructed by decomposing the overall goal of the system into a set of necessary and sufficient sub-goals, continuing until physical components are needed to satisfy a sub-goal. This represents known uses of Capturing Goals and Structuring Goals.

According to Sean Chen [2] "web designers try to create a site which enables users to stay longer, search deeper, revisit more frequently and generate clearer navigation patterns. Web designers must understand how users navigate the Cyberspace and how their seamless interactivity occurs." A user's navigational behavior on the Internet is tracked with a goal-centric approach in [2]. Chen explains that a user navigates in two ways: goal-directed and experiential. Chen captures the user's activities that are goal directed and models them as a goal hierarchy. His goal hierarchy involves choices among products or services, information sources and navigational alternatives.

The ACT-R model of submariner situation assessment [6] aims to detect a hostile submarine and determine various parameters, such as range, course and speed. The system starts with a set of scenarios and identifies the goal of each scenario (Capturing Goals and Goal Cases). This is then modeled in the form of a goal hierarchy (Structuring Goals), with common goals shared between different sub-trees (Goals Cases as Objects). The goal hierarchy is then used to identify a suitable tactical approach that can be used in hostile situations.

ACKNOWLEDGEMENTS

The authors thank Alistair Cockburn and our EuroPlop'98 shepherd, Andreas Rueping, for their expert guidance.

REFERENCES:

1. Beedle, M., "cOOherentBPR – A Pattern Language to Build Agile Organizations," PLOP'97.
2. Chen, S., "Exploring Customers' Network Behavior: A study of Flow on the Web Navigation", http://chen.syr.edu/inter3.html, August, 1997.
3. Cockburn, A., "Structuring Use Cases with Goals: Part 1," Journal of Object Oriented Programming, Sept - Oct 1997, pp. 35- 40.
4. Cockburn, A., "Structuring Use Cases with Goals: Part 2," Journal of Object Oriented Programming, Nov - Dec 1997, pp. 56 - 62.
5. Coplien, J., "A Generative Development-Process Pattern Language", in J.O. Coplien, D.C. Schmidt (Eds.), Pattern Language of Program Design, Addison- Wesley, 1995.
6. Ehret, B.D., Gray, W.D., Hutcheson, T.D., "ACT-R Models of Submariner: Situation Assessment", http://act.psy.cmu.edu/ACT/ftp/workshop/Workshop-97/Ehret/base.000.html, August 2, 1997.
7. Jacobson, I., et al., *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, Rading, MA, 1992.
8. Jacobson, I., Ericsson, M., Jacobson, A., *The Object Advantage*, Addison- Wesley, New York, 1995.
9. Kendall, E. A., C. V. Pathak, P.V. Murali Krishna, C. B. Suresh, "The Layered Agent Pattern Language," PLOP'97.

10. Kendall, E. A., U. Palanivelan, S. Kalikivayi, N. Panvalkar, “Agent Analysis and Design with Roles and Role Modeling," work in progress, British Telecom Research Laboratories.

11. Kinny, D, Georgeff, M., Rao, A.,”A Methodology and Modelling Technique for Systems of BDI,” In MAAMAW’96, LNAI 1038, pages 56-71, The Netherlands, 1996.

12. Nielsen, J., “Goal Composition: Extending Task Analysis to Predict Things People May Want to Do”, http://www.useit.com/papers/goalcomposition.html, 1994.

13. Ould, M., *Business Processes: Modelling and Analysis for Reengineering and Improvement*, John Wiley & Sons, West Sussex, England, 1995.

14. Oulіddren, K., "An Approach to Incremental Fuzzy Modelling of Dependencies in Complex Systems," http://hades.sckcen.be/dept/lhma/kohpage.htm,1996.

15. Praxis , Deloitte & Touche Consulting Group, “STRIM: A Unique Method for Business Process Modeling and Analysis,” http://www.praxis.co.uk, March 7, 1998.

16. Rao, A. S., “Modelling the Service Assurance Process for Optus Using GEM: A Case Study,” Australian Artificial Intelligence Institute Technical Note 69, www.aaii.oz.au/research/techreports, March, 1998.

17. Whitenack, B., “RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development”, in J.O. Coplien, D.C. Schmidt (Eds.), Pattern Language of Program Design, Addison- Wesley, 1995.

18. Wirfs- Brock, R., B. Wilkerson, L. Wiener, *Designing Object Oriented Software*, Prentice Hall, Englewood Cliffs, NJ, 1990.