A Pattern Language for Goals

Stephen Cook, Kenneth Boness, Marc Bartsch and Rachel Harrison
School of Systems Engineering
University of Reading
Reading, UK
Email: s.c.cook@reading.ac.uk

Paul Wernick
Department of Computer Science
University of Hertfordshire
Hatfield, UK
Email: p.d.wernick@herts.ac.uk

Abstract—Previous work has established the value of goal-oriented approaches to requirements engineering. Achieving clarity and agreement about stakeholders’ goals and assumptions is critical for building successful software systems and managing their subsequent evolution. In general, this decision-making process requires stakeholders to understand the implications of decisions outside the domains of their own expertise. Hence it is important to support goal negotiation and decision making with description languages that are both precise and expressive, yet easy to grasp. This paper presents work in progress to develop a pattern language for describing goal refinement graphs. The language has a simple graphical notation, which is supported by a prototype editor tool, and a symbolic notation based on modal logic.

I. INTRODUCTION

In this paper we describe work in progress to develop a pattern language for describing goal graphs. Our work builds on previous research into goal-oriented requirements engineering [15] and software evolution [19]. In particular, we propose a pattern language [1] for goals that has both a graphical and a symbolic notation. We show how this approach can be useful for clarifying the goals, both explicit and implied, of a software system. Although our immediate purpose is intellectual — we want to understand the original rationale for a legacy system that subsequently evolved — we envisage that our techniques could also be applied in industrial situations where it is important to discover stakeholders’ goals and assumptions for a complex system.

Section II explains the background to this work. In Section III we provide a brief summary of our related proposal for a goal-oriented flavour of modal logic. Section IV presents a case study of inferring goals from a project proposal document. Sections V and VI present a selection of goal patterns using our notations.

II. BACKGROUND

A. Goals and Evolution in Software Systems

The relationship between (a) the original articulation of a system’s goals and assumptions, and (b) the system’s subsequent evolution, is an interesting research question. The phenomenon of software evolution was first identified by Lehman [18]. His subsequent work includes proposals for eight ‘laws of software evolution’ [20] and the SPE [17] classification of evolving software systems. In recent joint work with Lehman, we proposed significant revisions and clarifications to SPE, which we refer to as SPE+ [6].

SPE+ defines three categories of evolving software systems:

- S-type systems evolve negligibly at most and are rare. An S-type system can only be found where the sole critical success factor for its stakeholders is satisfaction of a finalised specification. In effect, an S-type system defines its own universe.
- The evolution of a P-type system is constrained, because it is a critical success factor for its stakeholders that the system complies with an externally defined artifact, such as a framework, standard, or scientific theory [1], and shares its assumptions.
- Most software systems are E-type; they inevitably evolve for as long as they remain useful to their stakeholders. This phenomenon occurs because stakeholders’ goals for an E-type system, their assumptions about it, and the environmental changes that can affect the system’s usefulness, are ultimately unbounded.

Thus the concepts of stakeholder, goal and assumption are critical for understanding and classifying the propensity for evolution in software systems.

B. Requirements engineering and decision support

To increase the probability that an engineering project will have a successful outcome, stakeholders should try to find an agreed set of requirements that are acceptable, feasible and affordable [11]. The effectiveness of this decision-making process depends in part on the coordination of one or more kinds of knowledge, principally:

- stakeholders’ goals [15];
- relevant real-world domains, including existing systems;
- technological opportunities and costs [3];
- the requirements of the system-to-be.

The decision-making processes for goals and requirements often oblige stakeholders to make value judgements about an uncertain future. To improve the quality of those decisions,
stakeholders need to understand the implications of decisions beyond the domains of their own expertise. For example, both software architects and business managers may need to understand the implications of a system’s architecture and evolvability for the time-to-market of products and hence for the viability of a business.

It is particularly important to discover relevant assumptions and expectations about the application domain of the system-to-be, significant obstacles to achieving stakeholders’ goals, and any conflicts between those goals. Otherwise negotiations about requirements are likely to be unproductive. Thus, decision support in requirements engineering needs description languages that are capable of being both precise and expressive.

In terms of Stamper’s [26] six levels of semiotic analysis, stakeholders’ goals and the implications of decisions about goals have to be understood at the pragmatic and social levels of meaning. Pragmatics refers to the intent of a communicative act; the social level refers to meaning that depends on the social context of a communication. This is consistent with Potts’ [24] observation that descriptions of software systems and their purposes are pervasively figurative and metaphorical. For example, a system might be described in terms of agents with desires and preferences who inhabit a landscape, where the emphasised terms are to be understood as metaphorical interpretations of program behaviour.

Nevertheless, most experience with formal methods in software development indicates that they can be invaluable for checking correctness of reasoning, clarifying meaning, and revealing omissions. Thus, in terms of Stamper’s categories, it is also desirable that goal graphs can be analysed at the semantic and verified at the syntactic levels of meaning. An important aim for research into decision support in requirements engineering is to find reconciliations between these contrasted styles of description and levels of meaning.

C. Use of pattern languages in software engineering

The use of patterns and pattern languages in software engineering derives from Alexander’s work [1] on improving the quality of solutions to design problems in the built environment. A pattern is a reusable solution to a recurring problem in a context. A pattern language is a structured collection of patterns that work together. A pattern encapsulates more knowledge than a template because, for example, it has a rationale that explains how it resolves the conflicting forces that make a situation into a problem. On the other hand, a pattern is defined more abstractly than a bespoke solution to a specific problem.

The pattern concept has become widely used in software design [10], software architecture [5], domain analysis [9], etc. It can also be applied to requirements engineering. For example, Kendall et al. [13] and Hagge and Lappe [12] describe process patterns for conducting goal analysis in a disciplined way.

D. A pattern language for goal graphs

Our use of patterns has similarities with the scenario-based approach described by Ian Alexander [2] and with the formal refinement patterns proposed by Darimont [8]. When stakeholders try to articulate the problem that a system-to-be should solve, particular relationships between goals, obstacles and assumptions tend to recur [22]. The repeated features can be abstracted into a pattern for subsequent reuse.

To resolve the description language problem discussed in Section II-B, our pattern language allows patterns to be defined both graphically and symbolically. The topological relationships between pattern elements, e.g. a conflict between two goals, can be represented visually by simple box-and-line diagrams, illustrated in Sections [V] and [VI]. This style of visual language can help non-expert readers by implying, but not imposing, simple spatial metaphors [24] of proximity and hierarchy. The logical relationships between pattern elements, e.g. a conjunction of sub-goals entails a parent goal, can be represented symbolically using a flavour of modal logic. This is discussed in more detail in Section III and illustrated in Section [V].

III. USING MODAL LOGIC TO DESCRIBE GOALS

A goal graph describes part of a possible world that is similar to and also different from the current real world. The description is partial in the case of P- and E-type systems because a goal graph is finite but the real world has an uncountable number of properties. It is also partial because the real world is extremely complex and in general some simplification is unavoidable to produce a usable artifact.

In parallel with the work reported here, we are proposing a flavour of modal logic for reasoning about goals. Our proposal [7] can be briefly summarised in the following way. When a stakeholder constructs a goal graph, the stakeholder is considering at least three ‘model sets’ or possible worlds:

- the current real world \( w_0 \);
- a description \( G \), possibly incomplete, of an imagined world \( w_G \) where the stakeholder’s minimum set of goals is satisfied;
- an implicit imagined world \( w_F \) which shares the assumptions of \( w_G \) but where the stakeholder’s minimum set of goals is not satisfied.

So, for example, when stakeholders define the critical success factors for a project, they implicitly distinguish between the current situation, where these factors may not be applicable, an imagined scenario in \( w_G \) in which the success factors are satisfied, and an alternative scenario in somewhat different circumstances, i.e. in \( w_F \), in which one or more of the success factors are not satisfied.

The relationship between the real world and P- and E-type software systems is complex and various philosophical traditions have been used to explain it. For example, Lehman [17] describes the relationship in terms of models, in the mathematical sense. Sowa [25] provides an explanation in terms of ‘possible worlds’. In previous work [6] we showed that the relationship can also be explained using hermeneutics. Stamper [26] and Liu [21] use a semiotic perspective.
We can reason about propositions in the three worlds using the usual modal operator symbols □ and ◦. They can be interpreted in a goal-oriented modality 3 by reading □p as ‘p is assumed’, and ◦p as ‘p is conceivable’. To assist reasoning about the imagined worlds w_F and w_G, we introduce two further modal operators, □̃ and ◦̃, where □̃p is read as ‘p is expected’ and ◦̃p is read as ‘p is required’.

The goal-oriented modality is related to the symbols of Kripke’s modal logic through the following axioms:

- everything that is required is conceivable (anything inconceivable is not required):
  \[ \Box p \subseteq \Diamond p \] (1)
- everything that is assumed is expected, and everything that is expected is conceivable (anything inconceivable is not expected, and anything unexpected is not assumed):
  \[ \Box p \subseteq \Box p \cap \Diamond p \] (2)

Within the goal-oriented modality, the operators can be defined by equivalences to sets of propositions in the worlds w_0, w_F and w_G:

- proposition p is assumed if it is true in all three worlds:
  \[ \Box p \equiv p \in (w_0 \cap w_F \cap w_G) \] (3)
- proposition p is conceivable if it is true in any world:
  \[ \Diamond p \equiv p \in (w_0 \cup w_F \cup w_G) \] (4)
- proposition p is expected if it is true in all imagined worlds but need not be true in the current real world:
  \[ \Box p \equiv p \in w_F \cap w_G \] (5)
- proposition p is required if it is true when the goals are satisfied but need not be true in other worlds:
  \[ \Box p \equiv p \in w_G \] (6)

In Section IV we will show that goal patterns can be defined as theorems of this modality, and that this is helpful for clarifying ambiguous semantics in goal graphs.

IV. CASE STUDY

To illustrate our approach, we present some preliminary results from our case study of a project that we call System-A. The purpose of the project was to extend the lifespan of a legacy telecommunications system. The system’s operator, Company-B, wished to migrate its telecommunications services from analogue to digital technologies in a cost-effective way. Company-C prepared a ‘Project Proposal’ document describing their recommended solution for System-A. A short extract from this document is reproduced below, marked up with our own inferences about the implied goals and obstacles of the enhancement project. In Section V we use this material to illustrate various goal patterns.

We have also inferred that the document itself was intended to satisfy the following goals:

1) Demonstrate that Company-C understands Company-B’s goals with respect to System-A.
2) Convince Company-B to authorise Company-C to proceed with the enhancement project.
3) Reassure Company-C’s managers that the project has a sound business case.

We will return to this set of goals in Subsection VI-A and discuss its possible abstraction into a goal pattern.

A. Conventions used

A sans serif typeface is used for the text of the ‘Project Proposal’ document. Where a goal or obstacle has been inferred, the relevant text is set in an oblique typeface and enclosed in ‘floor’ brackets [ ] with a superscript reference to the lists of inferred goals and obstacles that follow. The use of markup techniques for locating inferences in system documents has also been discussed by Boness et al. [4] in the context of assessing whether requirements have been understood.

B. Text and inferred goals

1) Document text with goals markup:

1 TECHNICAL OVERVIEW

1.1 Introduction

The System-A Integrated Enhancement System draws heavily upon Company-C’s extensive experience [G1] in designing and supplying System-A exchanges. By applying high technology systems concepts [G2] we are able to supply a package of enhancements [G3] which can provide Company-B’s customers with almost all the benefits of modern digital exchanges [G4]. This can be achieved without the System-A exchanges [G5] and without disruption of service [G6].

The Enhancement System was designed with System-A’s current technology and current techniques in mind to provide for the following facilities:

- Common Channel Signalling [G6]
- Call Logging [G7]
- Supplementary Services [G8]

The result is a System which is easy to integrate into a working exchange [G9], modular in design [G10], providing adaptability [G11] to meet both future requirements [G12] and innovative [G13] in that it will provide compatibility to the digital network [G14].

2) Implicit goals: The following goals are not explicit in the text but can be inferred from it. Two root goals can be inferred using knowledge of organisations and business strategies:

- Company-B: achieve business objectives [G15]
- Company-C: achieve business objectives [G16]
The wording of goal [G4] can be understood using pragmatics as implying an additional goal to complete the refinement of its parent [G12]:

\[ \text{Provide outstanding benefits of digital exchanges} \]

3) Lists of inferred goals and obstacles:

G1: Leverage Company-C’s expertise
G2: Use high technology systems concepts
G3: Enhance existing exchanges
G4: Company-B’s customers get most benefits of digital exchanges
G5: Replace System-A exchanges
G6: Provide CCS
G7: Provide Call Logging
G8: Provide Supplementary Services
G9: Easy integration
G10: Modular design
G11: Adaptable system
G12: Meet expectations for future requirements
G13: Devise innovative solution
G14: Provide digital network compatibility
G15: Company-B: achieve business objectives
G16: Company-C: achieve business objectives
G17: Provide outstanding benefits of digital exchanges

O1: Service disruptions
O2: Demand exceeds capacity

4) Goal graph: The goal graph that we derived from the document text and our own inferences is shown in Figure 8. The diagram has been placed towards the end of the paper to illustrate some combinations of the individual patterns described in Section V.

C. Goal inferencing techniques

A full description of the techniques that can be used to infer goals from system artifacts is beyond the scope of this paper. In general, goal graphs cannot be constructed deterministically. Provided that a suitable language is used, goal graphs can be validated by stakeholders and some errors can be detected by model-checking, but it is difficult to eliminate some residual arbitrariness.

V. EXAMPLES OF PATTERNS

The example patterns described below illustrate both the graphical notation used in Figures 1-8 and a symbolic notation based on the goal-oriented modal logic outlined in Section III. Our graphical notation is similar to that used in the Objectiver tool, which is a consequence of common ancestry in Lamsweerde’s work on the KAOS [15] method for requirements engineering. However, compared to Objectiver, our VisualGoals tool uses a simplified palette, principally goal, obstacle, refinement, and task.

Space limitations prevent us from presenting full descriptions of our patterns using the template recommended by Alexander [1] and exemplified by Gamma’s catalogue [10] of OO design patterns. We have chosen to emphasise the communicative value of goal patterns, heuristics for finding them, and our notations for representing goal graphs.

A. Chain of Refinements pattern

Following Lamsweerde, we model how explanations as child goals, and why rationales as parent goals. For example, the sentence

‘By applying high technology systems concepts we are able to supply a package of enhancements which can provide Company-B’s customers with almost all the benefits of modern digital exchanges.’

yields the chain of child [G2], parent [G3] and grandparent [G4] goals shown in Figure 1. We call this pattern ‘Chain of Refinements’, by analogy with the OO design pattern Chain of Command [10].

![Chain of Refinements goal pattern](http://www.objectiver.com/)

The example in Figure 1 is an instance of the theorem:

\[ \Box p_1 \land \Box[(p_n \models p_{n-1}) \land (p_{n-1} \models \ldots \models p_1)] \]

\[ \Rightarrow \Box p_n \]

(7)

Theorem (7) can be read informally as:

In \( w_{G} \), if we require \( p_1 \) and we expect that \( p_n \) will entail \( p_{n-1} \) and so on in a chain culminating in \( p_1 \), then it follows that we require \( p_n \).

When a sub-goal refines a goal, it must be consistent with its parent but its definition can and usually does include additional information, e.g. a description of how its parent’s abstractions will be implemented. We capture this relationship using the semantic entailment relation, written as \( \models \). The entailment relationships are specified as expectations, using the \( \Box \) symbol, because they need not be true in the current world but are expected to become true in any world that is equivalent to \( w_{G} \).

B. Divide and Conquer pattern

Grammatical structures can imply goal patterns, e.g. lists of sub-goals imply the logical AND relationship between them. We call this pattern ‘Divide and Conquer’, see example in Figure 2. It can be expressed as the theorem:

\[ \Box p \land \Box[(q_1 \land q_2 \land \ldots \land q_n) \models p] \]

\[ \Rightarrow \Box(q_1 \land q_2 \land \ldots \land q_n) \]

(8)
This can be read informally as: if we require \( p \) and we expect that the conjunction of sub-goals \( q_1 \cdots q_n \) will entail \( p \), then it follows that we require \( q_1 \cdots q_n \).

Useful heuristics for detecting this pattern include:
- often associated with proactive goals, e.g. ‘achieve \( x \)’, ‘provide \( y \)’;
- the sub-goals are often temporally ordered [16].

This can be valued in agile approaches because architectural decisions may have to be revisited as a result of evolution in either a system’s requirements or its environment.

In contrast, the Strategy pattern that is familiar from Gamma’s work [10] is concerned with facilitating run-time choices by using a plug-in architecture. From a goals perspective, this pattern is about tactics rather than strategies. In the context of a goal graph it is more appropriately named A La Carte. It can be distinguished from Strategic Decision by its theorem:

\[
\Diamond p \land \Box \{ q_1, q_2, \ldots, q_n \} \models p \\
\Rightarrow \Diamond \{ q_1, q_2, \ldots, q_n \} \quad (10)
\]

Informally, this theorem means that to satisfy \( p \), we need to provide the set of possibilities \( q_1, q_2, \ldots, q_n \), whereas in Theorem 9 any one of the possible options is sufficient.

A useful heuristic for detecting the A La Carte pattern is its association with responsive goals, e.g. ‘detect event’, ‘handle exception’. For an example, see Section VI-B and Figure 7.

Returning to the issue of pragmatics, it is arguable that the second ‘without’ phrase [O1] can be understood more naturally as an obstacle than as either an alternative goal or a tactical choice. That is, the goal of ‘enhance existing exchanges’ [G3] could be obstructed by ‘service disruptions’ [O1] but the conflict can be resolved by achieving the sub-goal of ‘easy integration’ [G9]. Figure 4 illustrates the ‘Resolve Obstacle’ pattern. It can be expressed by the theorem:

\[
\Diamond p \land \Box (r \models \neg p) \land \Box (q \models \neg r) \\
\Rightarrow \Box q \quad (11)
\]

This can be read informally as: if we require \( r \), which could be blocked by \( r \), and we expect that \( q \) would block \( r \), then it follows that we require \( q \).

2) ‘Social’ level of meaning: It is also possible to take the social level of meaning into account in goal analysis. For example, the social context of the System-A enhancement project included the business objectives of Company-C, a large company owned by shareholders. Using our knowledge of this kind of organisation, we can infer that the phrase ‘draws heavily upon Company-C’s extensive experience’ [G4] refers to a shared sub-goal of two parent goals:
that we can satisfy

1) an explicit Company-B, i.e. customer-facing, goal of ‘easy integration’ [G9], and

2) an implicit Company-C, i.e. supplier-facing, goal of achieving ROI objectives [G16], e.g. by maximising the value of the stakeholder’s investments in the skills and experience of its staff.

The ‘Win-Win’ pattern is illustrated in Figure 5. It can be expressed by the theorem:

\[ \diamond p_1 \land \diamond p_2 \land [q \implies p_1] \land [q \implies p_2] \implies \diamond q \]  

(12)

This can be read informally as: if we require \( p_1 \) and \( p_2 \) and we expect that \( q \) would entail both \( p_1 \) and \( p_2 \), then it follows that we can satisfy \( p_1 \) and \( p_2 \) by requiring \( q \).

Fig. 4. Resolve Obstacle goal pattern

VI. COMPOUND PATTERNS

Natural languages enable more complex utterances to be constructed from simpler components. Similarly, a pattern language adds value to isolated patterns by suggesting ways in which they can be combined. In this section we briefly present two illustrative examples of larger patterns that themselves contain two or more of the ‘atomic’ patterns described in Section V.

A. Authorise Project pattern

Atomic goal patterns can be combined and customised to construct domain-specific patterns. The example in Figure 6 generalises the goals that we inferred in Section V for the Project Proposal document. This pattern is specific to the domain of project management and assumes a business process, e.g. the PRINCE2 project management method, in which projects are authorised to commence by a Project or Programme Board.

B. Mitigating Actions pattern

The Resolve Obstacle pattern described in Section V-C.1 often occurs in more elaborate forms, e.g. the primary goal, the obstacle, and its resolution each have refinement sub-graphs. Consider the goal graph in Figure 7 for starting a car and driving off from a parking place.

The primary goal rationale is shown (incompletely) in the refinement commencing from ‘Drive off from parking place’. However the ‘Car engine running’ sub-goal may be obstructed; several (of many possible) alternative obstructions are shown in the diagram. Each sub-obstacle requires its own resolution by a sub-graph of resolving goals such as that shown for ‘no petrol’. This produces a visual arrangement of refinement sub-graphs that are alternately of primary goals, obstacles, and resolving goals which is easy to spot from a ‘bird’s-eye’ view of the graph.

Fig. 5. Win-Win goal pattern

http://www.ogc.gov.uk/prince2/
requirements engineering, particularly in the context of agile approaches to system development.

The theoretical aspects of our work lead in several directions. For example, the use of patterns as a communication technique is helping us to explore the complex question of how and whether stakeholders reach a shared understanding of goals, assumptions and requirements. The relationship between goal patterns and the subsequent evolution of a system also merits further study. In particular, we wish to investigate whether there is a consistent relationship between goal patterns and different scenarios of system evolution. This would shed valuable light on the validity and applicability of Lehman’s ‘laws of software evolution’.

ACKNOWLEDGMENTS

The goal graph diagrams were produced using the Visual-Goals tool, developed in the Applied Software Engineering Research Group, University of Reading by Marc Bartsch in collaboration with Ken Boness and Stephen Cook.

Access to the case study material was kindly provided by the Centre for Empirical Software Process Research, University of Hertfordshire.

REFERENCES


[16] A. van Lamsweerde and E. Letier, “From object orientation to goal orientation: a paradigm shift for requirements engineering” in Radical Innovations of Software and Systems


