

Towards More Realistic Assumptions about Organizations in Goal-Oriented Requirements Engineering Frameworks

Ivan J. Jureta and Stéphane Faulkner¹

Abstract—Requirements engineers analyze information system (IS) requirements with a number of explicit and implicit assumptions about human organizations. Such assumptions influence the construction and use of requirements engineering (RE) frameworks. Ultimately, they affect IS requirements' quality. This paper overviews recent goal-oriented RE (GORE) frameworks by discussing three assumptions about human organizations: bounded human rationality, opportunism in human behavior, and organizational complexity. A discussion of implications results in a set of desirable and undesirable characteristics for GORE frameworks. They are implemented in a framework, named REQUEST, to illustrate one possible implementation in a RE framework. Theoretical discussions are interwoven with examples from a real world industrial case study in which REQUEST was used to engineer IS requirements at a large international steel producer.

Index Terms— Goal-oriented requirements engineering, goal, softgoal, framework, meta-model, case study, steel industry.

I. INTRODUCTION

Requirements engineering (RE) is concerned with the identification of goals to be achieved by an information system (IS), the operationalization of these into specification of IS services and constraints, the identification of resources required to perform those services, the assignment of responsibilities for the resulting requirements to agents, such as humans, devices and software available or to be developed, the specification of timing issues in the operation of the IS, and localization of people and services that take part in the IS.

Over the last two decades conceptual models, frameworks, and methodologies have been suggested to assist engineers in realizing RE activities (e.g., [33], [14], [11], [28], [32], [42], etc.). Among these, the *Goal* concept is widely regarded as a useful tool for RE (e.g., [44]) and plays a central role in Goal-Oriented RE (GORE) frameworks (for reviews, see, [44], [45], [21]). Several case studies and real industrial applications illustrate the practical value of this concept (see, e.g., [45]). It has been suggested that Goals are a useful tool in various RE activities, such as elicitation, elaboration, structuring, specification, analysis, negotiation, documentation, and

modification of IS requirements [44].

Considerable research efforts focus on constructing and improving GORE frameworks (e.g., [11], [4], [20], [21], [26]). The motivation of this paper is not novel in this respect. However, a series of novel research issues are identified by introducing explicitly three assumptions about human organizations in which RE activities are situated. In particular, it is assumed that human rationality is bounded [38], their behavior at least partly opportunistic [48], and that human organizations are complex entities [6]. By discussing the implications of these assumptions, desirable and undesirable characteristics of GORE frameworks are suggested in light of related work. To illustrate one possible integration of the desirable features within a GORE framework, a framework comprising them (named REQUEST) is presented. Theoretical discussions are interwoven with real-world examples to illustrate the use of the framework.

The principal aim of the paper is to initiate a discussion of the impact of explicitly introducing the three assumptions on the design of GORE frameworks. In this respect, REQUEST is primarily a platform for discussing framework requirements derived from the identified implications.

The following section presents preliminary information about the case study so as to facilitate the comprehension of examples used further in this paper. Section 3 relates the aforementioned assumptions with seminal organization science and economics research. Section 4 discusses the way that bounded rationality, opportunism, and organizational complexity influence the very design of an RE framework – i.e., tools and techniques for various RE activities. Section 5 concludes and points to further work.

II. OVERVIEW OF THE CASE STUDY

The REQUEST framework has been applied in a European steel production plant, part of one of the largest international steel producing companies. While the plant's installations carry out transformations of coal into successively coke, sinter, and pig iron, the scope of the case study is limited to the IS used in the coking plant for the transformation of coal into coke. Coke is a combustion agent used in the production of steel and is produced in coke ovens. It is obtained through a treatment of coal.

¹ Both authors are with PReCISE, University of Namur. Emails: ivan.jureta@fundp.ac.be, stephane.faulkner@fundp.ac.be.

After the coal is delivered by rail or road, it is stored. The first physical treatment of coal consists of blending and crushing different types of coal, in order to obtain a coal charge. The coal charge is transported to a coal tower using a series of conveyor belts. The coal tower, placed above coke ovens, fills a charging machine, which is then positioned over opened ovens in order to charge them with the coal mixture. Once an oven is charged, it is closed and heated to approximately 1200°C to produce coke. After coking is finished, a pusher car empties each oven onto a coke guide. From the coke guide, coke is charged into a quenching car, which transports it in the quenching tower. Once in the quenching tower, coke is cooled with 25m³ of water. The quenching car then transports coke to and discharges it onto a screen that filters the pieces of coke to separate ones suitable for use in a blast furnace, from others that are to be carried off to a sinter plant.

Because coke production activities are repetitive and dangerous (due to high temperatures and quantities of materials transported around the plant), the automation of most activities in the coke production plant is considered as a viable way to improve safety and productivity. A GORE approach has been applied to specify requirements for an IS that will control the automated coke production machinery in the plant.

III. MAKING ASSUMPTIONS EXPLICIT

GORE frameworks share a common approach: they attempt to identify goals of an IS and then derive required behavior of software, hardware, and people. When goals are not explicitly stated in a requirements document, [44] suggests that they can be found by analyzing an existing IS – i.e., the requirements engineer can negate the formulations of problems and deficiencies of the existing IS to find goals for the future one. Goals can also be found by systematically searching for intentional keywords in preliminary documents provided by the organization and from transcripts of interviews with members of the organization.

Experience shows that goal identification is difficult. Additional assumptions about the context from which Goals originate shed light on why this activity is tricky. Seminal works in organization sciences (see, e.g., [36], [38], [39], [6]) argue that organizations involving people may not have a consistent and agreed upon set of preferences (i.e., goals) and that their members may not be aware of inner workings of organization's operational processes. This is due essentially to bounded rationality (e.g., [37]-[39]), opportunism of organization's members (e.g., [48]) and complexity of the organization (e.g., [6]). In a seminal paper [48], Williamson summarizes behavioral assumptions of economics in the following:

“Human nature as we know it is marvelously rich and needs to be reduced to manageable proportions. The two behavioral assumptions [...] without which the study of economic organization is pointless – are bounded rationality and opportunism. As a consequence of these two assumptions, the human agents that populate firms

[...] differ from economic man [i.e., a human agent who makes choices in decision situations based on rational evaluation of all possible alternatives, while having a complete understanding of the problem at hand] in that they are less competent in calculation and less trustworthy and reliable in action. [...] A condition of bounded rationality is responsible for the computational limits of organization man. A proclivity for (at least some) agents to behave opportunistically is responsible for their unreliability. The term bounded rationality was coined by [H. A.] Simon to reflect the fact that economic actors, who may be presumed to be 'intendedly rational', are not hyperrational. Rather, they experience limits in formulating and solving complex problems and in processing (receiving, storing, retrieving, transmitting) information. [...] Opportunism effectively extends the usual assumption of self-interest seeking to make allowance for self-interest seeking with guile.”

Organizational complexity is a different issue. In an influential paper on decision-making in organizations [6], it is suggested that human organizations have the following characteristics (although the degree to which they are present varies from one organization to another):

“The organization operates on the basis of a variety of inconsistent and ill-defined preferences. It can be described better as a loose collection of ideas than as a coherent structure; it discovers preferences through action more than it acts on the basis of preferences. [...] Although the organization manages to survive and even produce, its own processes are not understood by its members. It operates on the basis of simple trial-and-error procedures, the residue of past experience, and pragmatic inventions of necessity. [...] Participants [i.e., members of the organization] vary in the amount of time and effort they devote to different domains; involvement varies from one time to another. As a result, the boundaries of the organization are uncertain and changing; the audiences and decision makers for any particular kind of choice change capriciously.”

Suggestions from organizational and economics need to be considered when constructing tools and methods of RE, as RE is an activity situated in human organizations.

Related work deals with the impact of assumptions (in general) on results of diverse IS development steps, including RE: User models have been constructed [8], suggestions have been made regarding the structuring of development problems [47], IS analyst and user values have been studied [23], and the impact of IS development paradigms on final IS characteristics and development approaches has been discussed [15]. Implications of the assumptions outlined above on the design of GORE frameworks have received limited treatment.

More restricted notions of bounded rationality and organizational complexity in relation to RE are discussed in [21]. The focus is on the impact of bounded rationality on the behavior of an IS once developed and implemented. It is argued that some constraints on the operation of an IS can be ensured, as hardware and software will be designed according

to requirements, but that constraints cannot be enforced on people. Therefore, the requirements engineer needs to formulate and test hypotheses about the operation of a future IS. Similar considerations have already been voiced in [42]: It is argued that an IS requirement should be distinguished from an assumption. The former is a goal that can be formulated in terms of states controllable by a software component, while the latter is controlled by an environmental agent (i.e., hardware and software not in the scope of the IS for which requirements are engineered, or humans who will participate in the IS).

Regarding organizational complexity, [21] suggests that conceptual models of an IS should be used to simulate IS behavior, facilitating choice between alternative IS structures: “without testing of the models it is impossible to comprehend their implications by merely observing, walking through, and debating about their contents. Nor is it always feasible to test them through observations from experimentation in the real world.” Few would disagree with these suggestions.

Research efforts mentioned above can be seen as treating problems of bounded rationality and complexity of already developed IS. Research presented in this paper is concerned with steps that can be taken *prior* to IS development, at the stage of GORE framework design. Notice that the former and the latter research questions are related, as, e.g., tests of models cannot be built if the RE framework employed does not allow testing.

Consequently, to assume that the rationality of members of an organization is bounded, that they behave opportunistically, and that the organization is a complex entity, influences considerably not only the way a GORE framework will be used, but more importantly the very design of the framework – i.e., its tools and techniques for the elicitation, modeling, analysis, and validation of IS requirements.

IV. IMPLICATIONS

The aim of this section is triple: (i) to suggest an initial and incomplete set of requirements (each subsection presents one such requirement) that need to be taken into account when designing a GORE framework. They result from the assumptions introduced in the previous section; (ii) to discuss the degree to which GORE frameworks (listed in Table I) satisfy these requirements; (iii) to suggest how the requirements can be satisfied within a novel GORE framework, called REQUEST; (iv) to illustrate, when possible the use of tools and methods with examples from a real industrial RE project.

A. *Precisely Define and Specify Concepts*

RE frameworks generally use concepts as means to represent and support reasoning about an IS. Common concepts such as *goal*, *actor*, and *task* are often enriched with attributes, links, and taxonomies (for a review of taxonomies for the *goal* concept, see [44]). The aim is to facilitate and improve heuristic, qualitative, and/or formal reasoning involved in RE.

The purpose of a concept is to manipulate information only about aspects of the real-world considered relevant for the problem at hand. Consequently, a small number of well

chosen concepts, relationships, and attributes should be integrated in a framework. While the choice is difficult due to the diversity of problems encountered in RE projects and the willingness to ensure expressivity, formal definition and specification of the chosen modeling elements can be of considerable help. Ambiguity and overlapping meaning (i.e., redundancy) can be avoided, thus reducing by objective means the set of elements included. Ambiguity and redundancy are particularly awkward when teams engineer IS requirements – lack of agreement on the meaning of abstractions lead to misunderstanding among team members. The latter results in incoherent IS requirements. Part of the RE project resources are wasted solely due to a lack of solid conceptual foundations.

In terms of the assumptions above, any framework that contains redundancies and/or ambiguity is of limited value, as it increases complexity of an already complex organizational context in which bounded rational agents attempt to solve intricate problems.

Consequently, formal definition and specification of RE modeling elements are desirable – even necessary – in response to bounded rationality of human agents who “experience limits in formulating and solving complex problems and in processing (receiving, storing, retrieving, transmitting) information” [48]. They make it possible to construct frameworks with internally consistent and non-redundant sets of concepts. While this requirement entails more involvement from the RE researcher when constructing a framework, it guarantees the practitioner that the framework is established on solid conceptual foundations.

TABLE I
Goal-oriented RE frameworks from related work

Framework name:	Some publications:
KAOS	[11], [24], [25], [42], [46]
Tropos	[4], [13]
NFR	[28], [5]
i-star	[49], [50]
REF	[10]
ESPRIT-CREWS	[32]
GRL	[26]
GDC	[19], [20]
GBRAM	[1], [2]
Lightswitch	[34], [35]

Frameworks in Table I feature precise semi-formal definitions of concepts. That is, definitions are given in precise English. However, meaning of key words that they contain is often not made explicit. This is true for example for the *goal* concept – a recent GORE review paper defines Goal with the following:

“Goals are prescriptive statements of intent whose satisfaction requires the cooperation of agents (or active components) in the software and its environment.” [45]

At least two questions arise when one attempts to interpret the definition: What is meant by “statement”, “intent”, and “cooperation”?; and: Does achieving a goal always require more than one agent to act? As a consequence, *Goal* may represent different things to different people, making it difficult to identify IS Goals. The content of a Goal (i.e., what aspect of the real world the Goal characterizes) does not have a clear meaning.

Two approaches can be adopted to provide a concept definition. A definition can use more abstract, higher-level constructs in order to allow freedom in interpretation (as in the definition above). On the contrary, precision can be increased if precise, already well defined constructs are used. A helpful definition based on solid theoretical foundations can be given if GORE modeling elements are considered in relation to the information used in subsequent IS development steps. This seems adequate as these abstractions manipulate attempt to manipulate information used extensively in subsequent IS development steps.

One of the main outputs of RE activities is a formal specification of at least some parts of the IS for which the requirements are engineered. KAOS [11] and Tropos [4] both include a formal layer for specifying properties of the IS using a variant of first-order logic. While other frameworks do not explicitly consider the need for formal specifications, it seems impossible to move from informal or semi-formal requirements specifications to the development of the IS without writing and checking formal requirements specifications (see, e.g., [3], [41], [43]).

Consequently, it is useful to define concepts in terms of formal specifications’ basic building blocks. According to [41], an IS has *static* and *dynamic* characteristics. *Static characteristics* include: states an IS can occupy, and invariant relationships that are maintained as the system moves from one state to another. *Dynamic characteristics* include: actions that can be executed in the IS, relationships between inputs and outputs of actions, and changes of state that happen in the IS. If relevant properties of an IS can be described in terms of static and dynamic characteristics, then all RE abstractions need to represent static and/or dynamic properties. Thus, to define RE abstractions in terms of logic building blocks allows a formal specification to be derived systematically and consistently from informal or semi-formal requirements specifications.

In REQUEST, the following definition of *Goal* is proposed:
A Goal is a desirable constraint on possible states of an IS.

As a constraint, Goal can be defined in terms of basic building blocks of logic. It can be conceptualized as a predicate of first-order logic (or of some of its variants, e.g., first-order temporal logic): **Goal** := **Predicate**, where **Predicate** represents a logic predicate including, e.g., temporal operators if required. Such definition of Goal leaves considerable freedom regarding the writing of the formal specification of Goal instances.

With a precise informal and a formal definition, the requirements engineer knows what kind of information a formal specification of a Goal needs to contain. With the exception of KAOS [11] (where a formal definition is not explicitly given, but can be inferred from examples), this issue is not treated in RE frameworks making it difficult to give a formal specification of instances of Goal and other modeling abstractions.

In contrast to KAOS, the suggested formal definition of Goal allows specialized semi-formal and formal definitions of Goal specialization taxonomy members to be proposed. In REQUEST, functional goals (called *HardGoals*), are defined as:

A HardGoal is a desirable and achievable constraint on possible states of an IS.

While formally defined in the same way as Goal, a formal achievability condition accompanies a *HardGoal*. If *State(sn, is)* gives a formal specification of an information system named “is” in a state named “sn”, then *HardGoal* achievability implies:

$$(\forall hg: \text{HardGoal} ; is: \text{InformationSystem}) \\ [hg \Rightarrow \exists sn: \text{ISState} ; (hg \wedge \text{State}(sn, is))]$$

The condition above indicates that if there is a state of an IS such that the conjunction of the *HardGoal* and of the IS state specification holds true, the *HardGoal* is achievable. In other words, there must be at least one state of an IS in which the *HardGoal* is achieved.

Nonfunctional Goal (called *SoftGoal* in REQUEST) is defined as:

A SoftGoal is a desired and satisficeable constraint on alternative HardGoals, and a criterion for selecting among alternative HardGoals of an IS.

Being defined in terms of *HardGoals*, a *SoftGoal* can be formally specified. An example of a *SoftGoal* is given below:

$$\forall co: \text{CokingOven} ; cc: \text{ChargingCar} \bullet \\ co.onrepair \Rightarrow (co.id-2) \leq cc.location \vee cc.location \geq \\ (co.id+2) \text{ affects (worker safety) with preference} \\ (((co.id-5) \leq cc.location \vee cc.location \geq co.id+5)) \square \dots \square \\ ((co.id-2) \leq cc.location \vee cc.location \geq (co.id+2)))$$

It indicates that a charging car (i.e., a device that carries several tons of coal and places it in coking ovens ready for coal cooking) is to be located at least one empty coking oven away from both sides of the coking oven that is being repaired.

The *SoftGoal* definition contains three kinds of information in a formal, hence precise way:

(i) It indicates the IS quality (or nonfunctional requirement) being affected by the specified property – here, worker safety is affected by the property *cc.location* when *cc.onrepair* holds.

(ii) It lists alternative *HardGoals* that can satisfice (i.e., partially satisfy [40]) the *SoftGoal*. For example, one of these *HardGoals* is:

$$\forall co: \text{CokingOven} ; cc: \text{ChargingCar} \bullet$$

$co.onrepair \Rightarrow (co.id-3) \leq cc.location \vee cc.location \geq (co.id+3)$

(iii) Using formal economics notation for preferences [22], preference among alternative HardGoals is formalized: in the example, $((co.id-5) \leq cc.location \vee cc.location \geq co.id+5))$ is preferred over $((co.id-4) \leq cc.location \vee cc.location \geq co.id+4))$.

This nonfunctional goal conceptualization is a significant departure from related work in that it allows the ideas of *satisficability*, *quality*, *alternatives*, and *preference* to be combined, and in a formal approach. Quality has been introduced in formal specifications, allowing traceability of desired high-level qualities to low-level precise IS property specifications. The notion of preference is integrated into formal specifications of nonfunctional requirements. The formalization of preferences is based on proven conceptualizations in economics, ensuring sound theoretical foundations and a powerful potential extensions for reasoning about nonfunctional requirements.

The preceding discussion illustrates benefits of providing formal modeling element definitions. Precise treatment is of significant value in the face of intricate problems in complex organizational contexts, implying the need for solid formal conceptual foundations of RE frameworks to limit ambiguity and redundancy.

Instantiation of formally defined modeling elements allows systematic and structured writing of formal specifications of IS requirements. In this paper, formal specification of modeling elements is considered different than their formal definition for two reasons. First, a formal definition of a concept concerns generally the concept itself, without constraining the way that the concept interacts with other concepts in a framework. Second, when constructing a framework, a methodology to assist heuristic, qualitative, and/or formal reasoning is generally proposed. Formal specification of modeling elements allows the designer of the framework to enrich concepts with e.g., attributes that allow some specific manipulation to be realized.

In REQUEST, the Z formal specification language [41] is used to specify all concepts. For example, the Goal concept is specified in the following way:

[NAME, INFORMAL_DESCRIPTION, Predicate,
GoalRefinementAlternative, Obstacle, Conflict]

Goal

GoalName: NAME

InformalDescription:
INFORMAL_DESCRIPTION

FormalSpecification: Predicate

GoalRefinedBy: Π GoalRefinementAlternative

ObstructedBy: Π Obstacle

ResolvesObs: Π Obstacle

ResolvesConfl: Π Conflict

$\forall g_1, g_2: \text{Goal} \bullet g_1.name = g_2.name \Rightarrow g_1 = g_2$

Two Goals that carry the same name are identical.

$\forall g: \text{Goal} \bullet ((\exists hg: \text{Hardgoal} ; g.name = hg.name) \vee (\exists sg: \text{Softgoal} ; g.name = sg.name)) \wedge ((\neg \exists hg: \text{Hardgoal} ; g.name \neq hg.name) \vee (\neg \exists sg: \text{Softgoal} ; g.name \neq sg.name))$

Any Goal is either a Hardgoal or a Softgoal, never neither and never both.

$\forall g: \text{Goal} \bullet ((\exists og: \text{OrganizationalGoal} ; g.name = og.name) \vee (\exists pg: \text{PersonalGoal} ; g.name = pg.name)) \wedge ((\neg \exists og: \text{OrganizationalGoal} ; g.name \neq og.name) \vee (\neg \exists pg: \text{PersonalGoal} ; g.name \neq pg.name))$

Any Goal is either an Organizational Goal or a Personal Goal, never neither and never both.

Reminiscent of formal definitions, formal specifications allow the framework designer to further reduce ambiguity by providing precise indications regarding concept instantiation, relationships, and attributes. The Goal concept specification associates a series of attributes (e.g., InformalDescription, obstructedBy, etc.) and constraints to the content of a Goal. The latter serve to restrict values of attributes. Attributes allow the specification of relationships between concepts: for example, a Goal instance in REQUEST can be linked to an instance of a Conflict concept with the relationships resolvesConfl, when the former can be said to resolve the latter. Using formal specifications, the condition that a Goal instance resolves a Conflict instance can also be specified formally as a constraint. Consequently, a resolveConfl link can be deduced from formal specifications of Goal and Conflict instances. This is impossible when semi-formal or informal approaches are used to engineer requirements. Earlier, a Goal has been specified with the following:

$\forall co: \text{CokingOven} ; cc: \text{ChargingCar} \bullet$

$co.onrepair \Rightarrow (co.id-3) \leq cc.location \vee cc.location \geq (co.id+3)$

This information is, in REQUEST terminology, the content of the HardGoal, and is the value of its FormalSpecification attribute. A Goal specification is written in the following way:

GoalName: ChargingCarAwayFromRepairedOven

InformalDescription: In order to ensure safety of workers repairing a coking oven, it is necessary to keep the charging cars at least three coking oven away from the coking oven being repaired.

FormalSpecification: $\forall co: \text{CokingOven} ; cc: \text{ChargingCar} \bullet co.onrepair \Rightarrow (co.id-3) \leq cc.location \vee cc.location \geq (co.id+3)$

GoalRefinedBy: \emptyset

ObstructedBy: \emptyset

ResolvesObs: \emptyset

ResolvesConfl: OvenCloseToRepairedInOperation

None of the frameworks cited above features formally specified modeling elements. KAOS and Tropos associate attributes to most concepts, but provide no constraints on their values, other than the type. It is thus difficult to verify the internal consistency of modeling elements' instantiations in these frameworks. This difficulty is illustrated in [9] where such an analysis of GRL revealed ambiguities, under-definitions, and internal inconsistencies, even though GRL is

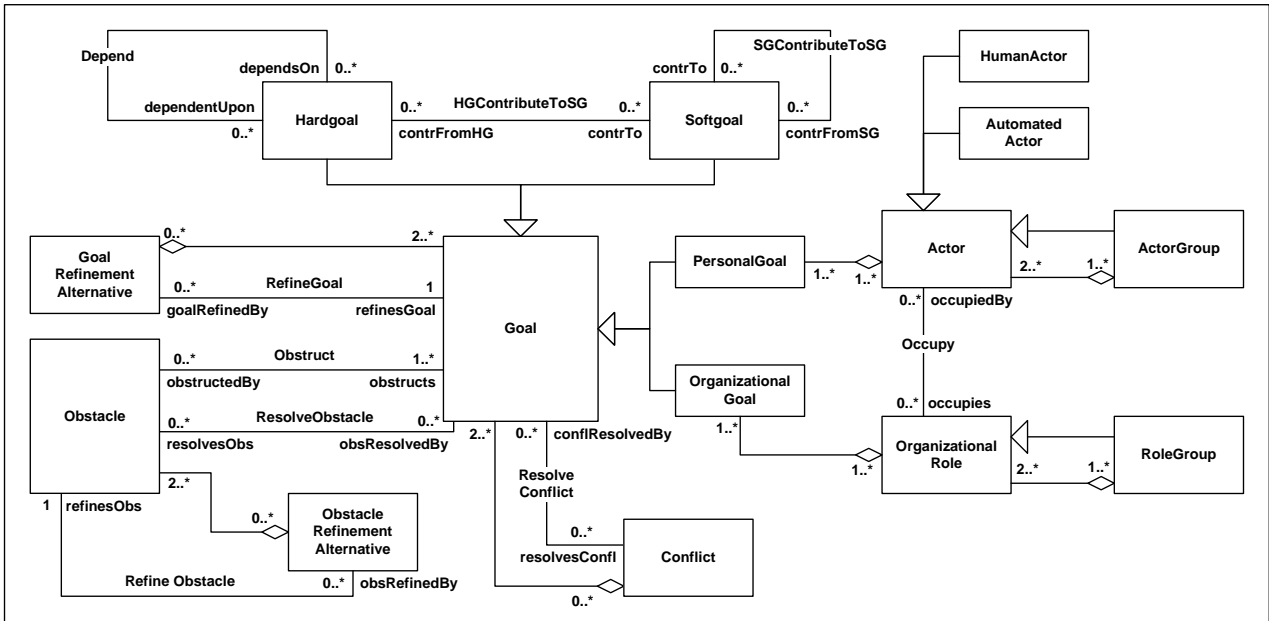


Figure 1. Part of the REQUEST Meta-Model

in the process of standardization. Formal specification at the framework construction stage allows at least some of the ambiguities and inconsistencies to surface, increasing the probability of resolution early in the framework lifetime.

B. Define a Conceptual Meta-Model

A meta-model containing modeling elements that are particularly relevant for the problem domain should be defined. Formal definition and specification of modeling elements goes hand in hand with meta-model definition (also referred to as abstract syntax definition [9]). When a complete formal specification is available, the meta-model is simply a visualization of a part/all of this information. There are several benefits to defining a meta-model:

- (i) It guides the RE effort by pointing to information considered relevant for the requirements engineer. For example, part of the REQUEST meta-model shown in Figure 1 indicates that the Goal concept participates in the Obstruct relationship with the Obstacle concept. Consequently, one of the activities in a methodology is to identify instances of the Obstruct relationship that may exist between instances of the Goal and of the Obstacle concepts.
- (ii) The meta-model determines the structure of the requirements database in which the acquired requirements are stored. It allows retrieval of reusable requirements fragments [11].
- (iii) It gives an overview of the concepts and relationships whose instances are used to organize information about the problem at hand.
- (iv) Framework extensions can be added with precision as interaction of new abstractions with existing ones can be carefully studied.
- (v) When resulting from formally defined and specified modeling elements, a meta-model allows objective and

systematic comparisons of frameworks so as to decide on e.g., framework suitability to a particular IS development project.

Similarly to formal definition and specification, the role of the meta-model is to reduce ambiguity and redundancy by forcing the designer of the framework to consider relationships between concepts, specify cardinalities, and restrict at the outset the freedom in the use of the framework. Complexity of the framework becomes visible, allowing reduction of the number of concepts and relationships to take place so as to bring the meta-model to manageable proportions.

A meta-model for a requirements acquisition framework has been first proposed in KAOS [11]. Tropos and GRL rely on the i-star meta-model (see, [49]), while REF and GBRAM do not provide on a meta-model. Notice that, although GRL extends the i-star metamodel, it does not precisely define its extensions since a new meta-model is not defined to integrate them. This is not a rigorous approach, as illustrated by the inconsistencies found in GRL in [9]. ESPRIT-CREWS [32] defines abstractions with a meta-model for a scenario, and a meta-model for the language structure of a goal (i.e., a goal is defined as composed of a verb and a parameter, which can be a target, a way, etc.).

Part of the REQUEST meta-model is shown using standard UML class diagram notation in Figure 1. Starting from the center of Figure 1, two taxonomies specialize the *Goal* concept. *HardGoal*, or functional *Goal* is distinguished from *SoftGoal*, or nonfunctional *Goal* (see previous section for details). Any *Goal* suggested by a stakeholder in the RE project is a *PersonalGoal*. If all stakeholders agree on the content of a *PersonalGoal*, an *OrganizationalGoal* with the same content is added (see below). Undesired constraints on IS states are represented as *Obstacles* that obstruct *Goal* achievement. *Obstacles* and *Goals* can be refined. Alternative sets of, respectively, *Goals* or *Obstacles*, constitute *Goal* or *ObstacleRefinementAlternatives*. When *Goals* specify

constraints that cannot hold in the same state of an IS, they generate *Conflict* that can be resolved by changing or introducing new *Goals*. To assign expected behavior (i.e., responsibility) to IS *Actors*, or active – action processing – entities, sets of *OrganizationalGoals* are grouped into *OrganizationalRoles*. In turn, the latter are assembled into *RoleGroups* that allow further treatment of *OrganizationalGoal* groups. *PersonalGoals* are grouped to characterize *Actors* by describing IS participants’ intentions, and *ActorGroups* represent composite intentional and active entities.

C. Reason about Inconsistent Preferences

While inconsistency management is recognized as a significant issue in IS development (e.g., [31], [16], [42]), only KAOS, NFR, and to some extent GBRAM feature concepts specialized for inconsistency analysis (among frameworks in Table I). As “[t]he organization operates on the basis of a variety of inconsistent and ill-defined preferences” [6], specialized concepts and methods need to be included in a framework to help make inconsistent preferences explicit, allowing prevention or resolution.

KAOS-specific Obstacle and Conflict concepts [42] are reused in the REQUEST framework. Also, the concept of goal contributions from NFR has been reused in REQUEST to detect potential Conflicts among goals.

To illustrate the use of the Conflict concept, consider the following example. While the *GuideCarAlignedWithCokingOven* Goal indicates that a guide car shall be placed next to an oven within D1 time units, *NoGuideCarAlignedWithCokingOvenDuringRepair* indicates that a charging car cannot be located next to an oven that is being repaired. The repairs are assumed to last at most D2 time units. Note that D1 and D2 are simply constants. Formal specifications of Goals are written in first-order logic extended with temporal operators such as e.g., “some time in the future, but not longer than X time units” (noted: $\diamond_{\leq X}$), “always in the future” (\square), etc. (e.g., [25] or [17] for more details on temporal operators.)

GoalName: *GuideCarAlignedWithCokingOven*

InformalDescription: When the discharging door of a coking oven is open, the guide car is placed next to the coking oven within a delay of D1 time units.

FormalSpecification: $\forall co: CokingOven ; GC: GuideCar \bullet \neg co.onrepair \wedge co.discharging_door = Open \Rightarrow \diamond_{\leq D1} GC.location \neq co.id$

GoalName:

NoGuideCarAlignedWithCokingOvenDuringRepair

InformalDescription: When a coking oven is repaired, the guide car shall not be located on the coking oven for at most D2 time units.

FormalSpecification: $\forall co: CokingOven ; GC: GuideCar \bullet co.onrepair \Rightarrow \diamond_{\leq D2} GC.location \neq co.id$

Consequently, there is a Conflict when $D2 > D1$. This is often the case as repairs last much longer than a coking oven remains open on the discharging side, provided the coking process operates at its regular performance. The Conflict is specified with the following:

ConflictName:

GuideCarAlignedWithCokingOvenDuringRepair

InformalDescription: Guide car is not next to a coking oven which is repaired for at most D2 time units, guide car is next to a coking oven after D1 time units, and $D2 > D1$.

FormalSpecification: $\exists co: CokingOven ; GC:$

$GuideCar \bullet co.onrepair \wedge \diamond_{\leq D1} GC.location \neq co.id \wedge \diamond_{\leq D2} GC.location \neq co.id \wedge D2 > D1$

One way to avoid this Conflict is to add $\neg co.onrepair$ to the FormalSpecification of the *GuideCarAlignedWithCokingOven* Goal, on the left side of the implication.

A Conflict can be resolved by framework-generic resolution tactics from e.g., [36] and [42], such as: goal weakening, goal replacement with more or less restrictive goal formulations, substitution of the set of conflicting goals with a “compromise” goal, etc. In the example above, an existing Goal was made more restrictive in order to resolve the Conflict.

D. Reason about Opportunistic Preferences

To distinguish between self-interest and shared interest of IS development stakeholders, the REQUEST meta-model uses a specific goal taxonomy novel to related work (overviewed in [44]).

Guidelines of the REQUEST methodology indicate whether a Goal is Personal or Organizational: Any Goal proposed by a RE project stakeholder is considered by default to be Personal. It becomes Organizational if other stakeholders are not opposed to the achievement of that Goal through the IS being developed. Otherwise, the Goal remains Personal, or is modified to accommodate others stakeholders’ concerns. It is important to note that an IS is developed in order to achieve Organizational Goals only.

Representing Personal Goals helps identify Conflicts that they may have with Organizational or other Personal Goals. None of the frameworks in Table I treats this important issue. The above approach relies on transparency and consensus to attempt to ensure absence of inconsistencies due to personal agendas. Although it is imperfect, it is simple and a starting point in treating opportunism in a systematic way, without requiring sociological analysis of the organizational context.

E. Use a Flexible Visual Syntax

A visual syntax defines the visual representation of meta-model element instances in diagrams that complement natural language and formal representation of IS requirements. They are particularly useful in communicating requirements to stakeholders inexperienced in formal methods and/or a GORE framework.

Frameworks in Table I use symbols as the principal visual syntax element. For example, actors of an IS are represented in Tropos as labeled circles; in KAOS, goals are represented as labeled parallelograms. Tropos, GRL, and REF, and in general any framework that uses i-star visual syntax are not particularly flexible. There is no mechanism for breaking diagrams into fragments. A consequence is that such diagrams

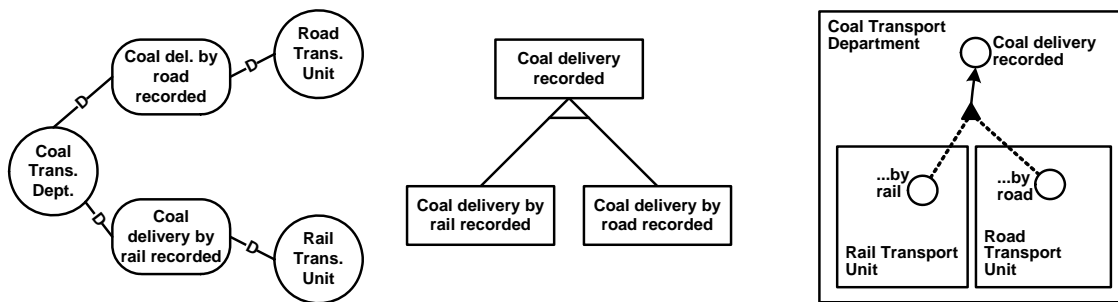


Figure 2. Representation of the Visual Syntax Example Using i-star (Left), Goal Graphs (Center), and REQUEST (Right)

do not scale well on large projects – i.e., it is difficult for users of the framework to focus attention on one part of a diagram while working with the entire diagram. An illustrative analogy is a geographic map. They exist in different scales, each scale showing a different level of detail, and each scale has its user group – some will require high level of detail, while others will be content with a large scale providing only an overview. The i-star visual syntax does not provide such desirable flexibility.

KAOS, GDC, and GBRAM use goal graphs. While goal graphs are simple, they do not show some useful information: They do not indicate responsibility assignment for abstract and specific (i.e., high- and low-level) goals nor goal ownership (i.e., who suggested the goal?). KAOS is to some extent an exception as it assigns responsibility to atomic (i.e., non-decomposable) agents only for low level, very specific goals.

Visual syntax from cartography served as the main source of rules for REQUEST visual syntax definition. It allows increased flexibility when moving between different levels of detail shown on requirements diagrams. Such flexibility requires specific relationships and concepts in a framework. For example, *RoleGroup* concept is a specialization of *OrganizationalRole* and is composed of instances of *OrganizationalRole*. The former relationship allows a *RoleGroup* instance to inherit all of an *OrganizationalRole* characteristics, thus making it possible to represent an entire *RoleGroup* as a single *OrganizationalRole* when less detail is required. In Figure 2, CoalTransportDepartment role would then be represented as a rectangle (i.e., an *OrganizationalRole*) without showing its internal composition.

None of the frameworks in Table I can represent complex actors and organizational roles. These are decomposable entities – e.g., an organizational unit can be a complex actor, because teams that work within the unit may also need to be represented at some point. Furthermore, no cited framework can show goals common to two or more actors or organizational roles. To illustrate this proposition, i-star, generic goal graph (applicable for KAOS, GBRAM, GDC, NFR, and ESPRIT-CREWS), and REQUEST visual syntax are used to model the following situation:

The coal transport department needs to record arrivals of coal deliveries to the coking plant. Because coal can arrive by rail or by road, two units are part of the coal transport department: Rail Transport Unit and Road Transport Unit.

Each unit is responsible to record coal delivery by rail or road.

Figure 2 represents this information using visual syntax common to most current frameworks, and compares it to the representation made with the visual syntax of REQUEST. Notice that the i-star notation does not contain information about units composing the department and having a common goal. The generic goal graph notation shows that a goal is refined into two more specific goals, without reference to responsibility or to composition of organizational roles. In KAOS, each of the two goals would need to be refined in order to be assignable to an atomic actor.

REQUEST visual syntax allows different views on formal requirements specifications to be given. While such visualization is common in e.g., UML (i.e., class diagrams are complemented with activity diagrams, etc. to represent different aspects of an IS), it is less elaborate in GORE cited frameworks. Figure 3 shows refinement links between goals in the case study IS, while Figure 4 shows interdependencies between goals. Each view on the data can have specific analyses associated to it. Views clearly facilitate manipulation of specific parts of requirements data by allowing RE project stakeholders to focus on parts of data at a time.

F. Facilitate the Learning about the Problem

To assume that an organization is a complex entity implies that it is a mistake to assume that its members have precise requirements. They face complex problems that they intend to resolve by using an IS. Due to the complexity of available technologies that can be used in an IS and of the organization, it is unlikely that they will have sufficient knowledge about both the available technologies and the problem. Restricted knowledge reduces the solution space of the problem. Consequently, a RE process should be structured as a learning process for stakeholders and the requirements engineer. The requirements engineer needs to learn about the problem because its solution will be strongly influenced by the specifics of the organization. Empirical studies (see, e.g., [7]) suggest that poor understanding of the domain is a primary cause for failure of the RE step of an IS development process.

Learning is needed because there is limited understanding of existing organization's operational processes [6]. Members of the organization need to study the problem that they aim to

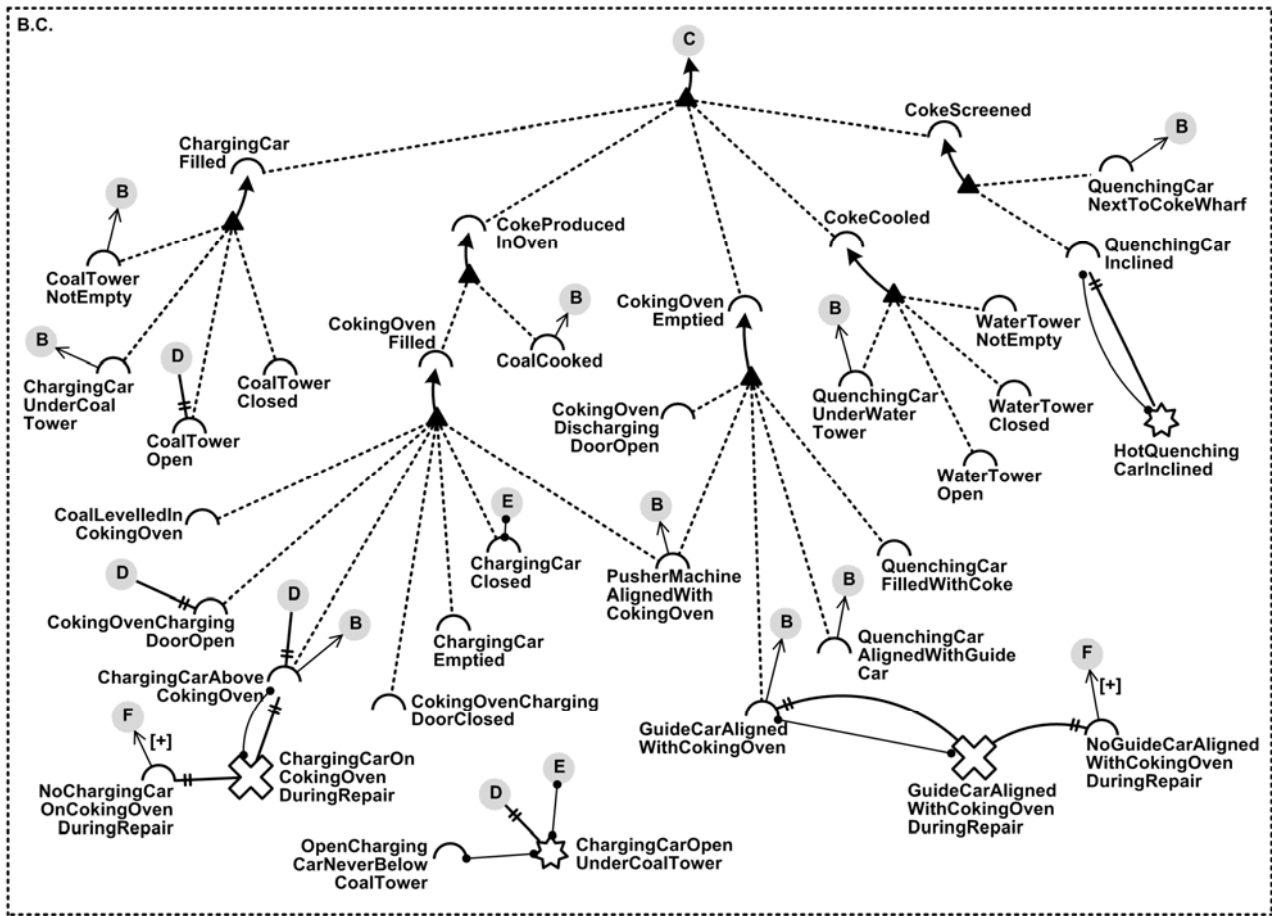


Figure 3. Refinement Links between nonoperational Goals. Single-letter labels allow us to avoid crossing links – that is, we break long links into two, and use a label to avoid drawing the entire line and thereby rendering the diagram less readable.

resolve through IS use. A GORE framework should make it possible to externalize tacit knowledge about the organization and the problem into explicit knowledge in order to gain a common understanding of the problem. Externalization will lead to the creation of new knowledge [29] – i.e., members will learn about aspects of the organization and available technologies that they did not know about. Learning about the problem will enlarge its solution space – e.g., they will learn how new information technologies might change operational processes and contribute favorably or unfavorably to the realization of the organization’s purpose. For the requirements engineer, the framework should allow to break the problem and its solution into intellectually manageable parts, suggest specific techniques to facilitate learning, and transform the obtained knowledge into IS requirements.

All frameworks in Table I provide heuristics for the elicitation, modeling, analysis, and validation of requirements. For example, a heuristic rule for goal identification in GBRAM is:

“Key action words such as: track, monitor, provide, supply, find out, know, ensure, keep, satisfy, allocate, speedup, improve, make, and achieve are useful for pointing to candidate goals.” [1]

On the more formal side, KAOS provides formal patterns for refining goals [12]. GDC is more focused on providing guidance in the interaction between the requirements engineer and members of an organization.

While REQUEST shares many methodological considerations and heuristics with existing frameworks, it has an important characteristic that facilitates learning about a problem. It decomposes the knowledge that needs to be learned in two parts: *strategic* and *operational knowledge*.

Strategic knowledge is tacit knowledge about desired and undesired aspects of a possible future state of the organization, in which responsibility is assigned to individual organizational roles or role groups occupied by potentially cooperating or conflicting actors or actor groups. Externalization of strategic knowledge will lead to a set of goals that result from the interpretation of organization’s purpose, conflicts and obstacles that harm the realization of the purpose, and responsibility assignments that need to be discharged by the members of the organization (i.e., actors that occupy organizational roles) in order to realize the purpose.

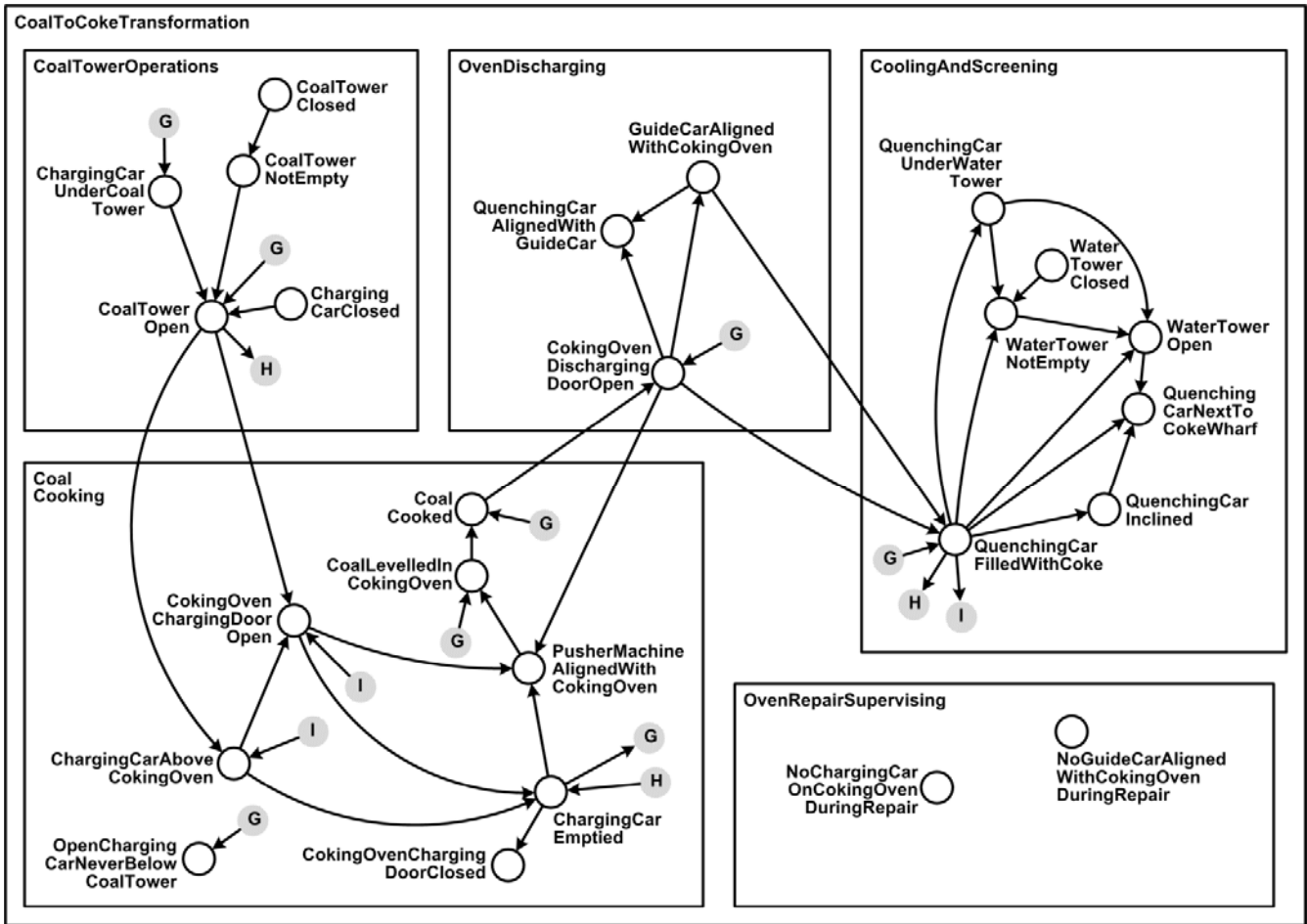


Figure 4. Interdependencies between operational Goals. As for refinement links, single-letter labels allow us to avoid crossing links. Rectangles define organizational roles; each role is responsible for the goals that fall within its rectangle.

In REQUEST, strategic information is acquired in terms of instances of concepts such as Goal and OrganizationalRole; relationships such as Occupy and Depend; and attributes such as FormalDescription. Part of the meta-model shown in Figure 1 is called the Strategic Requirements Model, and is instantiated into Strategic Requirements Diagram (for an example, see Figures 3 and 4).

Operational knowledge is tacit knowledge about ways in which a possible future state of the organization can be reached. Externalization of operational knowledge will result in descriptions of operational processes in which actors execute activities in order to transform objects, in the aim of realizing the purpose of the organization. Operational information is represented by instantiating modeling elements such as *Process* and *Capability* from the second part of the REQUEST meta-model, named Operational Requirements Model. For lack of space, this second part is not presented here (see, [17]). A distinctive feature of the ORM is its integration with UML 2.0, allowing reutilization of e.g., activity diagrams for operational and business process modeling and reducing time required to learn to use REQUEST for UML users.

The above distinction has several benefits:

(i) The two kinds of knowledge are fundamentally different (i.e., the former is about ends, and the latter about the means) and require different heuristics and guidance for elicitation, modeling, analysis, and validation. The REQUEST methodology has been constructed to provide guidance for each kind of knowledge.

(ii) Strategic and operational knowledge are separated as it is very likely that each will be provided by different stakeholders. Consequently, suggestions can be made regarding the identification of stakeholders relevant for each RE activity – that is, those likely to provide relevant strategic or operational knowledge helpful for the elicitation of requirements.

(iii) A problem in an organization can be analyzed with two, often complementary approaches: top-down, starting from strategic aspects of the IS and operationalizing them into operational requirements, and bottom-up, inferring strategic requirements from the analysis of operational aspects of an IS.

V. CONCLUSIONS AND FUTURE WORK

This paper introduces three assumptions about human organizations: bounded human rationality, opportunism in behavior, and organizational complexity. It discusses

implications of the assumptions and suggests desirable and undesirable characteristics of any GORE framework. They are implemented in a framework, called REQUEST, is outlined.

Compared to existing requirements engineering frameworks, REQUEST has several distinctive characteristics:

- It is developed by taking into account the hypothesis of bounded rationality of members of an organization in which the requirements for a future information system are being elicited, modeled, analyzed, and validated.
- It relies on a meta-model whose abstractions have been defined in terms of basic building blocks of mathematical logic. Hence, abstractions used during the requirements engineering process are defined in a precise manner.
- It selects, discusses, integrates and combines a set of concepts that have been proposed in different goal-oriented requirements engineering frameworks, but rarely used or discussed in a common context.
- It has precise conceptual and methodological links with the current standard software specification and documentation language – the Unified Modeling Language (UML). We argue that it can be used without much difficulty by UML users. See [17] for details.
- Relative to most existing goal-oriented RE frameworks, REQUEST has a number of desirable characteristics such as a flexible visual syntax, abstractions for dealing with inconsistent requirements, a formal requirements acquisition layer, etc.

Further work remains to be done on the comparison of existing GORE frameworks and on possible implementations of desirable framework characteristics discussed in this paper.

REFERENCES

- [1] Anton, A. *Goal-Based Requirements Engineering Method*. Ph.D. Thesis, Georgia Institute of Technology, 1997.
- [2] Anton, A. Goal-Based Requirements Analysis. In *Proceedings of ICRE '96*, IEEE, Colorado Springs, USA, 1996.
- [3] Bowen, J. *Formal Specification and Documentation using Z: A Case Study Approach*. International Thomson Computer Press, 1996.
- [4] Castro, J., Kolp, M., and Mylopoulos, J. Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27, 6 (2002) 365-389.
- [5] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J. *Non-Functional Requirements in Software Engineering*. Kluwer Publishing, 2000.
- [6] Cohen, M. D., March, J. G., and Olsen, J. P. (1972) A Grabage Can Model of Organizational Choice. *Administrative Science Quarterly*, 17, 1 (1972) 1-25.
- [7] Curtis, B., Krasner, H., and Iscoe, N. A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*, 31, 11 (1988) 1268-1287.
- [8] Dagwell, R. and Weber, R. System designers' user models: A comparative study and methodological critique. *Communications of the ACM*, 26, 11 (Nov. 1983), 987-997.
- [9] Dallons, G., Heymans, P., and Pollet, I. A template-based analysis of GRL. In *Proceedings of EMMSAD 2005*. ???
- [10] Donzelli, P. A goal-driven and agent-based requirements engineering framework. *Requirements Engineering*, 9 (2004) 16-39.
- [11] Dardenne, A., van Lamsweerde, A., and Fickas S. Goal-directed requirements acquisition. *Science of Computer Programming*, 20 (1993) 3-50.
- [12] Darimont, R., and Van Lamsweerde, A. Formal Refinement Patterns for Goal-Driven Requirements Elaboration. In *Proc. FSE'4 - Fourth ACM SIGSOFT Symp. on the Foundations of Software Engineering*, San Francisco, 1996, 179-190.
- [13] Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., and Traverso, P. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering*, to appear.
- [14] Hice, G. F., Turner, W. S., and Cashwell, L. F. *System Development Methodology*. North Holland, 1974.
- [15] Hirschheim, R. and Klein H. K. Four Paradigms of Information Systems Development. *Communications of the ACM*, 32, 10 (Oct. 1989), 1199-1216.
- [16] Hunter, A. and Nuseibeh, B. Managing Inconsistent Specifications: Reasoning, Analysis, and Action. *ACM Transactions on Software Engineering and Methodology*, 7, 4 (Oct. 1998) 335-367.
- [17] Jureta, I. *An Organizational Perspective on Goal-oriented Requirements Engineering Systems*. M.Sc. Thesis, IAG School of Management, University of Louvain, 2005.
- [18]
- [19] Kavakli, E. *Goal-Driven Requirements Engineering: Modelling and Guidance*. Ph.D. Thesis, University of Manchester, 1999.
- [20] Kavakli, E., and Loucopoulos, P. Goal-driven Business Process Analysis Application in Electricity Deregulation. *Information Systems*, 24, 3 (1999) 187-207.
- [21] Kavakli, E., and Loucopoulos, P. Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods. In Krogstie J., Halpin T., and Siau K. (Eds.) *Information Modeling Methods and Methodologies (Adv. topics of Database Research)*. IDEA Group, 2003, 102-124.
- [22] Kreps, D. *Notes on the Theory of Choice*. Westview Press: Boulder and London, 1988.
- [23] Kumar, K. and Welke, R. Implementation failure and system developer values: Assumptions, truisms, and empirical evidence. In *Proceedings of the Fifth International Conference on Information Systems*. (Tucson, Ariz., 1984), 1-13.
- [24] Letier, E., and van Lamsweerde, A. Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. In *Proceedings FSE 2004 - 12th International Symposium on the Foundation of Software Engineering*, ACM Press, Newport Beach, CA, Nov. 2004, 53-62..
- [25] Letier, E. *Reasoning about Agents in Goal-Oriented Requirements Engineering*. Ph.D. Thesis, University of Louvain, 2001.
- [26] Liu, L., and Yu, E. Designing information systems in social context: a goal and scenario modeling approach. *Information Systems*, 29 (2004) 187-203.
- [27] Munford, E., Participative Systems Design: Structure and Method. *Systems, Objectives, Solutions*, 1 (1981) 5-19, North-Holland.
- [28] Mylopoulos, J., Chung, L., and Nixon, B. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18, 6 (1992) 483-497.
- [29] Nonaka, I. A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5, 1 (1994) 14-37.
- [30] Nuseibeh, B., and Easterbrook, S. Requirements Engineering: A Roadmap. In *Proceedings of the 22nd International conference on Software Engineering (ICSE)*, 2000.
- [31] Nuseibeh, B., Easterbrook, S., and Russo, A. Leveraging Inconsistency in Software Development. *IEEE Computer* (Apr. 2000).
- [32] Rolland, C., Souveyet, C., and Ben Achour, B. Guiding GoalModeling Using Scenarios. *IEEE Transactions on Software Engineering*. Special Issue on Scenario Management (Dec. 1998) 1055-1071.
- [33] Ross, D. T., and Schoman, K. E. Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering*, 3, 1 (1977) 6-15.
- [34] Regev, G. *A Systemic Paradigm for Early IT System Requirements Based on Regulation Principles: The Lightswitch Approach*. Ph.D. Thesis, EPFL, 2003.
- [35] Regev, G., and Wegmann, A. Where do Goals Come From: the Underlying Principles of Goal-Oriented Requirements Engineering. In *Proceedings of the 9th IEEE International Symposium on Requirements Engineering (RE'05)*, IEEE Computer Society, 2005.
- [36] Robinson, W. N. and Volkov, S. A Meta-Model for Restructuring Stakeholder Requirements. In *Proceedings of the 19th International Conference on Software Engineering*, IEEE Computer Society Press, Botson, USA (May 17-24 1997).
- [37] Simon, H. A. *Models of man: Social and rational mathematical essays on rational human behavior in a social setting*. New York: John Wiley and Sons, 1957.

- [38] Simon, H. A. *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*. 3rd ed., New York: The Free Press, 1976.
- [39] Simon, H. A. Rational Decision Making in Business Organizations. *The American Economic Review*, 69, 4 (1979) 493-513.
- [40] Simon, H. A. *The Sciences of the Artificial*. 2nd Edition, Cambridge, MA: The MIT Press, 1981.
- [41] Spivey, J. M. *The Z Notation: A Reference Manual*. 2nd Edition, Prentice Hall International, 1992.
- [42] van Lamsweerde, A., Darimont, R., and Letier, E. Managing Conflicts in Goal-driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24, 11 (1998) 908-926.
- [43] van Lamsweerde, A. Formal Specification: a Roadmap. In Finkelstein, A. (Ed.) *The Future of Software Engineering*. ACM Press, 2000.
- [44] van Lamsweerde, A. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proceedings of RE'01, the 5th IEEE International Symposium on Requirements Engineering*, IEEE, 2001, 249-263.
- [45] van Lamsweerde, A. Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. In *Proceedings of RE'04, the 8th IEEE International Symposium on Requirements Engineering*, IEEE, 2004.
- [46] van Lamsweerde, A. The KAOS Metamodel –Ten Years After. University of Louvain, Internal report, Contact the author for a copy.
- [47] Vitalari, N. and Dickson, G. Problem solving for effective systems analysis: An experimental exploration. *Communications of the ACM*, 26, 11 (Nov. 1983), 948-956.
- [48] Williamson, O.E. The Modern Corporation: Origins, Evolution, Attributes. *Journal of Economic Literature*, 19 (Dec. 1981) 1537-1568.
- [49] Yu, E. *Modelling Strategic Relationships for Process Reengineering*. Ph.D. Thesis, University of Toronto, 1995.
- [50] Yu, E. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, IEEE Computer Society, 1997.