2010 18th IEEE International Requirements Engineering Conference

Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling

Ivan J. Jureta
FNRS & Louvain School of Management
University of Namur
ivan.jureta@fundp.ac.be

Alex Borgida
Department of Computer Science
Rutgers University
borgida@cs.rutgers.edu

Neil A. Ernst, John Mylopoulos
Department of Computer Science
University of Toronto
jm,nernst@cs.toronto.edu

***Abstract*—Techne is an abstract requirements modeling language that lays formal foundations for new modeling languages applicable during early phases of the requirements engineering process. During these phases, the requirements problem for the system-to-be is being structured, its candidate solutions described and compared in terms of how desirable they are to stakeholders. We motivate the need for Techne, introduce it through examples, and sketch its formalization.**

***Keywords*-Requirements models, goal-oriented requirements engineering, requirements modeling languages**

I. Introduction

Three intertwined questions remain among the central ones in Requirements Engineering (RE) for at least three decades now (e.g., [11]): (1) What information should be elicited from the stakeholders of the system-to-be? (2) What models should be used to represent the elicited information? (3) What kinds of reasoning should be performed over these models?

The initial response was to apply formal specification methods (FSMs) [4], [25] made for the design of the system-to-be, such as VDM [2], Larch [24], Z [20], B [1], Alloy [13]. As interest shifted from software and hardware alone to sociotechnical systems, it was recognized that RE must account for the variously (im)precise and (in)consistent expectations of the stakeholders, including the future users, owners, and so on. An understanding of the functions of the system-to-be could only be sought after beliefs, desires, and intentions of stakeholders had been grasped to a feasible extent. This led to a distinction between the *early* and the *late* phase of RE; FSMs remain applicable to the latter.

For the early phase of RE, seminal answers took the form of requirements modeling languages (RMLs) and typically included (1) an ontology of requirements stating the information to elicit, relevantly describing optative properties and behaviors of the system-to-be and its operational environment, (2) modeling primitives for the concepts and relations of the ontology, the instances of which together form models to record the elicited information, and (3) often automated methods that could be applied to the models in order to answer questions of methodological interest, such as if a model is consistent, or if the properties and behaviors it attributes to the system would allow the latter to satisfy its designated purpose. Early-phase RMLs such as RML [10], ERAE [6], KAOS [5] and *i** [26] often served as the starting point for further research on various issues of interest in RE.

The answers that an RML gives to the three key questions reflect its respective fundamental assumptions about the central concepts relevant for the early phase of RE, the very problem — the *requirements problem* — that RE aims to resolve during this phase and within the broader process of systems engineering, of how the problem can and should be resolved, and of when it is resolved. The concept of system or stakeholder *goal* (to represent desires) became central, as KAOS made clear. *i** went one step further with its focus for how various intentional agents/stakeholders are interdependent on each other and the system-to-be for the realization of their individual and joint goals.

However, as argued in [16], the core ontology for requirements and the requirements problem [29] — which are implicit in state-of-the-art RMLs such as KAOS and *i** — are limited in several respects that are critical for the successful performance of early-phase RE for contemporary systems. The CORE ontology for requirements [16] recognized that in addition to goals and tasks, different stakeholders have different preferences over requirements, that they are interested in choosing among candidate solutions to the requirements problem, that potentially many candidate solutions exist (as in the case of service-/agent-oriented systems, where different services/agents may compete in offering the same functions), and that requirements are not fixed, but change with new information from the stakeholders or the operational environment. In absence of preferences, it is (i) not clear how candidate solutions to the requirements problem can be compared, (ii) what criteria (should) serve for comparison, and (iii) how these criteria are represented in requirements models. New concepts suggested in CORE led to the revised formulation of the requirements problem: **Given** the elicited domain assumptions, goals, quality constraints, softgoals, tasks, some of which are optional, mandatory, and/or preferred over others, the engineer ought to **find** tasks and domain assumptions which satisfy all mandatory goals, quality constraints, and ideally also satisfy at least

1090-705X/10 $26.00 © 2010 IEEE
DOI 10.1109/RE.2010.24

some of the preferred and/or optional goals and quality constraints. A candidate solution to this problem will be a consistent set of tasks and domain assumptions which satisfy all mandatory goals and quality constraints. Candidate solutions are compared on the basis of which preferred and/or optional goals and quality constraints they satisfy. New concepts and the revised formulation of the requirements problem have created the need for new RMLs to be designed to accommodate these concepts and to resolve the revised problem.

The main objective of this paper is to introduce *Techne*, an abstract RML designed as the core on which to build new RMLs. By core is meant a minimal set of components which are argued as necessary to an RML, if it is to respond to the issues raised above: to model goals, softgoals, quality constraints, domain assumptions, and tasks used to define the requirements problem for a specific system-to-be, to define candidate solutions, to model preferences and optional requirements, and use them as criteria for the comparison of candidate solutions. The simplest way to make an RML from Techne is to add a visual syntax, e.g., a diagrammatic notation, and map its syntactic elements to those of Techne's syntax. Making an RML from Techne ensures that the RML is equipped to model instances of the concepts from the CORE ontology for requirements, then identify and compare candidate solutions to the requirements problem. Techne is a formal language. Its semantic domain is made up of structures called *candidate solutions* to the requirements problem, which are consistent sets of requirements satisfying some additional properties. Candidate solutions are found via paraconsistent and non-monotonic reasoning. Reasoning is paraconsistent because an inconsistent model should not allow us to conclude the satisfaction of all requirements therein; it is non-monotonic in that prior conclusions drawn from a model may be retracted after new requirements are introduced.

The paper reviews the features of Techne and sketches its formalization via realistic examples of manageable size from an online music-on-demand system (§II). Related work is presented (§III), limitations of Techne are discussed (§IV), and conclusions and pointers for future work are given (§V).

II. Features of Techne

At the very outset of the RE process — the early phase — the first interactions with the stakeholders result in variously (im)precise and (in)consistent information about the system-to-be and its operational environment. To be applicable to the early phase, an RML ought to handle the *classification* and *relation* of this information, its use in *modeling* in order to formulate the requirements problem for the system-to-be, and the *analysis/reasoning* on models to answer questions asked towards the resolution of the requirements problem. Techne offers facilities to support each of these tasks; we review them in turn in the rest of this section.

A. Classification

Techne builds on the CORE ontology [16] for classifying elicited requirements: the overall idea is to distinguish in a piece of information the psychological mode from the rest of the statement, then establish which CORE concept the statement instantiates, and this based on the psychological mode (belief, desire, and so on) and on some properties of the statement itself. Stakeholder desires become instances of the goal concept, if they refer to conditions, the satisfaction of which is desired, binary and verifiable (e.g., "Deliver music to clients via an online audio player"). If desires constrain desired values of non-binary measurable properties of the system-to-be, then they are instances of the quality constraint concept (e.g., "The bitrate of music delivered via the online audio player should be at least 128kb/s"). When desired values are vaguely constrained and on not necessarily directly measurable properties, they instantiate the softgoal concept (e.g., "Buffering before music starts in the audio player should be short"). Stakeholder intentions to act in specific ways become instances of tasks to be accomplished either by the system-to-be, or in cooperation with it, or by stakeholders themselves. Beliefs are instances of domain assumption, stating conditions within which the system-to-be will be performing tasks in order to achieve the goals, quality constraints, and satisfy as best as feasible the softgoals. Stakeholder evaluations of requirements — their preferences for some goal (or otherwise) to be satisfied rather than another, or that some must be satisfied, while others are optional — result in relations over requirements (cf., §II-B) subsequently used to compare candidate solutions to the requirements problem.

To solve the requirements problem, it is necessary that the categorized requirements be recorded, refined, expanded by iteratively acquiring new ones. To record requirements, Techne maps statements to labels, thereby sorting them. Let p, q, r (indexed or primed as needed) refer to statements, and $\mathbf{g}()$, $\mathbf{q}()$, $\mathbf{s}()$, $\mathbf{t}()$, and $\mathbf{k}()$ are labels for, respectively, goals, quality constraints, softgoals, tasks, and domain assumptions. A labeling function simply follows the rules of CORE recalled above: if p is an instance of goal, then we write $\mathbf{g}(p)$, if q is an instance of quality constraint, we write $\mathbf{q}(q)$, and so on. Hereafter, *requirement* is synonym for any labeled statement.

B. Relation

There are five relations on requirements in Techne: (i) *inference*, (ii) *conflict*, (iii) *preference*, (iv) *is-mandatory*, and (v) *is-optional* relations. The first two are used to describe and distinguish between candidate solutions, the last three to compare candidate solutions.

1) Inference: When a requirement is the immediate consequence of another set of requirements, the former is called the conclusion, the latter the premises, and they stand related through the inference relation. Say there are two goals, $\mathbf{g}(r_1)$ and $\mathbf{g}(r_2)$, with r_1 for "Music plays in

a player integrated in the web page" and r_2 for "Player has all standard functionalities for listening music". If there is also a domain assumption $\mathbf{k}(\gamma_1)$, with γ_1 for "If r_1 and r_3 then music is delivered to clients via an online audio player", we can conclude the goal $\mathbf{g}(r_3)$, with r_3 for "Deliver music to clients via an online audio player". From two goals and an assumption stating a conditional, the conclusion is another goal. Reading this backwards, from $\mathbf{g}(r_3)$ to the three premises, resemblance to refinement becomes clear: the inference relation can be used to connect the refined requirement to the requirements that refine it. The refinement of a goal by other goals has been a salient feature of KAOS, while other RMLs had their own proxies (e.g., task decomposition in *i**) of the refinement relation. The intuitive meaning of these relations is that if the set of more precise requirements is satisfied, then the less precise requirements are assumed satisfied. Techne considers that, say, goal refinement and task decomposition ask basically the same question: What more precise requirements should be satisfied in order to assume that the less precise — refined, decomposed — requirement is satisfied as well? Instead of relating less precise to more precise requirements by a refinement or decomposition relation, Techne generalizes these via the inference relation. Note that in both these cases the form of the rules $\mathbf{k}(\phi)$ is a *definite Horn clause*.

2) Conflict: Contradictory/inconsistent requirements cannot be in the same candidate solution, or equivalently, are in conflict. The conflict relation stands between all members (two or more) of a minimally inconsistent set of requirements. That a candidate solution should be conflict-free means that conflict relations play a crucial role in distinguishing between consistent sets of requirements, and if these sets satisfy some additional properties, in distinguishing between candidate solutions. To say that n requirements are in direct conflict, another piece of information is needed, namely an implication which explicitly states that if these requirements together hold, then they imply an inconsistency: e.g., to say that $\mathbf{g}(r_1)$ and $\mathbf{k}(r_4)$ are in conflict, where r_4 is for "The user cannot download the audio files", it is necessary to say that the two are contradictory, which is done via an assumption: e.g., $\mathbf{k}(\gamma_2)$, with γ_2 for "$\mathbf{g}(r_1)$ and $\mathbf{k}(r_4)$ are contradictory".

3) Preference: Stakeholder evaluations of requirements convey that not all requirements are equally desirable. E.g., perhaps "The bitrate of music delivered via the online audio player should be at least 256kb/s" is strictly preferred to "The bitrate of music delivered via the online audio player should be at least 128kb/s". If a requirement is strictly more desirable than another one, then there is a preference relation between them and by *strictly*, we mean that they cannot be equally desirable.

4) Is-mandatory: Evaluation is not only comparative, as in the case of preference: individual requirements can be qualified in terms of desirability regardless of other requirements. The is-mandatory relation on a requirement indicates that the requirement *must* be satisfied, or equivalently, that a conflict-free set of requirements which does not include that requirement cannot be a candidate solution. If $\mathbf{k}(r_4)$ is mandatory, then every candidate solution will include it, and exclude all requirements contradicting $\mathbf{k}(r_4)$ (because a candidate solution cannot include conflicts).

5) Is-optional: In contrast to the is-mandatory relation, the is-optional relation on a requirement indicates that it would be desirable for a conflict-free set of requirements to include that requirement, but that set can still be a candidate solution if it fails to include the optional requirement; e.g., if $\mathbf{k}(r_4)$ is optional, then a conflict-free set of requirements which does not contain $\mathbf{k}(r_4)$ can still be a candidate solution. Stated otherwise, if there are two candidate solutions which *differ only* in that one has an optional requirement and the other not, then the former is strictly more desirable than the latter.

C. Modeling

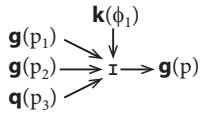
Requirements and relations between them are recorded in graphs called *r-nets*. Each requirement and each relation obtains its own node in an r-net, while edges are unlabeled and directed, having contextual informal interpretation: how one reads/calls an edge depends on which requirements and relations it connects (see below). Note already that, as an r-net contains all requirements and all relations for a system-to-be: the r-net thus defines the requirements problem for a given system-to-be, and includes all (if any) candidate solutions to the problem, so that it is by the analysis of the r-net that candidate solutions are sought (cf., §II-D).

1) Modeling Inference: To show an inference relation, put in the r-net a node (`I`) for the inference relation, then a line from every premise requirement node to `I`, and a line from `I` to the conclusion requirement node.

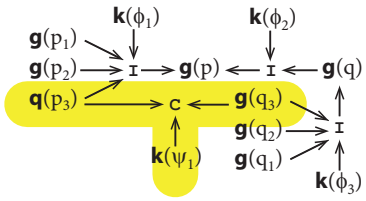
Example II.1. Assume the aim is to build a system that would deliver music on-demand: a user visits a website, chooses songs from a database, and can play them in the audio player on the website. Let $\mathbf{g}(p)$, with p for "Generate revenue from the audio player". We can refine it with two goals and a quality constraint: $\mathbf{g}(p_1)$, $\mathbf{g}(p_2)$ and $\mathbf{q}(p_3)$, where p_1 is for "Display text ads in the audio player", p_2 for "Target text ads according to users' profiles" and p_3 for "Maintain the player free to all users". To conclude $\mathbf{g}(p)$ from $\mathbf{g}(p_1)$, $\mathbf{g}(p_2)$ and $\mathbf{q}(p_3)$, we need to assume that $\mathbf{k}(\phi_1)$, with ϕ_1 for "if $\mathbf{g}(p)$ from $\mathbf{g}(p_1)$, $\mathbf{g}(p_2)$ and $\mathbf{q}(p_3)$, then $\mathbf{g}(p)$". Figure 1(a) shows the r-net with this refinement. ■

2) Modeling Conflict: To show a conflict between requirements, put a node for each one of the conflicting requirements in the r-net, a conflict node (`C`), and a line from every requirement node in the conflicting set to the conflict node.

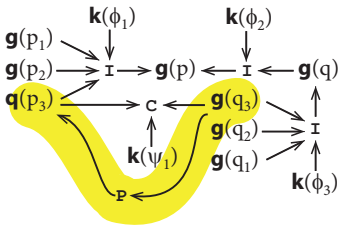
Example II.2. (Contd. Example II.1) We start with $\mathbf{g}(q)$ with q for "Charge subscription to users", and add $\mathbf{k}(\phi_2)$, with ϕ_2 for "if $\mathbf{g}(q)$ then $\mathbf{g}(p)$". We then refine $\mathbf{g}(q)$ onto $\mathbf{g}(q_1)$,



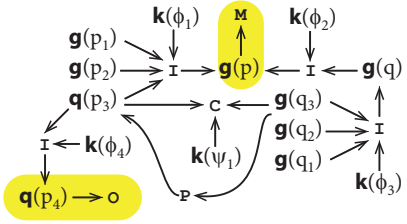
(a) Inference (I) in Example II.1.



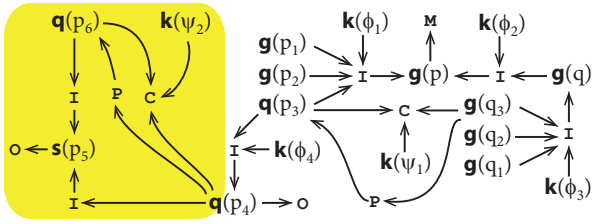
(b) Conflict (C) is highlighted, from Example II.2.



(c) Preference (P) is highlighted, from Example II.3.



(d) Is-mandatory (M) and is-optional (O) are highlighted, from Example II.4.



(e) Softgoal approximation is highlighted (Example II.5).

p	Generate revenue from the audio player.
p1	Display text ads in the audio player.
p2	Target text ads according to users' profiles.
p3	Maintain the player free to all users.
p4	Listen to music in an average of no more than three clicks.
p5	It is easy for new users to access audio content.
p6	An average of ten clicks are needed to a new user to get to audio content.
q	Charge subscription to users.
q1	Music database is restricted to subscribers.
q2	Users can subscribe.
q3	Music player is available to subscribers only.

(f) Symbols and corresponding statements used in Figures 1(a)-1(e).

Figure 1. R-nets from Examples II.1–II.5.

$\mathbf{g}(q_2)$, and $\mathbf{g}(q_3)$, with q_1 for "Music database is restricted to subscribers", q_2 for "Users can subscribe" and q_3 for "Music player is available to subscribers only". This requires the assumption $\mathbf{k}(\phi_3)$, ϕ_3 for "If $\mathbf{g}(q_1)$, $\mathbf{g}(q_2)$, and $\mathbf{g}(q_3)$, then $\mathbf{g}(q)$". It appears that "we cannot both maintain the player free to all users ($\mathbf{q}(p_3)$) and make music available to subscribers only ($\mathbf{g}(q_3)$)" which ψ_1 abbreviates, so we add $\mathbf{k}(\psi_1)$. We thereby have the conflict between $\mathbf{q}(p_3)$ and $\mathbf{g}(q_3)$, shown in Figure 1(b). ■

3) Modeling Preferences: Preference is a binary relation: if a requirement x is preferred to requirement y, add a preference node (P), and draw a line from the preferred requirement (x) to the preference node (P), and from the preference node (P) to the less preferred requirement (y).

Example II.3. (Contd. Example II.2) The r-net in Figure 1(b) includes two refinements of $\mathbf{g}(p)$. The conflict C indicates that these are two *alternative* refinements, as they cannot appear together in a candidate solution. The preference in Figure 1(c) says that $\mathbf{g}(q_3)$ is strictly preferred to $\mathbf{q}(p_3)$. This preference becomes one (of potentially many) criteria for the comparison of candidate solutions: if this were the only criterion, then we would choose the candidate solution which includes $\mathbf{g}(q_3)$ instead of another which includes $\mathbf{q}(p_3)$. ■

4) Modeling Mandatory and Optional Relations: Both the is-mandatory and is-optional relations are unary. To say in an r-net that a requirement is mandatory, add a node (M) for the is-mandatory relation, and a line from the requirement node to the is-mandatory node. To state instead that a requirement is optional, add a node (O) for the is-optional relation, and a line from the requirement node to the is-optional node.

Example II.4. (Contd. Example II.3) If every solution must include $\mathbf{g}(p)$, then we add the node M to the r-net in Figure 1(c) and a line from $\mathbf{g}(p)$ to M, as shown in Figure 1(d).

To illustrate the use of the is-optional relation, suppose that maintaining the player free to all users ($\mathbf{q}(p_3)$) will allow new users to listen to music in an average of no more than three clicks through the audio service (as they do not need to register or provide their billing details); we denote $\mathbf{q}(p_4)$ the latter quality constraint. We add $\mathbf{q}(p_4)$ as a node to the r-net, along with the assumption $\mathbf{k}(\phi_4)$, with ϕ_4 for "if $\mathbf{q}(p_3)$, then $\mathbf{q}(p_4)$", and thus an inference relation. Let $\mathbf{q}(p_4)$ be optional: to make it so in the r-net, we connect it to the node O. If we consider the r-net in Figure 1(d), it is no longer obvious which of the two refinements is more desirable than the other: if a candidate solution includes $\mathbf{g}(q_3)$, then it will not contain $\mathbf{q}(p_4)$, but will have the preferred $\mathbf{g}(q_3)$; if a candidate solution includes $\mathbf{q}(p_3)$, then it will have $\mathbf{q}(p_4)$, but not the preferred $\mathbf{g}(q_3)$. ■

5) Softgoal Approximation: As softgoals vaguely constrain values of properties that are not necessarily directly measurable, every softgoal ought to be approximated in an r-net. A set of requirements can be an *approximation*

of a softgoal if it is assumed that, once its members are satisfied, the softgoal will be satisfied to some extent. As different approximations may satisfy the same softgoal to different extents, preference relations can be added between the members of different approximations. These preferences let us compare approximations in terms of how well one satisfies the softgoal relative to others.

Example II.5. (Contd. Example II.4) We introduce the optional softgoal $\mathbf{s}(p_5)$, with p_5 for "It is easy for new users to access audio content" into the r-net from Figure 1(d). There are no universal criteria that tell us what "easy" precisely means in the context of this system-to-be. There are thus different ways to approximate $\mathbf{s}(p_5)$. One of them is to say that the smaller the average number of clicks needed to a new user to access audio content (computed over some number of sessions and for a given focus group), the easier it is to access that content. We can introduce at least two quality constraints, one being $\mathbf{q}(p_4)$ and another $\mathbf{q}(p_6)$, with p_6 for "An average of ten clicks are needed to a new user to get to audio content", with corresponding assumptions $\mathbf{k}(\phi_5)$ and $\mathbf{k}(\phi_6)$, with ϕ_5 for "if $\mathbf{q}(p_4)$, then $\mathbf{s}(p_5)$" and ϕ_6 for "if $\mathbf{q}(p_6)$, then $\mathbf{s}(p_5)$". A preference is added, to indicate that the approximation by $\mathbf{q}(p_4)$ is preferred to the approximation by $\mathbf{q}(q_6)$. Finally, we abbreviate "$\mathbf{q}(p_4)$ cannot be satisfied together with $\mathbf{q}(p_6)$" by ψ_2, and add a conflict between $\mathbf{q}(p_4)$ and $\mathbf{q}(p_6)$, along with the assumption $\mathbf{k}(\psi_2)$. ■

6) Modeling with Visual Syntax: Concrete RMLs (e.g., KAOS, *i**) have a visual syntax as a diagrammatic notation that aims to simplify the making and reading of requirements models created with these languages. Techne is an *abstract* RML as it has no visual syntax. To make a concrete RML out of Techne, it would be necessary to add a visual syntax.

Example II.6. As a brief illustration of how this might proceed, consider Figure 2, in which *goals* are shown to be represented by rounded rectangles, while the *refinement relation* is shown as a dark dot, connected to the refined requirement and the requirements refining it. This second case shows how the visual syntax can simplify the model, since $\mathbf{k}(\phi)$ can be inferred in the case of decomposition and need not be drawn. ■

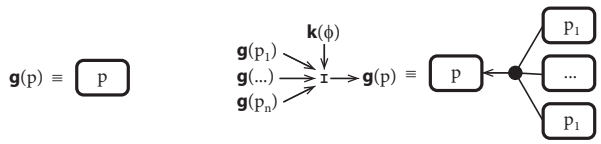


Figure 2. An example of visual syntax for goals and the inference relation.

D. Analysis

Analysis in Techne should answer two questions: given an r-net, (i) What are the candidate solutions to the requirements problem in it? and (ii) What are the preferred and optional requirements that each candidate solution contains? Example II.7 informally presents how these answers are sought; we then look into the formalization of the r-nets towards the automation of analysis.

Example II.7. (Contd. Example II.5) A candidate solution must be conflict-free, so that we are interested in conflict-free subnets of the r-net in Figure 1(e). There are many conflict-free subnets in Figure 1(e): e.g., $\mathbf{g}(p)$ taken alone is a conflict-free subnet, as is the refinement shown in Figure 1(a). Since $\mathbf{g}(p)$ is itself a subnet of the said refinement, we are more interested in the entire refinement than in any one of its subnets alone. Stated otherwise, conflict-free/consistent subnets can be ordered by the subset relation $\subseteq$, and instead of looking for all consistent subnets, those maximal with regards to $\subseteq$ are the most interesting ones. Figures 4(a) and 4(b) highlight two maximal consistent subnets in the r-net from Figure 1(e). These are, however, *not* also candidate solutions to the requirements problem, as each has goals and quality constraints as source nodes (i.e., nodes without incoming lines). Recall that we are interested in finding tasks and domain assumptions which satisfy goals, quality constraints, and softgoals. We can add hypothetical tasks to the r-net in Figure 1(e) so that no source nodes are goals, quality constraints, or softgoals. Figures 4(c) and 4(d) highlight two maximal consistent subnets of the resulting r-net. Each of these is a candidate solution, because (i) neither has goals, quality constraints, or softgoals as source nodes, and (ii) each includes the only mandatory requirement $\mathbf{g}(p)$.

Once we have found the candidate solutions, the question is how do they compare? We can establish that the two candidate solutions, denoted $\mathcal{S}_A$ (the subnet highlighted in Figure 4(c)) and $\mathcal{S}_B$ (the subnet highlighted in Figure 4(d)), have the following mandatory, optional, and preferred nodes:

- $\mathcal{S}_A$ (i) has $\mathbf{q}(p_4)$, is both an optional and a preferred requirement; (ii) has $\mathbf{s}(p_5)$ which is an optional requirement; and (iii) has $\mathbf{g}(p)$ which is a mandatory requirement.
- $\mathcal{S}_B$ (i) has $\mathbf{g}(q_3)$ which is a preferred node; (ii) has $\mathbf{s}(p_5)$, an optional requirement; and (iii) has $\mathbf{g}(p)$, a mandatory requirement.

The following comparison table gives the summary:

	$\mathtt{P} : \mathbf{g}(q_3)$	$\mathtt{P} : \mathbf{q}(p_4)$	$\mathtt{O} : \mathbf{q}(p_4)$	$\mathtt{O} : \mathbf{s}(p_5)$
$\mathcal{S}_A$	no	yes	yes	yes
$\mathcal{S}_B$	yes	no	no	yes

Each column in the comparison table is a criterion for the comparison of candidate solutions. How to rank the candidates is beyond the scope of this paper. ■

Automating the search for candidate solutions requires that the elements of Techne discussed up to this point obtain mathematically formal definitions. To sketch the formalization, recall that a modeling language has four parts:

(i) an alphabet of symbols, (ii) rules of grammar to combine symbols into expressions, (iii) a semantic domain with the objects of interest to the purpose of the language, and (iv) mappings from the symbols and expressions to the objects in the semantic domain. The first and second components are usually called *syntax*, the last two *semantics*.

1) R-net Alphabet: To draw r-nets, we used symbols for (i) atomic statements (indexed/primed p, q, r), (ii) complex statements (Greek letters) (iii) labels (**k**(), **g**(), **q**(), **s**(), **t**()), (iv) relations (**I**, **C**, **P**, **M**, **O**), and (v) arrow-headed lines.

2) R-Net Grammar: Grammar is dictated by the CORE ontology for the use of labels, and the arity of relations for the use of relation symbols and lines. All allowed expressions are shown in Figure 3, and every r-net is exactly the finite set of elements shown in that figure.

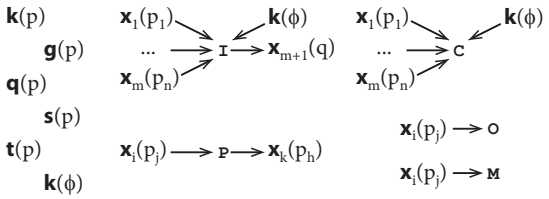
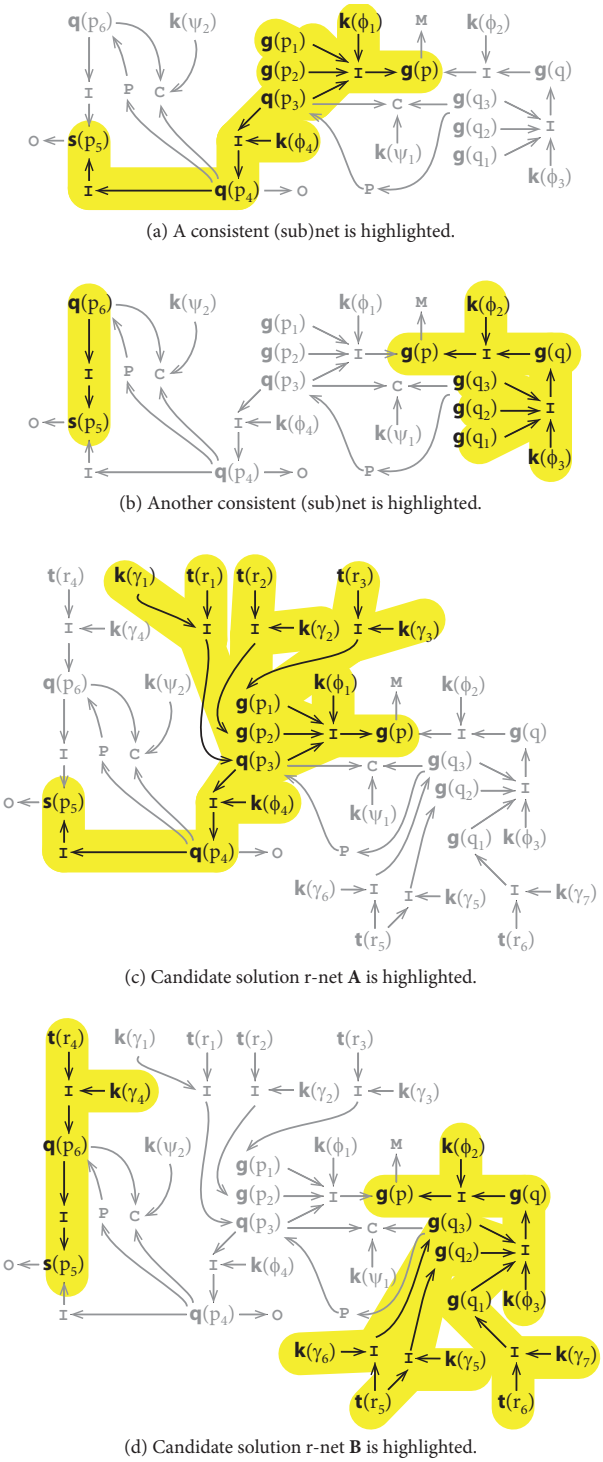


Figure 3. Allowed expressions in an r-net.

In Figure 3, every p, q is an arbitrary atomic statement, every ϕ an arbitrary complex statement, and every **x** an arbitrary label. For **I**, ϕ abbreviates "if $\mathbf{x}_1(p_1)$ and ... and $\mathbf{x}_m(p_n)$, then $\mathbf{x}_{m+1}(q)$"; for **C**, ϕ is for "if $\mathbf{x}_1(p_1)$ and ... and $\mathbf{x}_m(p_n)$, then contradiction". As every complex statement refers to an assumption, it must have the label **k**().

3) Semantic Domain and Mapping: The elementary objects in the semantic domain of r-nets are pieces of information stating the properties of the system-to-be and its operational environment and the inference, conflict, preference, is-mandatory, and is-optional relations between them. Atomic and complex statements in the alphabet refer/map to these pieces of information, relation symbols map to relations, while expressions refer to combinations of the two. Following the statement of the requirements problem and as we want to avoid contradictory solutions, a candidate solution is information which is (i) not contradictory and (ii) from which we can conclude that mandatory goals and quality constraints are satisfied.

4) Proof-Theoretic Characterization of Candidate Solutions: To find candidate solutions, we need to find their counterparts in syntax, that is, those parts of r-nets which map exactly to candidate solutions in the semantic domain. As we will be comparing solutions after we find them, we leave out the information for the comparison of candidate solutions (the preference, is-mandatory, and is-optional relations). In syntax, this means that we focus not on a given r-net, but on its *attitude-free* variant: given an r-net $\mathcal{R}$, to make its attitude-free variant $\bar{\mathcal{R}}$, delete all **P**, **M**, and **O** nodes and all lines entering and leaving from these nodes from $\mathcal{R}$. $\bar{\mathcal{R}}$ contains only the atomic and complex statements, and the inference and conflict relations.



(a) A consistent (sub)net is highlighted.

(b) Another consistent (sub)net is highlighted.

(c) Candidate solution r-net **A** is highlighted.

(d) Candidate solution r-net **B** is highlighted.

Figure 4. Consistent subnets and candidate solution r-nets in Example II.7.

An $\bar{\mathcal{R}}$ can be seen as a set of proofs. To do so, we observe

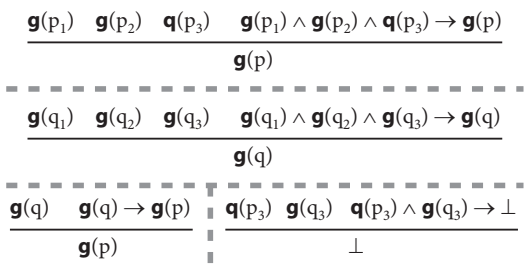


Figure 5. The $\bar{\mathcal{R}}$ from Figure 1(b) rewritten as four proofs.

that our complex statements are sentences in the conditional if-then form in which the fragment after the *if* references requirements, while the one after *then* references either a single requirement, or contradiction (cf., Examples II.1–II.5). We rewrite every complex statement ϕ in $\mathbf{k}(\phi)$ as a formula with conjunction and implication: every ϕ is such that either $\phi \equiv \bigwedge_{i=1}^{n} pl_i \rightarrow pl$, or $\phi \equiv \bigwedge_{i=1}^{n} pl_i \rightarrow \bot$, where every pl is some requirement (e.g., $pl \equiv \mathbf{g}(q)$) and $\bot$ refers to logical inconsistency. Figure 5 shows the sentences obtained by applying the said rules on the $\bar{\mathcal{R}}$ in Figure 1(b).

Attitude-free r-nets are sets of proofs of the formal system in which the atoms pl of the alphabet are symbols for requirements (e.g., $\mathbf{g}(p)$), the only allowed expressions are $\bigwedge_{i=1}^{n} pl_i \rightarrow pl$ and $\bigwedge_{i=1}^{n} pl_i \rightarrow \bot$, and the only rule of inference is modus ponens. Given a set of such requirements and expressions denoted $\bar{S}$ and $x \in \{pl, \bot\}$:

1) $\bar{S} \mid\!\sim_{\mathcal{T}} pl$ if $pl \in \bar{S}$, or
2) $\bar{S} \mid\!\sim_{\mathcal{T}} x$ if $\forall 1 \leq i \leq n,\ \bar{S} \mid\!\sim_{\mathcal{T}} pl_i$ and $\mathbf{k}(\bigwedge_{i=1}^{n} pl_i \rightarrow x) \in \bar{S}$.

The consequence relation $\mid\!\sim_{\mathcal{T}}$ is sound w.r.t. standard entailment in propositional logic, but is incomplete in two ways: it only considers deducing positive atoms, and no ordinary proofs based on arguing by contradiction go through, thus being paraconsistent.

The consequence relation leads us to the following conception of the *candidate solution* concept: Given an $\mathcal{R}$ with all of its domain assumptions in the set **K**, tasks in **T**, goals in **G**, quality constraints in **Q**, and softgoals in **S**, *a set of tasks* $\boldsymbol{T}^*$ *and a set of domain assumptions* $\boldsymbol{K}^*$ *are a* ***candidate solution*** *to the requirements problem of* $\mathcal{R}$ *if and only if (i)* $\boldsymbol{K}^*$ *and* $\boldsymbol{T}^*$ *are not inconsistent, (ii)* $\boldsymbol{K}^*, \boldsymbol{T}^* \mid\!\sim_{\mathcal{T}} \boldsymbol{G}^*, \boldsymbol{Q}^*$*, where* $\boldsymbol{G}^* \subseteq \boldsymbol{G}$ *and* $\boldsymbol{Q}^* \subseteq \boldsymbol{Q}$*, (iii)* $\boldsymbol{G}^*$ *and* $\boldsymbol{Q}^*$ *include, respectively, all mandatory goals and quality constraints, and (iv) all mandatory softgoals are approximated by the consequences of* $\boldsymbol{K}^* \cup \boldsymbol{T}^*$*, so that* $\boldsymbol{K}^*, \boldsymbol{T}^* \mid\!\sim_{\mathcal{T}} \boldsymbol{S}^{\boldsymbol{M}}$*, where* $\boldsymbol{S}^{\boldsymbol{M}}$ *is the set of mandatory softgoals.*

The candidate solution concept leads us in turn to a more precise formulation of the requirements problem: ***Given*** *an r-net* $\mathcal{R}$*,* ***find*** *its candidate solutions.* Once candidates are found, the comparison table can be constructed in the straightforward way so that they can be compared. It is beyond the scope of this paper to give guidelines on how to rank candidates

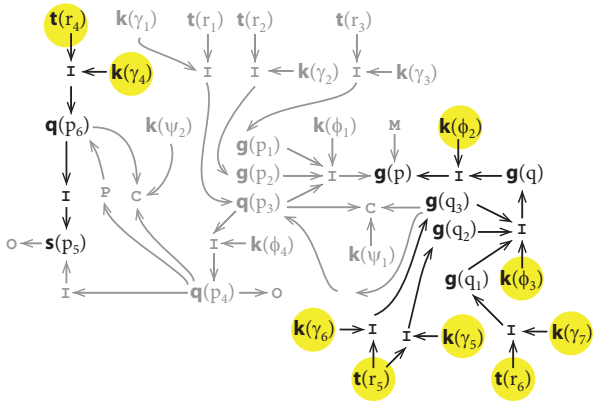


Figure 6. Members of $\mathbf{T}^*$ and $\mathbf{K}^*$.

on the basis of the comparison table. Figure 6 highlights the members of these two sets and thus a candidate solution for the r-net from Example II.7. Note that Figures 4(c) and 4(d) highlight candidate solutions and all of their consequences. We give elsewhere [14] further details on the formalization of r-nets and candidate solutions.

A note on expressiveness: observe that if we treat the *pl*s as atomic propositions, then an r-net is a Horn theory (every formula has at most one positive atom), which is known to be less expressive than full propositional logic, let alone predicate logic. Among others, there is no provision for world knowledge that is disjunctive (e.g., composite *pl* like $p \vee q$), but we can express exclusive disjunction (e.g., in Figure 1(b), $\mathbf{g}(p)$ is refined by either $\mathbf{g}(q)$ or by the conjunction of $\mathbf{g}(p_1)$, $\mathbf{g}(p_2)$, and $\mathbf{g}(p_3)$). There is also no provision for inference nodes that might use lemmas as $\mathbf{k}(\phi)$, which might lead to case-based reasoning. On the other hand, if we consider only attitude-free r-nets, the problem of finding candidate solutions can be reduced to variants of solving non-standard reasoning problems in logic, such as abduction ("What tasks are needed to ensure the mandatory goals?"). Interestingly, it is known that Horn abduction is one level lower in the polynomial complexity hierarchy than abduction with full propositional logic, so our version of Techne has lower expected computational cost — a typical expressiveness/complexity tradeoff. Only extensive practical experience in modeling will show whether more expressive power is needed.

III. Related Work

Surveys of RE research — from van Lamsweerde [21] and Robinson et al. [19] in particular — confirm Zave and Jackson's [29] prior observation that the field had already in the 1980s left behind simplistic approaches to understanding what a system-to-be would do in favor of novel and varied terminology, methods, languages, tools, and issues considered to be critical. One constant is the observation that RMLs play a central role in both research and practice of RE. It

does not require much knowledge of the field to see that many research efforts that fall within Zave's classification [28] relate in one way or another to one or more RMLs; e.g., elicitation of information from stakeholders, validation, specification, checks for incompleteness and inconsistency, all suppose that some model of requirements is available.

Despite the important position that RMLs play in RE, there are no widely-accepted and precise standards that a formalism must satisfy in order to be called an RML. The evolution of RMLs seems to be one of testing of and converging on similar ideas (e.g., we find refinement in one way or another in most RMLs), rather then the design of formalisms following clear desiderata. We highlight some of these key ideas in the rest of this section and position Techne in relation to them.

Original RML: Highly developed languages for the specification of the properties of a system-to-be — i.e., formal methods such as Z, VDM, Larch, temporal logic, CSP, transition axioms, among others (e.g., [4], [25]) — have been available alongside most RMLs, and they have been used to perform some of the tasks of RMLs. That there is more to writing requirements than functional specification was recognized in the original RML [10] (hereafter ORML), "a notation for requirements modeling which combines object-orientation and organization, with an assertional sublanguage used to specify constraints and deductive rules" [9]. Formal semantics is given to ORML via a mapping from its descriptions to assertions in first order logic (hereafter FOL). One thereby obtains facilities for the structuring and organization of FOL theories. The ontology in ORML distinguishes between entities, activities, and assertions. The ontology was judged limited [9] and responses to limitations went in two directions. RMLs such as KAOS and *i** took the direction in which the ontology remains fixed (i.e., one cannot add or remove concepts when applying the RML) but include more concepts, designed to cover concerns such as the desires of the system's stakeholders (see below). The other direction was adopted in Telos [18] and consists of leaving the ontology undefined, while having in the language the facilities needed to define the ontology. The second approach is more expressive, but its abstraction makes it difficult to provide methodological guidance which can be given when a fixed set of concepts is known and manipulated every time the language is used.

KAOS: "The overall approach taken in KAOS has three components: (i) a conceptual model for acquiring and structuring requirements models, with an associated acquisition language, (ii) a set of strategies for elaborating requirements models in this framework, and (iii) an automated assistant to provide guidance in the acquisition process according to such strategies." [5] The conceptual model specifies the ontology in KAOS, which includes a considerable number of concepts (object, operation, agent, goal, obstacle, requisite/requirement/assumption, scenario) and relations (specialization, refinement, conflict, operationalization, concern, and so on) [5], [22], [23]. A KAOS model of requirements instantiates the concepts, relates these instances, declares instances' properties which are relevant to the elaboration/transformation of the model, and allows the engineer to formally define the instance as a theory of linear temporal FOL. In light of ORML and specification languages, KAOS can be understood as an RE *methodology* (i.e., a combination of an RML and of methods for the use of that RML) which is defined on top of linear temporal FOL that serves as a specification language in KAOS.

I-Star (i):* *i** [26], [27] is an RML that distinguishes itself strongly from those mentioned above both in its design and its focus. In terms of design, *i** is not defined on top of a specification language. The focus of *i** is on the interdependencies of actors within a socio-technical system, their individual and joint goals, tasks, and available or necessary resources, the roles they occupy. A model of requirements made with *i** aims to be a snapshot of the intentional states of actors, along with what roles they adopt, and how they depend on each other for the satisfaction of individual and joint goals, the performance of tasks, and use of resources. The system-to-be or its components are actors alongside individuals and groups. In contrast to both ORML and KAOS, the engineer cannot formally verify the satisfaction of requirements (i.e., check if a system's properties satisfy goals [21]) via an *i** requirements model; the closest the engineer can do is validate them instead via informal discussion with the stakeholders. It is perhaps this departure from specification languages as foundations for RMLs that led to considerable work on *i**. It is a lightweight RML, the non-formal character of which makes it easy to learn, a critical feature given that requirements must be validated by stakeholders who cannot be expected to manipulate artifacts produced with specification languages.

Tropos and Formal Tropos: Tropos [3], a methodology for information systems engineering, uses *i** as its RML at the very first steps of the RE process, when it is impractical to start writing formal theories in a variant of FOL or another formalism. Once *i** models of the system-to-be within its organizational environment are available, Tropos explains how to proceed towards data and behavior models of the system-to-be. Formal Tropos [7] continued the tradition of giving formal semantics to RMLs by mapping instances of *i** concepts and relations between them (i.e., *i** requirements models) to theories of linear temporal FOL. Formal Tropos argued "that formal analysis techniques are useful during early development phases. Novelty lies in extending model checking techniques — which rely mostly on design-inspired specification languages — so that they can be used for early requirements modeling and analysis" [7].

IV. Discussion & Limitations

Techne is an abstract RML intended to be used as the starting point for the definition of new RMLs applicable to the early phase of the RE process, when the requirements

problem for the system-to-be and its candidate solutions are still unclear, and before one candidate solution is singled out. If an RML is made from Techne, the RML can help the structuring of the requirements problem and preliminary identification of candidate solutions thereto, as well as of the criteria for the comparison of the candidates.

RMLs made from Techne are bound to be quite different from ORML, KAOS, and Formal Tropos. ORML obtains formal semantics via the mapping of its models/descriptions to FOL. In KAOS requirements models, formal definitions of concept instances have formal semantics via their writing in linear temporal FOL. In Formal Tropos, instances of *i** concepts are — similarly to KAOS — defined in linear temporal FOL. ORML and KAOS are object-oriented, featuring the specialization relation. Techne is not object-oriented and does not incorporate the specialization relation. It cannot model conditional preferences. Atoms in Techne are propositions, and given the purpose of Techne, these propositions are likely to be written as sentences of natural language. Techne supports neither the definition of temporal constraints, nor task sequencing, nor can it distinguish between domain assumptions which are facts (e.g., laws of nature) from those which are open to debate. Emphasis is on straightforward knowledge representation and its use towards the identification of candidate solutions. That we limit expressions to being either of the form $\bigwedge_{i=1}^{n} pl_i \rightarrow pl$ or $\bigwedge_{i=1}^{n} pl_i \rightarrow \bot$, means that we cannot state logical disjunction in Techne. As we have shown throughout the paper, Techne can, however, represent AND/OR graphs.

Techne and *i** differ in several respects. *i** has no notion of conflict, preference or mandatory/optional requirements, no formal semantics, and thus has no precise notion of what a candidate solution to the requirements problem is. Alternative decompositions of a goal are compared in terms of their contributions to softgoals. Techne keeps softgoals, but due to the vagueness of softgoal instances [15], [16] we require that they are approximated, i.e., "refined" by other non-softgoals, among which preference relations can be added to indicate which satisfy the softgoal in more desirable ways than others. Techne includes no concepts pertaining to actors and roles.

Giorgini et al. [8] recognized the need to formalize goal models so as to automatically evaluate the (degree of) satisfaction of goals. Their goal models are AND/OR graphs, in which nodes are goals, and a number of relations is provided to indicate if the interaction is positive or negative (i.e., how the satisfaction of a goal influences the satisfaction of the other goal related to it), as well as to specify the strength of the interaction. Techne uses preferences to indicate in the relative degrees of satisfaction (cf., Example II.5), while quantitative estimates of satisfaction levels are not used. Goal models from Giorgini et al. do not incorporate the notion of conflict as inconsistency, they do not include concepts other than goals, cannot distinguish optional from mandatory requirements, and have no notion of robust solutions [14].

Techne's handling of inconsistency is similar in aim to Hunter & Nuseibeh's, who are interested in reasoning on an inconsistent specification and "keeping track of deductions made during reasoning, and deciding what actions to perform in the presence of inconsistencies" [12, pp.363–364], while avoiding trivial deductions from inconsistencies. An r-net in Techne keeps track of *all* the deductions made and inconsistencies (conflicts) are identified. A significant difference is that their work is based on clausal resolution, which may lead to $\bot$ being derived, but this is prevented from leading to irrelevant formulas being inferred. In contrast, Techne addresses directly the identification of maximally consistent subnets, from which $\bot$ cannot be derived.

Techne by its very design avoids asking stakeholders for quantitative estimates of preference, in contrast to, e.g., the approach from Liaskos et al. [17]. Preferences are binary relations, and two preferences cannot be compared in an r-net itself, but only after the comparison table is constructed. Techne thereby recognizes that there are different approaches to decision-making in the presence of multiple criteria and no ideal decision rules, leaving it to the designer who makes a new RML from Techne to choose herself the decision rules to apply on the comparison table. It is not possible in Techne to indicate the rationale for a preference in an r-net, as preference nodes cannot appear in domain assumptions.

Techne can be used as the foundation for new CASE tools, which either define a visual syntax on top of the Techne backend, or let the user define their own visual syntax. It is important to note that an RML made from Techne, one with an appropriately designed visual syntax, will require its users to know very little, if anything about Techne itself: they will write models using the visual syntax, the models will have a Techne form which can then be processed by the algorithms that can find candidate solutions.

We noted earlier (cf., §II-C6) how a visual syntax can be defined over Techne. The designer of the RML may also choose to add new concepts and/or relations on top of Techne. That they are added "on top" means that every added concept and relation has no counterpart in the Techne formalization: e.g., one can add the concept of *actor* on top of Techne by defining an actor as a subnet of the r-net; if so, there is no need to add notions to the formalization of Techne. In case new concepts/relations require extending Techne itself, symbols for the new notions must be added to Techne's alphabet, their role in grammar, the semantic domain, and the semantic mappings explained, and finally, effects of the new notions on the analysis of r-nets must be discussed.

V. Conclusions

Techne serves to describe requirements problems and find criteria to compare candidate solutions early on in the RE process. Choosing a candidate is a separate issue, one of decision-making in the presence of multiple criteria and is beyond the scope of this paper. Features such

as object-orientation, predicates, temporal constraints, and task sequencing are absent, as they serve for the detailed specification of the chosen candidate solution. Techne thus precedes and complements RMLs for detailed specification which do include such features.

More can be said than we did here on how to analyze preferences and optional requirements in r-nets, as well as on how to manipulate the information in the comparison table in order to inform the choice of a solution. Further ongoing work on Techne focuses on the definition and testing of efficient reasoning methods for the search of solutions in r-nets, on tool support for modeling and reasoning, and on guidelines for how to make new RMLs from Techne.

REFERENCES

[1] J.-R. Abrial, *The B-book: assigning programs to meanings*. New York, NY, USA: Cambridge Univ. Press, 1996.

[2] H. Bekic, D. Bjørner, W. Henhapl, C. B. Jones, and P. Lucas, "On the Formal Definition of a PL/I Subset (Selected parts)," in *Programming Languages and Their Definition - Hans Bekic (1936-1982)*. Springer-Verlag, 1984, pp. 107–155.

[3] J. Castro, M. Kolp, and J. Mylopoulos, "Towards requirements-driven information systems engineering: the Tropos project," *Inf. Syst.*, vol. 27, no. 6, pp. 365–389, 2002.

[4] E. M. Clarke and J. M. Wing, "Formal methods: state of the art and future directions," *ACM Comput. Surv.*, vol. 28, no. 4, pp. 626–643, 1996.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, no. 1-2, pp. 3–50, 1993.

[6] E. Dubois, J. Hagelstein, and A. Rifaut, "Formal Requirements Engineering with ERAE," *Philips Journal of Research*, vol. 43, no. 3/4, pp. 393–414, 1988.

[7] A. Fuxman, L. Liu, J. Mylopoulos, M. Roveri, and P. Traverso, "Specifying and analyzing early requirements in Tropos," *Requirements Eng.*, vol. 9, no. 2, pp. 132–150, 2004.

[8] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal reasoning techniques for goal models," *J. Data Semantics*, vol. 1, pp. 1–20, 2003.

[9] S. Greenspan, J. Mylopoulos, and A. Borgida, "On formal requirements modeling languages: RML revisited," in *Proc. 16th Int. Conf. Software Eng.*, 1994, pp. 135–147.

[10] S. J. Greenspan, A. Borgida, and J. Mylopoulos, "A requirements modeling language and its logic," *Inf. Syst.*, vol. 11, no. 1, pp. 9–23, 1986.

[11] S. J. Greenspan, J. Mylopoulos, and A. Borgida, "Capturing more world knowledge in the requirements specification," in *Proc. 6th Int. Conf. Software Eng.*, 1982, pp. 225–234.

[12] A. Hunter and B. Nuseibeh, "Managing inconsistent specifications: Reasoning, analysis, and action," *ACM Trans. Softw. Eng. Methodol.*, vol. 7, no. 4, pp. 335–367, 1998.

[13] D. Jackson, "Boolean Compilation of Relational Specifications," MIT, Tech. Rep. MIT-LCS-735, 1997.

[14] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne," 2010, CSRG-606, Computer Systems Research Group Technical Report, University of Toronto. ftp://ftp.cs.toronto.edu/pub/reports/csri/606/.

[15] I. J. Jureta, S. Faulkner, and P.-Y. Schobbens, "A more expressive softgoal conceptualization for quality requirements analysis," in *Proc. 25th Int. Conf. Conceptual Modelling*, 2006, pp. 281–295.

[16] I. J. Jureta, J. Mylopoulos, and S. Faulkner, "Revisiting the core ontology and problem in requirements engineering," in *16th IEEE Int. Requirements Eng. Conf.*, 2008, pp. 71–80.

[17] S. Liaskos, S. McIlraith, and J. Mylopoulos, "Goal-based Preference Specification for Requirements Engineering," in *18th IEEE Int. Requirements Engineering Conf.*, 2010.

[18] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: representing knowledge about information systems," *ACM Trans. Inf. Syst.*, vol. 8, no. 4, pp. 325–362, 1990.

[19] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 132–190, 2003.

[20] J. M. Spivey, *Introducing Z: A Specification Language and Its Formal Semantics*. Cambridge Univ. Press, 1988.

[21] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proc. 5th IEEE Int. Symposium on Requirements Eng.*, 2001, p. 249.

[22] A. van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," *IEEE Trans. Software Eng.*, vol. 24, no. 11, pp. 908–926, 1998.

[23] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Trans. Software Eng.*, vol. 26, no. 10, pp. 978–1005, 2000.

[24] J. M. Wing, "Writing Larch interface language specifications," *ACM Trans. Program. Lang. Syst.*, vol. 9, no. 1, pp. 1–24, 1987.

[25] ——, "A specifier's introduction to formal methods," *IEEE Computer*, vol. 23, no. 9, pp. 8–24, 1990.

[26] E. Yu, "Towards modeling and reasoning support for early requirements engineering," in *Proc. 3rd IEEE Int. Symposium on Requirements Eng.*, 1997, pp. 226–235.

[27] E. S. K. Yu and J. Mylopoulos, "Understanding "Why" in Software Process Modelling, Analysis, and Design," in *Proc. 16th Int. Conf. Software Eng.*, 1994, pp. 159–168.

[28] P. Zave, "Classification of research efforts in requirements engineering," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 315–321, 1997.

[29] P. Zave and M. Jackson, "Four dark corners of requirements engineering," *ACM T. Softw. Eng. Methodol.*, vol. 6, no. 1, pp. 1–30, 1997.