Requirements Engineering (2021) 26:1–23
https://doi.org/10.1007/s00766-020-00329-x

ORIGINAL ARTICLE

A retrospective on Telos as a metamodeling language for requirements engineering

Manolis Koubarakis[1] · Alexander Borgida[2] · Panos Constantopoulos[3] · Martin Doerr[4] · Matthias Jarke[5] · Manfred A. Jeusfeld[6] · John Mylopoulos[7] · Dimitris Plexousakis[4]

Received: 15 May 2018 / Accepted: 2 March 2020 / Published online: 12 March 2020

Abstract
Telos is a conceptual modeling language intended to capture software knowledge, such as software system requirements, domain knowledge, architectures, design decisions and more. To accomplish this, Telos was designed to be extensible in the sense that the concepts used to capture software knowledge can be defined in the language itself, instead of being built-in. This extensibility is accomplished through powerful metamodeling features, which proved very useful for interrelating heterogeneous models from requirements, model-driven software engineering, data integration, ontology engineering, cultural informatics and education. We trace the evolution of ideas and research results in the Telos project from its origins in the late eighties. Our account looks at the semantics of Telos, its various implementations and its applications. We also recount related research by other groups and the cross-influences of ideas thereof. We conclude with lessons learnt.

Keywords Metamodeling · Conceptual modeling · Knowledge representation · Software engineering · Requirements modeling · Semantic networks · RDF · Cultural informatics

1 Introduction

Telos is a conceptual modeling language intended to capture software-related knowledge, such as requirements, architectural design, design rationale, evolution history and domain knowledge. To capture such disparate kinds of knowledge, Telos was designed to be extensible in the sense that one can define in the language the notions used to model the various topics. For example, one can define the concepts of entity and activity as metaclasses and then use these to build SADT models with formal semantics. After four years of language development, Telos was presented in 1990 [59]. It has seen several implementations and it is still in use.

✉ Manolis Koubarakis
koubarak@di.uoa.gr

Alexander Borgida
borgida@cs.rutgers.edu

Panos Constantopoulos
panosc@aueb.gr

Martin Doerr
martin@ics.forth.gr

Matthias Jarke
jarke@informatik.rwth-aachen.de

Manfred A. Jeusfeld
Manfred.Jeusfeld@his.se

John Mylopoulos
jm@cs.toronto.edu

Dimitris Plexousakis
dp@ics.forth.gr

[1] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens University Campus, 15784 Ilisia, Athens, Greece

[2] Department of Computer Science, Rutgers University, Piscataway, NJ 08855, USA

[3] Department of Informatics, Athens University of Economic and Business, 76 Patission Street, 10434 Athens, Greece

[4] Institute of Computer Science, Foundation for Research and Technology - Hellas, Nikolaou Plastira 100, Vassilika Vouton, 70013 Heraklion, Crete, Greece

[5] RWTH Aachen, Lehrstuhl Informatik 5, Ahornstr. 55, 52056 Aachen, Germany

[6] School of Informatics (IIT), University of Skövde, Box 408, 54128 Skövde, Sweden

[7] School of Electrical Engineering and Computer Science (EECS), University of Ottawa, 800 King Edward Ave., Ottawa, ON K1N 6N5, Canada

The research baseline for Telos was the requirements modeling language RML, which adopted the primitive concepts of SADT [117], but was formal, adopting ideas from knowledge representation languages in artificial intelligence (AI), specifically semantic networks. However, RML was rather impoverished with respect to the concepts it offered (just activities and entities, called data in SADT), and more variety would be needed to capture requirements but also other software-related knowledge, such as domain knowledge, designs and design rationale. Not being sure what these other concepts would be, we proposed Telos as an extensible language. From a user perspective, a critical success factor for the long-term success of Telos was that it combined three important perspectives: a rather standard logic foundation suitable for formal reasoning as in deductive databases of the late 1980s; a frame syntax suitable for an object-oriented view as well as an extensible graph visualization preceding the UML metamodel of the late 1990s; and an atomic proposition structure as a precursor to the RDF triple store of the early 2000s.

For such a long-running research project, it is valuable to take a retrospective look at the features of Telos, its implementations and applications, along with an assessment of what worked and what didn't. The main objective of this paper is exactly that.

In the rest of this paper, we first present a brief history of research activities and the evolution of ideas Telos is founded on (Sect. 2) followed by a detailed example of modeling with Telos (Sect. 3). We then discuss the main challenges for giving a formal semantics to Telos and the ideas we adopted, mostly from knowledge representation (KR) in AI and deductive databases (Sect. 4). Section 5 presents the implementation of Telos in the SIS and ConceptBase systems together with related work at the University of Toronto. Section 6 discusses important application areas that used Telos as a metamodeling language. We then present the bigger picture of metamodeling and KR research related to Telos (Sect. 7). In Sect. 8, we discuss the lessons we learned from our experiences with Telos. Finally, Sect. 9 concludes the paper with lessons learnt and research directions on software modeling languages that used Telos as research baseline.

2 History and evolution of ideas

Semantic networks have a long and illustrious history as a framework for representing knowledge starting with the Ph.D. thesis of Quillian in [109]. Many research groups were working on semantic-network-based knowledge representation schemes in the early seventies, including the KR group at the University of Toronto. The first proposal for a semantic-network-based KR language came in the Master thesis of Levesque (1977) in the form of procedural semantic networks (PSN) where the semantics of nodes and edges were defined procedurally through attached procedures, in a similar spirit to Minsky's frame theory [86], but in a more constrained and disciplined fashion [80]. Among other features, PSN treats attributes as first-class objects that are instances of attribute classes, allows for metaclasses, and includes most general classes, such as the class of all classes. In this sense, PSN was self-descriptive like the AI programming language LISP. PSN was followed by modeling languages for information systems (Taxis [88]) and software requirements (RML [32]), both of which adopted many of the features of PSN but focused on particular applications within SE, namely Information Systems Design and Requirements Engineering. The retrospective paper [33] places the requirements modeling language RML in the context of its roots in knowledge representation and to languages that followed it such as CML and Telos. RML focused specifically on providing a formal language to express requirements for information systems. It adopted an object-oriented standpoint to define so-called activity, entity and assertion classes. A first-order language was proposed to define integrity constraints, but no implementation was ever completed.

Telos [89] is a descendant of RML and CML (which was an intermediary step from RML to Telos). RML was developed in the Ph.D. thesis of Sol Greenspan, while CML was defined in the Master thesis of Martin Stanley in 1986. Telos was intended to be *extensible*, in the sense that one could define new metaclasses in order to build models for a particular domain. For example, a model for software would use metaclasses for requirements, design and implementation concepts.

Telos, like PSN, treats both individuals and attributes as first-class citizens of the model and represents them uniformly as *propositions*. Propositions consist of a *unique identifier*, a *source*, a *label* and a *destination*; hence, they can be represented graphically as labeled directed arcs with an identifier. Propositions are organized along four dimensions: instantiation/classification, specialization/generalization and two temporal dimensions representing historical and transaction time. Classes are themselves instances of metaclasses. The class hierarchy is *infinite*; one can define metametaclasses and so on. In other words, the classification dimension defines an unbounded linear hierarchy of strata/planes of ever more abstract propositions. There are also ω *-classes* with instances on more than one such plane, such as the class that has as instances all classes (including itself).

Telos also allows one to represent temporal knowledge by extending propositions with a history and a belief time component. *History time* is the time a fact is true in the domain, while *belief time* is the time a fact is added to the model.[1] Historical knowledge in Telos is allowed to be incomplete using Allen's interval algebra [4], extended with dates and times, which is used to encode this incomplete knowledge in terms of temporal constraints.

Another important feature of Telos is the ability to attach typed *integrity constraints* and *deductive rules* to classes. The semantics of these concepts is as in deductive databases [84], but Telos also offers metalevel reasoning based on the predicate *Holds*.

The first formal semantics of Telos was given through a translation of Telos models into first-order logic. This semantics was presented in a technical report [70] and summarized in [89]. Later, a possible-worlds semantics was developed in the Master thesis of Dimitris Plexousakis, also completed at the University of Toronto [106].

Telos was never implemented in its full generality as defined in [89]. In Toronto, there was an initial Prolog implementation (with its temporal reasoner written in C), presented in the Master theses of Manolis Koubarakis and Thodoros Topaloglou, followed by a Lisp implementation called KNOWBEL [71]. These initial efforts were followed by two efficient implementations, one completed at the University of Crete, named Semantic Index System (SIS) [20]. A second one, named ConceptBase [43, 44, 51], was developed at the University of Passau and subsequently at RWTH Aachen. ConceptBase is open source[2] and actively maintained at the University of Skövde as a system for metamodeling and method engineering [52]. SIS is also actively maintained although it is not open source; it is used in various projects of ICS-FORTH in the domain of cultural informatics.

Ever since its inception, the KR ideas of Telos, and its SIS and ConceptBase implementations, had impact and many applications in research areas beyond AI. The first such area is Software Engineering and in particular, Requirements Engineering and Model Management. As the earliest such example which actually motivated much of the Telos development, the DAIDA [47] project aimed to develop an integrated, model-based software development environment

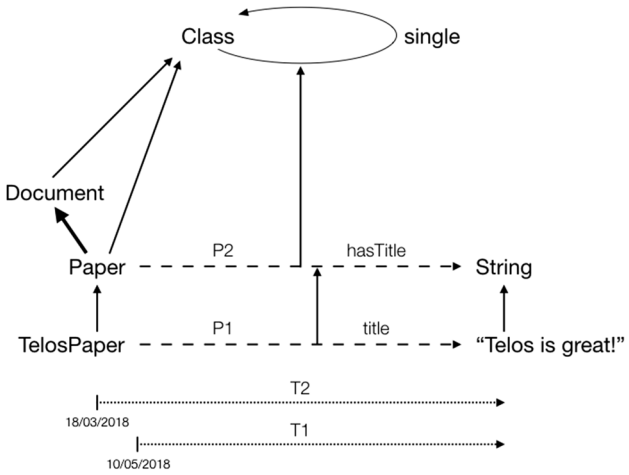

Fig. 1 An example of a Telos model

where developers maintained and evolved requirements, design and implementation models and their inter-relationships in a single repository. The requirements, design and implementation languages adopted were RML, Taxis and DBPL (a database programming language). Telos was used to define the primitive concepts of the three languages so one could maintain cross-linked models [21].

The second area is cultural informatics, especially through the introduction of the CIDOC CRM model [24]) which became ISO 21127 standard in 2006. CIDOC is the International Committee for Documentation of the International Council of Museums, and CIDOC CRM is the conceptual reference model for integrating information about museum objects. The third area where Telos had great impact is education. The SIS and ConceptBase implementations have been used for many years in the teaching of KR concepts to Computer Science and Humanities students with positive results.

3 Telos by example

In this section, we present an example of a Telos knowledge base which is shown graphically in Fig. 1.

Telos models consist of *propositions*, which are classified as *individuals* and *attributes*. Some propositions are *classes*. Individuals are denoted by nodes in the graph of Fig. 1, while attributes are denoted by dashed arrows. In Fig. 1, the proposition `TelosPaper` is an individual and it is an instance of another proposition, the class `Paper`. Instantiation links in Fig. 1 are denoted by arrows of normal thickness.

`TelosPaper` might have been defined by the following Telos statement:

[1] Relational database researchers were the first to make this distinction by introducing the concepts of valid time and transaction time of a tuple in 1985 [120, 124]. Since this paper is on the evolution of the ideas of Telos, let us point out a fact regarding the evolution of these two time dimensions in the database community: It took 26 years for these concepts to be introduced into SQL as part of the standard SQL:2011. This has also accelerated the introduction of temporal features in commercial DBMS.

[2] http://conceptbase.sourceforge.net/.

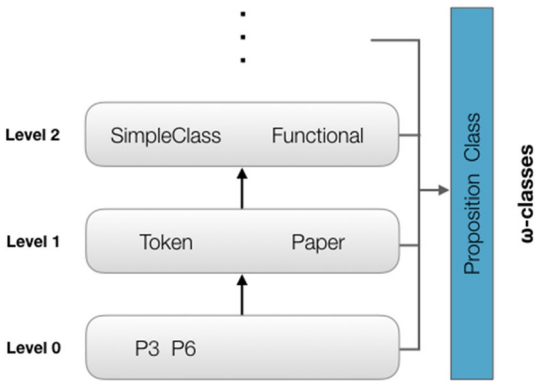

Fig. 2 The infinite instantiation hierarchy of Telos

```
TOKEN TelosPaper (at 10/05/2018..*)
IN Paper
WITH
  hasTitle
    title: "Telos is great!"
END
```

When this statement is processed, the following knowledge base propositions are generated:

```
P1=(TelosPaper, instanceOf, Token, T1)
P2=(TelosPaper, instanceOf, Paper, T1)
P3=(TelosPaper, title, "Telos is great!", T1)

T1 at 10/05/2018..*
```

As mentioned earlier, a proposition consists of a *unique identifier*, a *source*, a *label*, a *destination* and a *time interval* representing history time. Hence, proposition `P2` represents the real-world knowledge that `TelosPaper` belongs to the class `Paper` throughout time interval `T1`.

Since `at` is a synonym for the relation `equals` of Allen's interval algebra [4], `T1` is identical with `10/05/2018..*`, an infinite time interval constant which starts on May 10, 2018. The historical knowledge in the above knowledge base is complete since we know the exact duration of `T1`. The semi-infinite time interval `10/03/2018..*` is a way of representing persistence, e.g., the title of `TelosPaper` remains the same until the knowledge base learns otherwise through an update. By using a relation other than `at/equals`, one can represent incomplete historical knowledge in Telos (e.g., the knowledge that a certain event took place in 1986 but we do not know the exact day or month in that year). The syntax of Telos allows using different temporal relations and time intervals for each part of a statement (e.g., the `IN` clause or a specific attribute and value).

The label `instanceOf` in propositions `P1` and `P2` is built-in, and it encodes the instantiation relationship between `TelosPaper` and classes `Paper` and `Token`. `Token` is a built-in class of Telos, and it has as instances all propositions at level 0 of the instantiation hierarchy of Telos propositions, which is depicted in Fig. 2.

The instantiation hierarchy of Telos is infinite and consists of the following levels:

- *Level 0* includes all *tokens* (i.e., objects having no instances) such as `P0` to `P3` above.
- *Level 1* includes all *simple classes* (i.e., objects having only tokens as instances) such as `Paper` and `Token`.
- *Level 2* includes all *metaclasses*, i.e., these having only simple classes as instances such as `Functional` to be defined below, and the built-in class `SimpleClass`.
- *Level 3* includes all *metametaclasses*, i.e., these having only metaclasses as instances.
- ...
- *Level* ω includes all ω-classes, e.g., `Class, AttributeClass` and `Proposition`. ω-classes can have instances from any level of the instantiation hierarchy.

Before defining `TelosPaper`, a knowledge engineer would have to define the class `Paper` using the following Telos statement:

```
CLASS Paper (at 18/03/2018..*)
IN SimpleClass
ISA Document
WITH
  attribute
    hasTitle: String
END
```

When this statement is processed, the following knowledge base propositions are generated:

```
P4=(Paper, instanceOf, SimpleClass, T2)
P5=(Paper, isA, Document, T2)
P6=(Paper, hasTitle, String, T2)

T2 at 18/03/2018...*
```

Notice that the class `Paper` has been defined for an interval longer than the one of its instance `TelosPaper` as expected. `SimpleClass` is another built-in class of Telos. It lives at Level 2 of the instantiation hierarchy and has as instances all the simple classes at Level 1. The `ISA` clause introduces superclasses of the defined class, and `isA` is the corresponding built-in proposition label. Subclass/superclass links are shown with arrows of double thickness in Fig. 1. In the above statement, the attribute `hasTitle` is also defined with *domain* the defined class `Paper` and *range* the built-in class `String`.

The above Telos statements give us the opportunity to discuss one more important feature of Telos. Proposition `P3` is an attribute proposition, and it is an instance of the

attribute class P6. This is represented by the lower vertical arrow in Fig. 1 and in textual terms by the following instantiation proposition:

```
P7=(P3, instanceOf, P6, T2)
```

In this way, attributes are *first-class objects*, exactly as individuals, in Telos.

Notice also how labels of a proposition such as P6, which is an attribute class, are then used in the level of tokens to introduce an attribute token such as P3. This is the way in which a proposition p living at level i of the instantiation hierarchy *inherits* all attributes defined at level $i+1$ for a proposition p′ of which p is an instance. For each such inheritance relationship at levels i and $i+1$ of the infinite instantiation hierarchy, the *attribute instantiation constraint* of Telos must hold. This constraint requires that if an attribute proposition is an instance of an attribute class, then the source of the attribute proposition must be an instance of the source of the attribute class, the destination of the attribute proposition must be an instance of the destination of the attribute class, and the history time of the attribute proposition must be included in the history time of the attribute class. The reader is invited to check that this holds for attribute class P6 and its instance P3.

The same way of modeling attribute instantiation is used in the built-in features of Telos. For example, the ω-class AttributeClass, which has all attribute classes as instances, has the components

```
(Class, attribute, Class, AllTime)
```

where AllTime is a built-in time interval including all other time intervals. As one might expect given the above Telos statement defining class Paper, P6 is an instance of AttributeClass and the attribute instantiation constraint holds here too.

Now imagine that a knowledge engineer wants to express the fact that papers have at most one title. In other knowledge representation languages (e.g., OWL-DL), this would have been expressed using a built-in construct of the language (e.g., a functional property in OWL-DL). Telos can define functional properties in the language itself, using its primitive mechanisms. For example, one can define the metaclass Functional with the following statement

```
CLASS Functional
COMPONENTS (Class, single, Class, AllTime)
IN AttributeClass, MetaClass WITH
integrityConstraint
   :$(forall s/Single)(forall p,q/Proposition)
     (p in s and q in s and from(p)=from(q) and
     when(p) overlaps when(q) implies p=q)$
END
```

and then redefine Paper as follows:

```
CLASS Paper (at 18/03/2018..*)
IN SimpleClass
ISA Document
WITH
  attribute, functional
    hasTitle: String
END
```

Now P5 is also an instance of the metaclass

```
Functional=<Class, functional, Class, AllTime>
```

where Class is the ω-class of all classes, AttributeClass is the ω-class of all attribute classes, MetaClass is a class which lives on Level 3 of the instantiation hierarchy and has all metaclasses as instances, and functions from and when return the source and history time of their input propositions.

The sorted first-order logic formula in the above statement is an *integrity constraint*. Using this mechanism of integrity constraints, Telos can define other constructs useful for a particular domain through the metamodeling features of Telos. The interested reader is invited to consult [70, 89] for more details of the language, and more examples of metamodeling.

4 Formalizing Telos

The semantics of Telos can be disentangled by considering three linguistic subsets: (1) *Telos0*: traditional semantic-network features like IN, ISA, factual assertions about attributes of objects and specifications of generic properties of attributes, like domains and ranges; (2) *Telos1*: typed FOL formulas used as integrity constraints and deductive rules; (3) *Telos2*: history time and belief time for all propositions; (4) *Telos*: update and query of a Telos model via a TELL/UNTELL/RETELL/ASK interface.

For *Telos0*, the main KR concern is to support reasoning with specialization and instantiation constraints [70, 89]. Notice that contrary to analogous inference rules in more recent ontology languages (e.g., RDFS and OWL), these are treated as *constraints* in Telos, thus better supporting model validation.

The introduction of integrity constraints as typed first-order logic formulas in *Telos1* raises the following question: If we treat the set of propositions told to the system as a model, how should we give meaning to an integrity constraint IC? Integrity constraints is a concept that has been studied for decades in databases, logic programming and KR [123]. Integrity constraints are statements in some formal language that should be true for a model. For example, in the model of Sect. 2, we might want to enforce the

constraint that a paper has only one title at every point in time. Integrity constraints can be static or dynamic. *Static* constraints enforce properties that should be true in a model. For example, in our paper model, every paper must have exactly one title. *Dynamic* constraints, on the other hand, enforce properties that refer to facts in two or more model states. For example, in a model about employees, the salary of an employee should never decrease.[3] Since *Telos3* has a notion of time, both kinds of integrity constraints can be expressed.

Three kinds of semantics of integrity constraints can be found in the literature. The first semantics requires that an integrity constraint IC be *consistent* with the model. The second semantics requires that an integrity constraint is *entailed* by the model. Both of these views come from the deductive database tradition [84] where integrity constraints are considered to be *statements about the world* modeled by a model. The third semantics treats an integrity constraint IC as an *epistemic statement* to be checked against the model, itself described by a set of formulas in first-order logic. This view was originally proposed by Reiter [113].

Reiter [113] uses the epistemic logic KFOPCE of Levesque [79] to formalize static integrity constraints over a model expressed in first-order logic. Checking whether an integrity constraint is satisfied can then be done using the *ASK* operator of Levesque [79] as follows. An integrity constraint *IC* is satisfied by a model *M* if and only if the query *ASK*(*M*, *IC*) returns *yes*.

Reiter [113] proves that if a model is a complete description of its domain, the three semantics of integrity constraints do not differ. However, once we allow existential quantification (null values), disjunction or other forms of incompleteness, the entailment semantics is stronger than the consistency semantics, and the epistemic semantics is the only one that allows us to properly state, e.g., that "the key of a class must be known" [113].

The epistemic semantics of integrity constraints also provides a solution to another dilemma faced by designers of a knowledge-based system: How to decide whether a formula stating something about the domain should go into the model or should be treated as an integrity constraint. The dilemma is typically solved using the detailed knowledge of the domain and the application that is developed. Hence, model formulas are used for answering queries, while integrity constraints are formulas that are checked to see whether they hold in the model, but they do *not* participate in query answering. This methodology is not entirely satisfactory given that integrity constraints are also formulas about the modeled world. Why are not they used in query answering too? Interestingly enough, the dilemma disappears in the epistemic view of integrity constraints: The formulas in the model are about the domain, whereas integrity constraints are about the model itself, thus they "live" outside of the model and do not participate in query processing.

Both formal semantics of Telos given in [70, 89, 106] adopt the semantics of integrity constraints based on consistency. This choice was motivated by practical considerations. By adopting this semantics, implementations of Telos such as ConceptBase have been able to draw on many methods from deductive databases and logic programming for developing efficient algorithms for integrity constraint checking for Telos. To facilitate multiple, perhaps conflicting integrity perspectives on a knowledge base, ConceptBase extended the simple TELL and ASK concepts by introducing a KB view concept called *query classes* which—similar to Reiter's epistemic approach—could be activated on demand to analyze a collection of Telos models [11].

Another important issue in the semantics of Telos is the semantics of propositions expressing deductive rules. Starting from quoted strings, [70, 89] propose reasoning facilities by amalgamating language and metalanguage [9] and defining a provability relation as part of the translation to first-order logic. If we allowed arbitrary formulas to be given as deductive rules, provability would be undecidable. [70] suggested limiting rules to Horn form in order to regain decidability. However, this is not guaranteed, due to the presence of function symbols, like `from()` used in the example of Sect. 3, which can be nested.

The formalization of history time [aspect (3) above] needed to be based on an appropriate theory of time intervals and dates, e.g., as studied by Ladkin [74]. The formalization of belief time, on the other hand, needed to deal with "frame axioms": What propositions remain true when there is a change to a Telos model? This was done in [106] where a possible-world semantics of Telos is presented together with a knowledge-level account which gives semantics to knowledge base operations `TELL`, `UNTELL`, `RETELL` and `ASK` [aspect (4)].

5 Implementations of Telos

In this section, we discuss the SIS and ConceptBase implementations of Telos since these have been operational for 25–30 years and drove significant applications, including commercial ones, as we will see in Sect. 6. While SIS focuses on the efficient evaluation of the structural and inheritance features of Telos, ConceptBase emphasized logic-based optimizations. We also discuss work on *knowledge base management systems (KBMS)* done at the University of Toronto which, although it did not result in a Telos implementation, was pioneering at that time and guided

[3] Sadly enough, in reality, this constraint doesn't always hold.

future work on RDF stores at the University of Crete after 2000.

SIS The SIS implementation of Telos was undertaken in the context of the EU project ITHACA that ran from 1990 to 1994 [20] at the Institute of Computer Science, Foundation of Research and Technology—Hellas. SIS is a scalable C++ implementation of Telos with a client/server architecture and a client side API for accessing and modifying the knowledge base [19]. It was further accompanied by configurable, application-independent data entry, browsing and query formulation components, as well as an interpreter for Telos statements in ASCII format. It was ported to different Unix and Microsoft Windows platforms.

SIS implements only the classical semantic-network features of Telos. It does not implement ω-classes, history time, belief time, integrity constraints and deductive rules. Object labels are globally unique, including attribute labels, which avoids logical conflicts due to multiple inheritance. Built-in reasoning only supports `ISA` transitivity, and the instantiation and generalization constraints of Telos.

The SIS server, implemented in the Master thesis of Georgiannakis [30], stores user-definable Telos data as Telos objects in the form of two core internal C++ classes: `Individual` and `Attribute`. Each object has a unique internal 32-bit ID, an instantiation level and contains 6 small, fixed-size arrays for storing sets of related IDs that can be extended without limits by additional array blocks. These ID sets pertain to the associations `has_instance`, `is_instance_of`, `has_class`, `is_class_of`, `has_superclass` and `has_subclass`. In addition, the class `Attribute` has a set for `has_attribute` and one ID for each of `has_from`, `is_From_of`, `has_To` and `is_To_of`. Thus, SIS implements a complete bidirectional linking between Telos objects. Built-in system classes follow the partition of Telos objects by type (individual, attribute) and instantiation level (individuals, classes, metaclasses, etc.), forming the user-visible endpoints of the linking system. An unlimited list of labels enables associating each ID with a unique name that can be modified by the user at run-time without losing the reference to related objects. In addition, SIS implements the data types `string`, `integer` and `real`. Users can refer to data values by a suitable syntax or data entry form.

SIS is tuned for very high performance and scalability of graph traversal. Storage space is optimized by adjusting storage block sizes against linking blocks according to actual usage statistics. It features highly optimized multi-level caching of disk blocks, set arrays and individual objects, using techniques borrowed from the Unix paging systems, and sorted sets for the bidirectional linkage, maintained from data entry time on. The query system consists of primitive set-valued operations. The most basic ones take a set of IDs as input, e.g., some persons, and return the union of associated IDs, e.g., "all attributes" of these persons or "all classes." Unions and intersections of sorted sets have complexity $O(n)$. This radical deviation from relational database techniques (implemented in the Master thesis of Ntantouris [98]) renders the pre-calculation of transitive closures unnecessary, and even cycle detection comes at very low cost. In 1994, the SIS was tested with a population of up to 850,000 objects (individuals and attributes), with a maximum capacity of 1 billion. A recursive query on a binary tree including cycle detection with 1024 links required about 2 s on a Sun Sparc station. Up to a total population of 500,000 objects, no significant influence of the population size on the query speed could be measured. Batch data import of 10,000 individuals and attributes required about 2 minutes on the same machine. The largest application, developed in 1996, was managing the Getty Thesaurus of Geographic Names[4] with some 100,000 records describing historical places. To the best of our knowledge, SIS was the fastest system comparable to Telos KR languages in the 1990s.

Another interesting feature of the SIS implementation is its simple custom scripting language which allows formulating queries by invoking stateful API components with temporary sets at the server side. This language has been widely used to customize user interfaces, but also interactively for exploring SIS knowledge bases. Set-based graph traversal renders joins practically unnecessary by essentially replacing them, according to our experience with dozens of applications where a real need for joins was never encountered. Second-order features still rare in current KR systems, such as cardinality tests, have been effectively deployed in SIS. Implementation of various reasoning tasks was delegated to procedural code using the API.

ConceptBase The ConceptBase implementation of Telos [43] was developed in the European projects DAIDA (where the Toronto KR group was also a collaborator) and COMPULOG between 1986 and 1992. In his Ph.D. thesis, defended in 1992, Manfred Jeusfeld showed that Datalog with stratified negation is sufficient to describe the semantics of a Telos version without history time. Together with the logic programming technique of partial evaluation, this result allowed to reuse all the query optimization and integrity checking results from deductive databases to automatically optimize rules and constraints *across multiple levels of metaclasses*, or even for externally materialized views (such as browser copies of a knowledge base with limited connection to the original) [122]. As one consequence, ConceptBase was, three years before the advent of the World Wide Web in 1989, the first knowledge-based system to be used in client/server mode across the Atlantic.

[4] http://www.getty.edu/research/tools/vocabularies/tgn/index.html.

Second-order predicates like *Holds(p)* [89] were not implemented but replaced by more restricted multi-level reasoning mechanisms as illustrated in the power type construct discussed below.

A key advantage of Telos over other metamodeling paradigms is its ability to represent its own core principles (instantiation, specialization, attribution) as predefined objects in the so-called ω-level. This resulted in just five predefined ω-classes in ConceptBase. `Proposition` has all propositions in a model as instances, including itself. It has four subclasses. `Individual` has all propositions not relating other propositions as instances. `Attribute` has as instances all propositions that are not individuals. `InstanceOf` is the class of all explicit instantiation relations (relating a proposition to a class), and finally, `IsA` is instantiated by all explicit subclass relations between objects.[5]

In ConceptBase, all objects including individuals have system-generated identifiers and there is no history time associated to propositions anymore.[6] Further, propositions are represented by facts of the predicate `P` (proposition):

```
P(id1, id1, TelosPaper, id1)
P(id2, id2, Paper, id2)
P(id3, id1, instanceOf, id2)
P(id4, id1, title, id5)
P(id5, id5, "Telos is great!", id5)
```

This change enabled the implementation of a proposition store that treats the identifiers like pointers since every reference to a proposition is now is a system-generated identifier.

Constants in ConceptBase are reified to objects. A class `Integer` thus stands for the currently stored integer constants, not for all integers. The predicate `P` is available in the rule and constraint language. It proves in particular useful to define generic properties of relations such as transitivity.

While the view of instantiation levels has a long tradition and is currently discussed in the multi-level modeling community [5], one can also argue that Telos has just two levels: the data type `P(id,x,m,y)` and the database of all propositions matching this data type. A proposition (i.e., object) is an instance of the object `Proposition`, but at the same time it may also be instance of user-defined classes (residing on any abstraction level).

The ω-class `Proposition` (being an instance of itself) is the most general object for defining properties. For example, the notion of transitivity can be expressed in terms of attributes of the object `Proposition` and then is applicable to any usage domain. The rules and constraints for defining the semantics of such generic relations are typically ranging over more than two instantiation levels.

The addition of such generic constructs is an important application of the metamodeling approach to Software Engineering. Most applications of ConceptBase use this ability. We present here the example of multi-level modeling to highlight the idea. In Telos, a predefined axiom defines class membership inheritance:

```
(forall x,c,d/Proposition)
  (x IN c) and (c ISA d) ==> (x IN d)
```

The formula is realized as a deductive rule that derives instantiations to the superclass. A simple addition of two relations and a user-defined deductive rule extends Telos to DeepTelos [55] for multi-level modeling (simplified variant):

```
Proposition WITH
  attribute
    powerType: Proposition
END

(forall m,x,c/Proposition)
  (x IN c) and (m powerType c) ==> (x ISA m)
```

The rule is similar to the membership inheritance axiom. The object `m` is declared as so-called most general instance of the "powertype" object `c`. Then, each instance of `c` becomes a subclass of `m`.

Cross-notational links Consider, for example, the concept of a data store in a process model. It stands for the location, where data are stored. In the multi-perspective approach, a link between the data store and its data model is planned at the metametaclass level:

```
TASK WITH
  attribute
    writes: STORE;
    reads: STORE
END
TYPE END
STORE WITH
  attribute
    hasType: TYPE
END
```

The link `hasType` bridges the process perspective to the data model perspective. One abstraction level lower, modeling languages are defined using this metametamodel:

[5] The original version of Telos presented in Sect. 3 has three more ω-classes: `IndividualClass`, `AttributeClass` and `OmegaClass`.

[6] ConceptBase supports the belief time of an object, i.e., the time when the object was first created and when it was marked as deleted, if applicable.

```
Activity IN TASK WITH
  writes output: Database
  reads input: Database
END
Database IN STORE WITH
  attribute
    hasType type: ObjectType
END
ObjectType IN TYPE END
EntityType IN TYPE ISA ObjectType END
RelationshipType IN TYPE ISA ObjectType END
```

Of particular interest are cross-notational links such as `hasType`. They come with integrity constraints expressing the proper linkages of models expressed in the perspectives. For example, each database such as `PERSONDB` must have at least one object type such as `Person`.

Note that the concepts `PERSONDB` and `Person` are one instantiation level below the concepts `Database` and `ObjectType`, which are themselves instantiated from the metametamodel. The constraint can still be expressed at the metametamodel level:

```
(forall s/STORE) (exists t/TYPE) (s hasType t)
```

This formula ranges over three instantiation levels. The concepts `STORE` and `TYPE` are metametaclasses. The variables `s` and `t` are instances of instances of these classes. The formula is equivalent to the formula:

```
(forall s,S) (s IN S) and (S IN STORE)
    (exists t,T) (t IN T) and (T in TYPE) and
                 (S hasType/m T) and (s m t)
```

ConceptBase partially evaluates such multi-level formulas to a set of formulas ranging over only two levels. For example, the above definition of `Database` yields the following facts:

```
(Database in STORE)
(ObjectType in TYPE)
(Database hasType/type ObjectType)
```

By evaluating the multi-level formula against these facts, we obtain:

```
(forall s) (s IN Database)
    (exists t) (t IN ObjectType) and (s type t)
```

ConceptBase architecture The architecture of ConceptBase has three levels. The lowest level is the object store. It has a highly optimized data structure for propositions and its four flavors (individuals, attributes, instantiations and specializations). The middle level (called the ConceptBase server) is an engine for rules, constraints and queries based on Datalog with stratified negation. Finally, there are a number of client programs that interact with the ConceptBase server by textual and graph views.

The ConceptBase server includes compilers that translate the first-order logic syntax of deductive rules, constraints and queries defined in [89] to executable Datalog code. Several optimization techniques are employed to achieve short response times. First, predicates that are implied by Telos axioms are removed. This eliminates most instantiation predicates. Second, the predicate order in rule bodies is rearranged based on the size of their extensions. Finally, partial evaluation is used to replace rules with variable-rich predicates by a set of rules whose predicates have more constants. The partial evaluation is in particular used to eliminate occurrences of instantiation predicates where the class parameter is a variable. Besides efficiency, the partial evaluation also yields Datalog rules with less variables, hence being more stratifiable.

Later versions of ConceptBase included a number of features that were not included in the original Telos specification. The rules and constraints of Telos are augmented by an active rule component. Similar to tuple-generating dependencies, this allows, for example, to specify mappings between modeling languages. Arithmetic and recursive functions allow to compute metrics such as the length of the shortest path between two nodes in a graph. Finally, a module concept allows to manage a large number of models in the same server and control the visibility of objects in modules. ConceptBase supports multi-user access, a primitive form of ACID transactions, and a mechanism to grant access rights to modules based on user-definable deductive rules.

ConceptBase can handle relatively large Telos models. The size is limited by the number of bits used for an object identifier, currently 32 bits. This limit allows to store 4 billion distinct propositions (metaclasses, classes, instances, attributes, rules, constraints, queries). Ludwig et al. [82] have compared the performance of ConceptBase and Protégé/Racer for various ontology engineering services such as definition of classes and querying of instances. While ConceptBase is relatively slow for class definitions (due to compiling constraints and checking of built-in axioms), it has a linear query response time for the sample queries in contrast to the quadratic response time by Protégé/Racer.

Table 1 summarizes the previous discussion by comparing SIS and ConceptBase.

The KBMS Project at Toronto Parallel to the activities leading to the development of SIS and ConceptBase, the Telos KBMS project at the University of Toronto proposed a generic architecture for a KBMS intended to support applications requiring the construction, efficient access and management of large, shared knowledge bases [14, 90]. The

Table 1 A comparison of SIS and ConceptBase

Feature	SIS	ConceptBase
Implementation languages	C++	Prolog, C++
Architecture	Client/server	Client/server
Platforms	Unix, Windows	Linux, Windows
ω-classes	No	Yes
Metaclass levels	4	Unlimited
Object identifiers	32 bit	32 bit
ISA axioms	Yes	Yes
Instantiation axioms	Yes	Yes
History time	No	No
Belief time	No	Yes
Integrity constraints	No	Yes
Deductive rules	No	Yes
Higher-order predicate *Holds*	No	No

architecture assumed Telos as the knowledge representation language with its assertional sublanguage used to express integrity constraints and deductive rules. It also provided for general-purpose deductive inference and special-purpose temporal reasoning. A number of results were produced in the context of the project addressing several knowledge base management issues. Specifically, for storage management, a new method was proposed for generating a logical schema for a given knowledge base [125]. Query processing algorithms were proposed for semantic and physical query optimization, along with an enhanced cost model for query execution cost estimation [126]. On concurrency control, a novel concurrency control policy taking advantage of knowledge base structure was shown to outperform two-phase locking for highly structured knowledge bases and update-intensive transactions [12, 13]. Finally, algorithms for the efficient processing of integrity constraints and deductive rules during knowledge base operations were described in [107, 108]. Original results included novel data structures and algorithms, as well as performance evaluation techniques [118, 119]. Some of the above results (e.g., the use of vertical partitioning for the storage of Telos knowledge bases in relational systems) were later recast in the context of RDF stores initially by the ICS-FORTH Semantic Web group [3] and later by other database researchers [1].

6 Applications of Telos

In this section, we discuss the role of Telos in four application domains: software and data engineering, cultural informatics and education.

Figure 3 presents an overview of the specific research fields where Telos and its implementations have had significant impact, together with indications of the years where this impact mostly took place.

6.1 Software engineering

The ConceptBase implementation of Telos was initially used in the EU DAIDA project, where it served two roles. First, Telos was used as the specification language for data-intensive applications. Second, it was used as the representation language for software design decisions [45, 49]. The deductive capabilities of Telos were used to specify rules for forward and backward traceability of design artifacts. In the subsequent NATURE project, the core European basic research project on the nature of RE in the early 1990s [42, 83], the ConceptBase implementation of Telos served as the metadata manager bridging the domain, process, and representation perspectives on requirements engineering; in the follow-up CREWS project, goal modeling and multimedia scenarios were added as key object types [36]. In the ITHACA project, the SIS implementation of Telos was used to develop a Software Information Base (SIB) [21] which featured a conceptual model for capturing software requirements and static analysis data of large applications and a software component repository indexed by functional descriptions.

An additional benefit of Telos in the software development domain was its ability to describe other modeling languages such as ER diagrams through metamodeling. The idea was therefore to use these capabilities of Telos to directly store such models in ConceptBase, and to make them available to external development tools through query processing. The metamodeling capability was later used in numerous applications of ConceptBase to define domain-specific conceptual modeling languages.

The uniform representation also allows to link the constructs of different modeling languages, such as a data modeling language and a process modeling language. This feature was pioneered in the DAIDA project. One can distinguish two types of relations: Two modeling languages are covering different aspects of the universe of discourses, or they cover different implementation stages of the software development process. The objects at the metaclass levels are used to define objects at the class level, instantiation relations and attributes defined at the metaclass level. This continues down to the instance level. The objects at the metaclass level are the framework to define the links between objects at the simple class level. Hence, models of some universe of discourse are linked to each other because their metamodels are linked as well.

Requirements traceability The requirements community has emphasized the need to trace modeling and software artefacts from early requirements to the implemented code, and backward [28]. The DAIDA project has presented a solution for traceability that was made possible by the flexibility of Telos in metamodeling (to represent the modeling and programming languages) and to super-impose the software

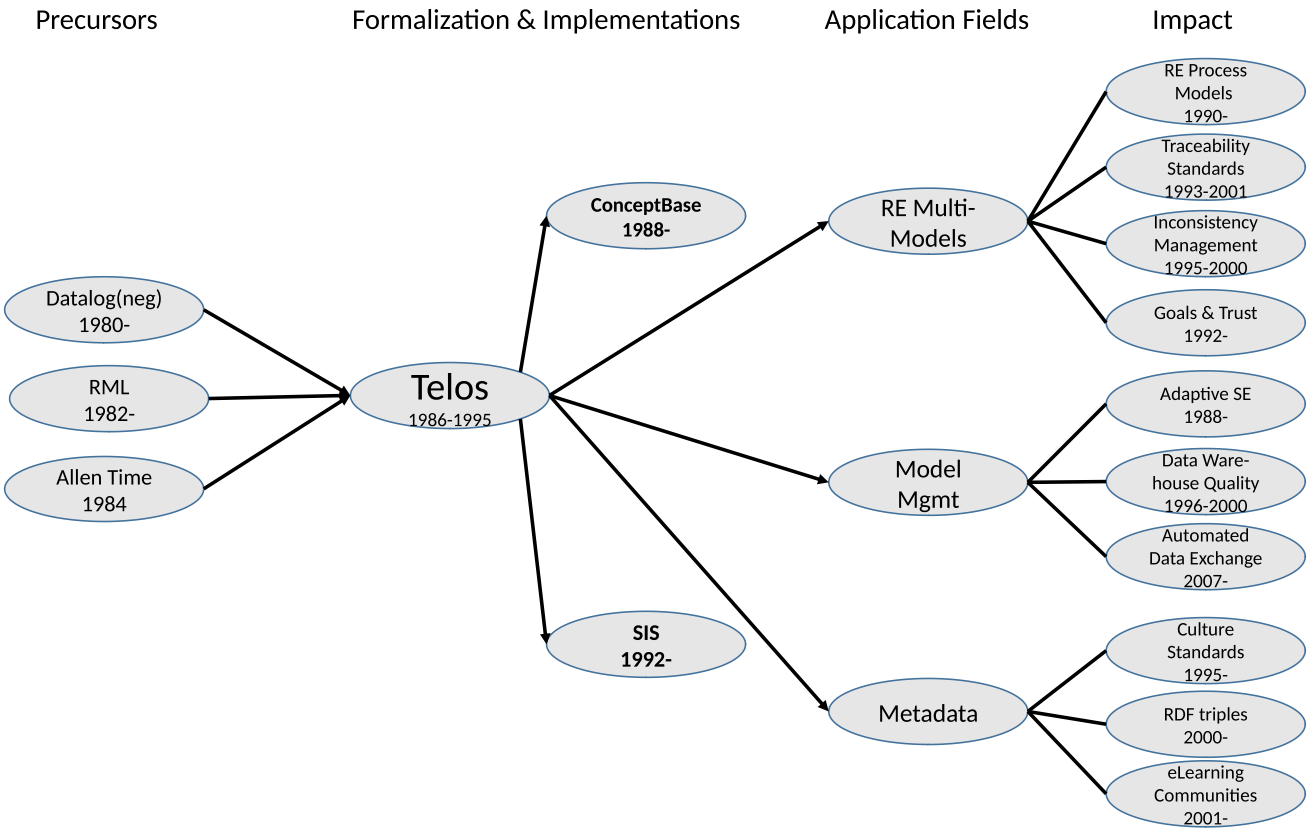

Fig. 3 Research fields linked to Telos

development framework, which realized full traceability. The DOT metametamodel [45] of DAIDA defined three metaclasses for establishing the dependency between artefacts: *design objects* linked by *design tools*, based on human or automated *design decisions*, which could be instantiated to a process model and trace how and why design artefacts across different languages were derived from each other.

The design objects are the containers for whole models. In DAIDA, Telos was used for two purposes. First, it was the language to specify conceptual models of information systems. Second, it was used as the data model for the repository that kept the dependency between design objects. In a way, the DOT model is a precursor of data provenance approaches such as W7 [111]. It has been extended in a project in the automotive domain that studied the interaction of classical engineering models and processes (usually based on differential equations), and discrete software-based control systems. A similar approach within RWTH Aachen's IMPROVE research center on the digitalization of process industries resulted in a first overall conceptual metamodel with detailed ontological submodels in the chemical and plastics engineering domain, and their cross-relationships [10].

Intensive research between New York University, Georgia State University, and RWTH Aachen University in the 1990s showed requirements traceability to be a very important, but extremely complex phenomenon in systems development practice, with its own maturity levels. In a seven-year effort interleaving interviews and empirical studies with leading software development teams in major and medium-sized organizations, with iterative model building using ConceptBase, resulted in a set of reference (meta)models for requirements traceability [112]. The reference models were adopted by most providers of requirements engineering tools and many important developer and user organizations; judging from the ongoing stream of citations, they still form the background for much of traceability research and practice.

As a typical example, Fig. 4 shows the graph visualization of the DesignRationale submodel, which combines multi-criteria decision support with a Rittel-style argumentation model and is (as the upper part of the picture shows) applicable to any kind of design project.

All such sub-metamodels are integrated via the Telos Traceability Metamodel shown in Fig. 5. Its core (upper part) is a network of design PRODUCT OBJECTS linked by SATISFIES relationships which are stamped as validated

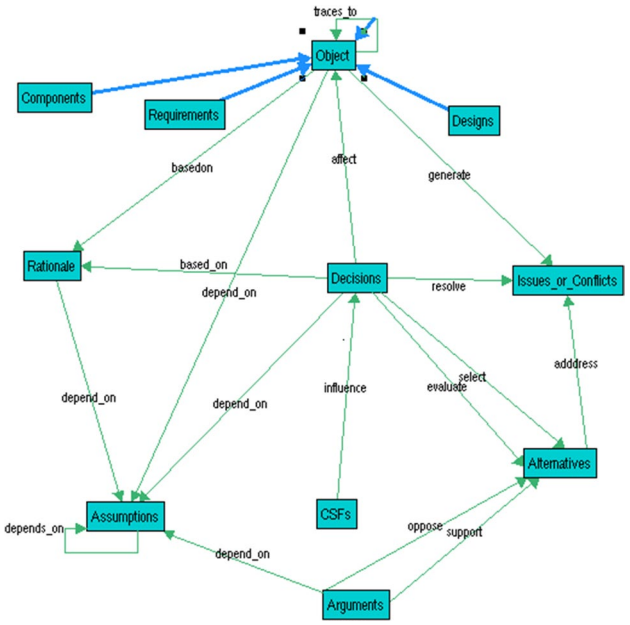

Fig. 4 Rationale Reference Model (screendump), adapted from [112]

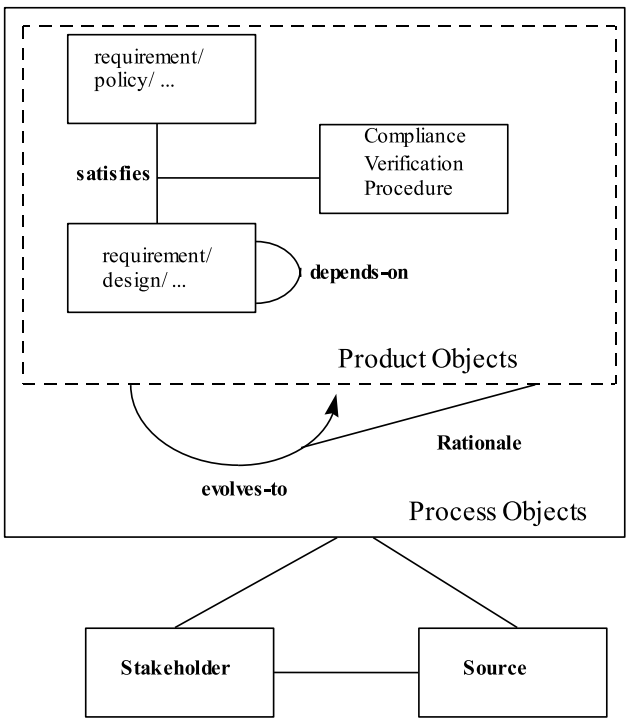

Fig. 5 Telos Metamodel for Traceability Reference Models, adapted from [112]

by COMPLIANCE verifiers, and possibly by technical DEPENDENCIES (e.g., design configurations which only jointly satisfy a higher-level specification). This product network is embedded in EVOLVES-TO process links, where each evolutionary step can include a RATIONALE, which can itself be a complex PRODUCT OBJECT (see Fig. 4). In addition to this product & process kernel, the metamodel also comprises a STAKEHOLDER metaobject which can be instantiated to the network of human and organizational stakeholders involved in the process, and a SOURCE metaobject instantiated to the physical context in which the traceability process and its documentation are placed. ConceptBase employs metalevel integrity constraints, deductive rules and query classes to support the avoidance or detection of inconsistencies between the many submodels in actual projects which instantiate the reference models.

Multi-perspective modeling and inconsistency management The DAIDA project pioneered the idea of using a single repository for storing design objects represented in heterogeneous modeling languages. However, the set of modeling languages was geared toward the implementation dimension, i.e., the focus was on mapping specifications to designs, and then to their implementation. There is a second dimension called the modeling perspective, which identifies interrelated modeling languages that cover only a certain aspect of the information system, e.g., the data perspective in contrast to the process perspective. Zachman [134] identified six such perspectives. Not by coincidence, these perspectives are virtually the same as in the W7 provenance model. In the approach of [96], the perspectives are planned by a rich user-defined metametamodel, which emphasizes the linkages between modeling languages.

Since 1994, a German consulting firm used ConceptBase to identify problems in business process models [96]. The key approach was to have each department in a user organization build their own Telos model following a standardized metamodel and then apply query classes (representing a kind of anti-pattern) for problem identification. As a real-world example, one department produced a monthly report which they considered very important for another department, but that department never knew why it got these reports; deeper analysis following automated problem identification by a query class showed that the report simply arrived several days after the decisions, which it was supposed to support, had to be made.

In a significant extension to this idea, Robinson et al. [114–116] used Telos and ConceptBase for their DealScribe system to manage possibly conflicting requirements of multiple stakeholders. The system is based on a metamodel (or ontology) of dialog statement expressed by stakeholders. The dialog with stakeholders is ordered by a goal model that enforces that certain questions are asked in a logical sequence. The stakeholder requirements are collected by a hypertext-based system and then analyzed by a set of ConceptBase queries. An example query is to retrieve the "most contentious requirements statement," i.e., the requirement of a user with the highest priority for which no resolution is known and that is not superseded by

another contentious requirements statement. Requirements interactions are pairs or sets of requirements that have an impact on each others fulfillment.

The DealScribe system manages a large number of such analysis queries and maintains their answer, i.e., the objects that match the query condition. The stakeholders are thereby kept up-to-date about the status of their requirements. The system promotes the concurrent development and monitoring of requirements by a team of stakeholders. The team is supported by a set of metrics that informs them about the number of unresolved conflicts.

Requirements modeling for telecom service design Eberlein [25] used Telos to realize a tools called RATS to support the design of telecommunication services. The tool targets requirements engineers and services designers, i.e., the upper level of the model-driven development of information systems. Requirements can be functional or non-functional and they have an agreement status. Two types of guidance are offered. Passive guidance informs the stakeholders about inconsistencies in their models (as expressed by ConceptBase queries). Active guidance is provided by textual explanations of development steps, as well as context-specific errors messages that, for example, inform users when they provide a design for a requirements that is not yet agreed upon. It also proposes logical next steps given the current status of a requirements model and its associated design model. The Telos language is used for modeling both the requirements and the design of the telecommunication system. Further, Telos is used to define the links between these two perspectives. The query language of ConceptBase supports both active and passive guidance.

Goal and trust modeling The first version of the social modeling language i* [132, 133] was formalized by mapping it to Telos using the OME tool [2]. In [29], Telos is used as a bridging representation between i*-based strategic requirements, process specifications and execution monitoring for inter-organizational trust management in shared information systems, implementing a sociological distinction between personal trust, trust in the system, and explicit distrust manifested by expensive monitoring mechanisms developed by sociologists from Berlin's Free University. Later, the adaptation of i* to control systems [97] was realized in Telos and ConceptBase as well using the query language to analyze i* goal models. The work also used the timestamps of Telos propositions to extract all changes between two time points. An integration [105] of the two requirements formalisms CIMOSA, Albert-II and i* was also based on Telos and ConceptBase. Finally, Telos and ConceptBase were used to realize a metamodel to bridge business models and requirements models [31].

The above applications highlight the usefulness of Telos for the requirements engineering domain. The unlimited instantiation hierarchy of Telos allows to represent multiple interrelated modeling languages in a single (and simple) framework based on the Telos propositions. The rule, constraint and query capabilities of ConceptBase support the analysis of large-scale conceptual models and cross-model analyses.

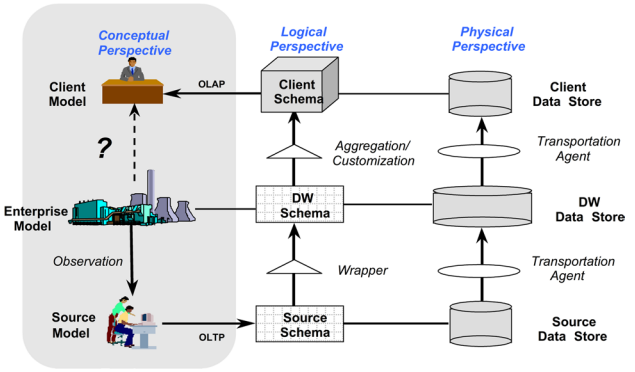

Fig. 6 Metamodel for data warehouses, adapted from [46]

6.2 Model management and data integration

Model Management 1.0 The main value of traceability obviously lies in making systems more adaptive through maintenance and monitoring of consistency between requirements and running systems. The DAIDA project had already shown how the documentation of such dependencies between models can be structured using Telos. The Model Management movement, initiated by Microsoft Research in 2000 [7], goes one step further, in order to deal with the rapidly growing number of software and data models. In his early papers, Bernstein—one of several ConceptBase users at Microsoft—cited DAIDA and Telos as important precursors of these ideas, but decided to follow an algebraic approach with operators such as model matching, mapping and composition instead of Telos' logic-based metamodeling strategy. Some algebraic model management technologies have in fact found their way into the Windows operating system.

Model-based data integration In the data engineering field, the DWQ project (1996–1999) demonstrated Telos models for quality-controlled mappings between data sources via data extraction, cleaning, transformation and loading into data warehouses [46]. As Fig. 6 shows, the Telos metamodel interpreted source data as (limited) views on the reality of an enterprise, whereas the data warehouse schema can be directly derived from a conceptualization of the enterprise. In contrast to earlier approaches which saw a data warehouse schema simply as an integration of source schemas, DWQ's "local-as-view" approach enables a data quality analysis to talk in natural ways about incomplete knowledge or overlap in data sources. Complementing

the Datalog-based analysis capabilities of ConceptBase, the DWQ group at University Roma Sapienza developed a description-logic approach to the specification of mappings between sources and conceptual data warehouse schema which enabled certain kinds of exclusion constraints and subsumption reasoning. The DAIDA metaconcept of DesignTool was employed to document reasoning results in the Telos. In follow-up work, Lenzerini expanded his DWQ approach into a general theory of DL-based data integration, resulting in the most-cited paper in the field [76].

Data Exchange Using Model Management 2.0 Around 2005, such novel declarative approaches to data integration refocused Bernstein's initial algebraic approach to Model Management on the core concept of model Mappings. While ConceptBase focused on the documentation and formal analysis of model mappings, "Model Management 2.0" [8] pursues more automation from two directions: the automated generation of data integration or data exchange code from the mapping specifications, and the at least semi-automatic "machine learning" of mappings from text-based or even instance-based matchings between source and target databases.

Here, we only elaborate the first issue even though joint work with Avi Gal at Technion Haifa has also created competitive solutions enhancing standard ontology matching by metamodeling [110]. Neither Telos semantics nor Description Logics provide declarative means to specify the actual creation of transportable data objects. In the CLIO project [27], an IBM Research team demonstrated that tuple-generating dependencies (TGD) from 1980's database theory research can solve this problem at least for the commercially highly important special case of mappings among extended relational database schemas.

The GeRoMe approach developed at RWTH Aachen University [59] employs role-based meta modeling to factor out the combinatorial explosion to individual metaproperties and enables model composition and efficient querying even across heterogeneous data models [60]. GeRoMe also improves view integration by provably minimal integrated views, thus strengthening model quality and query/data exchange performance in different settings such as database design from requirements views, or data warehouse data integration [81].

6.3 Cultural informatics

In the years after 1992, SIS was used to implement a series of cultural information systems for documentation and education applications in cultural heritage organizations. The most notable of these systems is the terminology management system SIS-TMS, commercially exploited by ICS-FORTH until recently, which is capable of handling large vocabularies, such as the Getty vocabularies Art and Architecture Thesaurus, the Thesaurus of Geographic Names, and the Union List of Artist Names. Based on the experience gained from developing and using the Telos-based museum documentation system CLIO [18], the SIS team was able to convince the International Committee for Documentation (CIDOC) of the International Council of Museums,[7] that a knowledge representation specification as a reference model is the best way to integrate multi-disciplinary information about museum objects in all its scientific depth, and was assigned the leadership of the CIDOC CRM Special Interest Group.[8] The activity of the group, which started in 1996 and is still ongoing, resulted in a high-level ontology, the CIDOC Conceptual Reference Model (CIDOC CRM[9]), which became ISO standard 21127 in 2006, updated in 2014. The model enjoys increasing uptake, e.g., by the British Museum [99], and a set of mutually harmonized extensions,[10] covering museum collections, history, archaeology, library data and the recording of scientific observations in various disciplines including archaeology, art conservation, geology and biodiversity. Whereas most users now employ an OWL version, the master copy is still maintained and validated in Telos by ICS-FORTH on SIS. In the above applications, there is recently an increasing interest for using attributes of attributes and metamodeling, both of which can be done nicely in Telos.

6.4 Education and ontology engineering

Telos has been put to educational use since 1998. The SIS implementation in particular has been used at the University of Crete and the Athens University of Economics and Business in undergraduate and graduate conceptual modeling and system analysis courses for computer scientists and, to a lesser extent, for digital humanists, along with UML diagrams, totaling about 1500 undergraduate and 250 graduate students. In the newer version of the conceptual modeling course, including semantic web applications, students are also exposed to OWL and RDF(S).

In all these courses, it has been consistently observed that two particular features of Telos turned out to be especially useful from an educational point of view: metaclasses and the attribute instantiation constraint. The compulsory explicit declaration of the instantiation level of an object in Telos imposes a clarity of perception concerning the role of the object in terms of specifying and/or conforming to a structure or behavior. It also helps convey a disciplined

[7] http://network.icom.museum/cidoc/.

[8] http://network.icom.museum/cidoc/working-groups/crm-special-interest-group/.

[9] http://www.cidoc-crm.org/.

[10] http://www.cidoc-crm.org/collaborations.

"bottom-up" thinking when designing models, i.e., working from token-level ground facts upward, thus always exposing the consequences of the object being at a particular instantiation level. Although metaclasses are also supported by OWL2 through punning, the characterization is not part of a specification process therefore not equally visible nor consequential. Similarly, the attribute instantiation constraint in Telos explicitly acknowledges the need for semantic validation, which is not quite well served by OWL. Using Telos, students acquire this practice. It is worth noting that SIS Telos proved equally accessible to computer science, digital humanities and archaeology graduate students. We believe that this might be attributed to the clarity and simplicity of the language along with the practice it imposes of making all modeling decisions explicit.

ConceptBase has been used in education at RWTH Aachen University and about a dozen other universities worldwide in the 1990s. Moreover, it served as prototyping tool for early ideas on Web-based learning networks, such as Nejdl's Edutella system [92], which spawned, together with other efforts, the European Conference series on Technology Enhanced Learning.

At Tilburg University, ConceptBase was used in a master-level method engineering course to design and formalize domain-specific languages, and to explore links between modeling languages. The course went on from 2004 to 2010 and attracted about 20 students each year. ConceptBase has also been used by Saïd Assar in a metamodeling course at Telecom Ecole de Management (France). Assar notes that the particular strengths of Telos are the support of multiple abstraction levels, the explicit nature of instantiation links, and the ability to represent class attributes. The observation about the usefulness of the attribute instantiation axiom has also been made in these courses. In 2019–2020, ConceptBase was used at University of Skövde (Sweden) in a Ph.D. course on domain-specific languages (including goal modeling) and on a master-level course on models for IT-security of cyber-physical systems. Finally, about 30 Ph.D. theses so far have used ConceptBase to a greater extent.

ConceptBase has been used by a number of researchers to define ontologies, for example, to model product information [129]. User-definable constraints facilitate the definition of correct ontologies, e.g., constraints on forbidden product configurations. Another challenging ontology-related ConceptBase application (1997–1998) provided semantic support for the distributed collaborative translation of the large SNOMED medical terminology with more than 10.000 concepts from English to German. A team of medical volunteers did this work in their free time mostly from home with very limited interaction among them. A ConceptBase model representing both versions and the translation mappings between them did not just point out emerging inconsistencies among the different translators working on related sections of the terminology, but also revealed several important inconsistencies in the original terminology itself which was at the time a commercial tool licensed for over 1.000 per copy [61].

While Telos at least in its ConceptBase formalization was directly based on the well-understood Datalog-neg semantics, F-Logic and HiLog originally pursued a more general first-order logic approach which proved hard to implement. In the late 1990s, their FLORID prototype and the later commercial Flora-2 implementation restricted the syntax in a similar way as ConceptBase.

7 Related work

In this section, we first survey research conducted in parallel with work on Telos and also based on similar ideas. Then, we concentrate on research that rediscovered core ideas of Telos. Our survey is selective and should not be understood as a complete survey of the relevant technical areas.

A detailed early discussion of the use of KR languages like RML in Requirements Modeling is given in paper [33], which won the most influential paper award for the 1982 IEEE International Conference on Software Engineering.

The first area related to Telos is that of deductive databases [85] and deductive object-oriented databases [62]. In fact, the ConceptBase implementation of Telos was originally presented as a deductive object-based system for metadata [43]. In the area of deductive object-oriented databases, influential parallel language developments have been the frameworks of F-logic [63, 64] and HiLog [15, 16]. F-logic can be used to model in a declarative way many structural aspects of object-oriented and frame-based languages, e.g., object identity, complex objects, inheritance, polymorphic types, query methods, encapsulation, etc., HiLog is a logic with higher-order syntax, but first-order semantics, which allows arbitrary terms to appear in places where predicates, functions, and atomic formulas occur in first-order logic.

Flora-2, an implementation of F-logic, is still actively maintained[11] and used, e.g., by the company Coherent Knowledge and SRI International. Coherent Knowledge markets an AI and NLP product called Ergo Suite which is based on Flora-2.[12]

The second area related to Telos is the area of meta-level representation and reasoning. In Description Logics and Ontologies, there have been a few interesting papers that consider metamodeling issues for ontology languages such as OWL. OWL supports metamodeling through *punning*, e.g., the same name can be used to denote a class and

[11] http://flora.sourceforge.net/.

[12] http://coherentknowledge.com/.

an individual (instance). However, the direct semantics of OWL2 treat punning in a way that an individual and a class with the same name are mapped to different elements of an interpretation. In this way, no changes are needed in the reasoning machinery of OWL2 to deal with punning.

In 2001, Pan and Horrocks presented a stratified approach to metamodeling in RDFS and OWL-DL [100–103] which, like Telos, considers a possibly infinite hierarchy of strata where individuals, classes and properties, metaclasses and metaproperties and so on can live. This stratified approach is stricter than the approach taken by Telos because each layer has its own vocabulary even for predicates like "subsumes," but it has the nice feature that reasoning tasks can be done in a stratified manner; hence, the defined logics have exactly the same computational properties as the logics of each stratum.

Subsequently, Motik utilized the first-order semantics of HiLog for giving semantics to metalevel features of the description logic $\mathcal{SHOIQ}$ (the logic underlying OWL-DL) and, in this way, he obtained a decidable logic. If the standard OWL-Full semantics are used, the same logic is undecidable (technically, even $\mathcal{ALC}$-full is undecidable) [87].

Another implemented ontology system that treats triples as objects, for which metalevel knowledge (e.g., provenance) can also be stated, is described in [128]. The authors take an OWL knowledge base, use annotations to represent metainformation, and then create a metaknowledge-base by reification. They also define a query language that allows both knowledge and metaknowledge to be queried together.

In [23], the problem of metamodeling in description logics was revisited. For every description-logic language $\mathcal{L}$, the authors show how to define its higher-order version $Hi(\mathcal{L})$ and how to give it semantics as in HiLog. More importantly, the authors also showed that the computational complexity of reasoning and question answering for unions of conjunctive queries remains the same for the higher-order version of the description logic $\mathcal{SHIQ}$ and logics in the DL-Lite family. More recently, following the same approach, [77, 78] defined a higher-order version of OWL2 QL and studied the computational complexity of various reasoning and question answering problems showing that this remains the same as in OWL2 QL. Related description logics with metamodeling features have also been proposed by other groups of researchers [72].

Omega [38] and CycL [75] are some of the early systems in this field with such features, which were inspired by Weyhrauch's work on reflection [130], and both use metareasoning mostly as a way of *describing/controlling the inference process*, rather than conceptual modeling.

The third area related to Telos is that of temporal representation and reasoning. Telos was the first KR language to propose the modeling of history and belief time. Historical knowledge in Telos can be incomplete, and incompleteness is captured using temporal constraints [68]. This feature of Telos was not implemented in its SIS and ConceptBase implementations; as we saw, ConceptBase implements only belief time. In parallel with Telos, another well-known KR formalism, the Event Calculus, was extended to handle belief time in the Ph.D. thesis of Sury Sripada at Imperial College [121]. Later on, the ability to model incomplete historical knowledge motivated one of the developers of Telos to study the framework of incomplete constraint databases in his Ph.D. thesis [65, 66].

Metamodel development and use often employ a graph visualization of the Telos frame syntax. A recent cooperative modeling environment at RWTH Aachen University even enables realtime support for metamodel-assisted cooperative modeling and view integration [94]. Other metamodeling environments employ directly an extensible graph syntax, with semantics often checked by services offered in the API. MetaEdit+ developed at the University of Jyväskylä, Finland [58] is the best-known early example of this kind. More recently, University of Vienna's commercialized metamodeling environment ADONIS [56] whose original development was also motivated by the Telos experience is actively used in research, education and practice. Initiated in 1997 and regularly revised and enriched ever since, the UML meta object facility (MOF) also follows this graph-oriented approach while adopting the four metalevel structure already pursued in the DAIDA project; over the past 20 years, formal semantics is gradually being added, e.g., using UML's Object Constraint Language OCL. Further details on these and other metamodeling approaches related to method engineering can be found in [50].

More recently, some of the ideas of Telos have been rediscovered by Semantic Web researchers in the context of the Resource Description Framework (RDF) in 1997.[13] RDF is based on triples of the form (*subject*, *predicate*, *object*) that correspond to the source, label and destination components of a Telos proposition. It is important to emphasize that, unlike Telos propositions, RDF triples do *not* have identity. This omission has resulted in the development of ideas such as singleton properties [93], application of named graphs [37] and nested triples [35], which would have been unnecessary within the Telos framework. As early as 2001, the paper [91] defined the framework O-Telos-RDF which demonstrated how RDF and RDFS could have been if they would have been based on Telos.[14] [91] argued eloquently for an RDF version of the core Telos ideas (ID for all propositions, an infinite instantiation hierarchy, etc.), but

[13] The W3C document https://www.w3.org/TR/WD-rdf-syntax-971002/ introduced RDF to the research community. RDF became a W3C Recommendation two years later; its most recent version is RDF 1.1.

[14] O-Telos is the Telos dialect implemented in ConceptBase.

Table 2 Main ideas of Telos and relevant related works by research areas

Main idea	RDF RDFS	Description logics	Deductive and object-oriented databases	Higher-order logics	Temporal databases	Temporal reasoning
Object-centered framework	[91]		[63, 64]			
Metaclasses	[100–103]	[23, 72, 77, 78, 87, 128]				
History time	[6, 34, 69, 73, 104]		[121]			[4, 65, 66, 120, 124]
Incomplete temporal knowledge					[65, 66]	[68]
Belief time			[121]			
Integrity constraints			[85]			
Deductive rules			[84, 85]			
Higher-order predicate *Holds*				[15, 16]		

this initial proposal was not taken up by the Semantic Web community. The Knowledge Graph community has recently been investing a lot of effort in defining worldwide unique identifiers in order to heal this omission.

Semantic Web researchers have also worked on extensions of RDF with valid time (called history time in Telos). The first such proposal is [34] where the concept of RDF triple is extended to a 4-tuple, with valid time being the fourth component, to model the time that a fact represented by the triple is true in reality. This idea was first proposed in CML (and then Telos) in 1986.

More recent Semantic Web proposals for introducing valid time in RDF (but also geospatial information) are the Ph.D. theses of Perry [104] and Kyzirakos [69, 73], and the Master thesis of Bereta [6]. The last two theses also resulted in the implementation of an efficient spatiotemporal RDF store called Strabon.[15]

The Telos group in Toronto also studied representation and inference with incomplete spatial knowledge in the context of Telos, in the Ph.D. thesis of Topaloglou [127]. The topic of modeling incomplete spatial and temporal knowledge using constraints was revisited 20 years later in the context of RDF, in the Ph.D. thesis of Charalampos Nikolaou [95].

The 4-tuple model for valid time discussed above was also rediscovered in the knowledge graph Yago2 [39, 40] where it has been extended to a 5-tuple model which can also represent geographical location.

Table 2 summarizes the previous discussion by offering of tabular view of the main ideas of Telos where similar ideas have appeared in related research.

8 Lessons learned

There are lessons learned from our experiences with Telos. These are discussed below.

The metamodeling framework of Telos constitutes its most distinctive feature and greatest strength. Equally pioneering was the direct integration of frame-based (i.e., close to object-oriented) syntax, relatively simple logic-based formalization, and a physical grounding of the proposition concept which turned out to become extremely popular for modern semantic-network structures such as RDF and its underlying compact storage structures. These Telos features have enabled scalable implementations with wide and long-term usage but also nice typed graph modeling user interfaces.

Another positive lesson learned which was later taken up by others was that the kernel of the axiomatic definition essentially based on Datalog with stratified negation was a critical success factor for efficient query optimization and integrity checking. Thus, the rules and constraints framework could be scalably implemented in ConceptBase and was used in multiple applications. The axioms were virtually never updated and helped users to identify semantic mistakes in their models and metamodels. The axioms related to instantiation of attributes force users to correctly use class/metaclass definitions when creating instances.

In contrast, the parts of the language definition that went beyond what could be mapped to Datalog (neg) proved very hard to implement efficiently. Most importantly, the temporal component of the language (both history and belief time) has seen only partial implementations and has been ignored in most applications of Telos. In hindsight, Telos offers a rather cumbersome representation

[15] http://strabon.di.uoa.gr/.

for time, compared to later proposals in the RE literature, e.g., KAOS [22], a requirements modeling language that uses linear temporal logic to represent temporal knowledge. In contrast, the history time as defined in Telos results in intractable query processing since it has the ability to represent incomplete temporal knowledge. This was established by complexity results of one of the authors in [67] where the problem of query evaluation for indefinite constraint databases is studied (query evaluation for the history time component of Telos can be cast in this framework).

From an application perspective, Telos applications in the area of requirements modeling required only the concept of belief time (called transaction time in the ConceptBase implementation), so history time was either not implemented or dropped at an early stage of the implementation. On the other hand, applications in the area of cultural informatics need very refined notions of time, but those can be defined using the semantic-network structures of the language; hence, no history time neither belief time was implemented in the SIS implementation.

A related gap in the original Telos definition proved to be its rather simplistic change operators (TELL, UNTELL). In particular, Telos was missing a transaction concept that tolerated temporary inconsistencies. As a consequence, users found the strict axiomatization and integrity constraints both a blessing and a curse, as it was very hard to avoid temporary constraint violations when making complex changes to the knowledge base—a problem that is also well known with, e.g., referential integrity preservation in commercial databases.

As an example, Telos had a signification impact on subsequent languages for requirements engineering, such as i*. The Telos axioms complicate changes to existing definitions in metamodels, when models already exist. The axiom violations are similar to violations of foreign key references in traditional databases. The difference is that Telos has many more such axioms, which can make incremental change cumbersome. Another observation from applications was that the presence of multiple instantiation levels is conceptually difficult to grasp for human users, in particular the omega-level of Telos, which subsumes objects of any abstraction level. As mentioned before, in the meantime a whole community on multi-level modeling has evolved around this issue, with Manfred Jeusfeld's DeepTelos as an early attempt to make this easier for users.

Similar to the SQL standard which has also faced the referential integrity problem, a number of workarounds have evolved in the Telos implementations and usage. For example, the authors of the Requirements Assistant for Telecommunications Services (RATS) tool [26] note in [54] a tradeoff between efficiency and number of classes and deductive rules. This led them to restrict the number of classes in their conceptualization of requirements for the telecommunications domain.

Another often-used workaround in requirements modeling applications was ConceptBase query classes which can be activated by users at their desire as opposed to always enforced integrity constrains. Additionally, ConceptBase users may employ active rules (not present in the original Telos definition) to heal, e.g., automatically referential integrity violations by constraint propagation. As an example, Robinson discusses the Telos features that were helping with the implementation of his DealScribe [114] tool to track requirements and goal evolution. He reports that an advantage of Telos (ConceptBase) is that the metamodel can be queried by the DealScribe user interface to adapt to changing definitions at the metamodel level. Another important feature was the predicate language used for defining rules, constraints and queries. DealScribe also used the hypothetical querying of ConceptBase, where certain objects are defined temporarily before querying the model ("what if questions").

The pioneering development of Telos well before similar standards such as RDF and XML has created the obvious backward compatibility issues when exchanging models and data with them. Export interfaces have partially been implemented, but import of RDF triples or XML files is still work to be done. Embarrassingly, the interoperability issues extend even to the two Telos implementations themselves. While both SIS and ConceptBase are implementing Telos, they have followed a different path on the supported Telos frame syntax. As a consequence, frame syntax cannot yet be easily exchanged between the two systems but have to resort to the logic or storage perspectives of Telos.

All of these gaps, however, are doable implementation tasks with no fundamental difficulties. For example, an API for the ADOxx tool was recently developed to interface with ConceptBase [53]. It allows to use the constraint checking function of ConceptBase to find semantic flaws in metamodels of domain-specific modeling languages (in particular for requirements engineering) developed with ADOxx [57].

9 Conclusions

Telos was intended as a extensible modeling language for software knowledge. It has seen two major implementations that addressed the hard problems of semantics and scalable reasoning. It has also been applied in multiple application areas, notably Software Engineering and cultural informatics. The literature suggests that core ideas of Telos were revisited, and sometimes extended, by other communities many years later, and put to good use in modern KR technologies, such as RDF stores and knowledge graphs.

The results on Telos, its implementations, and applications reported in the paper have been widely cited, adopted

in standards and practice (e.g., cultural heritage and traceability), or achieved additional visibility by being selected for best paper special issues of reputed journals. At least six edited books report on metamodeling concepts and applications using Telos as the major integrating basis [17, 41, 46, 48, 50, 131] but also including chapters on closely related research of international cooperation partners. The same holds for special issues of journals such as CACM (1998), IEEE Transactions on Software Engineering (1992 and 1998), the RE Journal (1998), and others. Telos-related papers were selected for "best paper" special issues from leading conferences including the ACM-IEEE RE Conference (1996, 1998, 2001) and the European CAiSE conference (1994, 1998, 1999, 2000, 2016).

Acknowledgements We are grateful to all of our colleagues who contributed to the design, implementation and applications of Telos from 1986 onward.

References

1. Abadi DJ, Marcus A, Madden S, Hollenbach KJ (2007) Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd international conference on very large data bases, University of Vienna, Austria, September 23–27, 2007, pp 411–422
2. Alencar FMR, Filho GAC, Castro J (2002) Support for structuring mechanism in the integration of organizational requirements and object orien. In: Anais do WER02—Workshop em Engenharia de Requisitos, Valencia, España, Novembro 11–12, 2002, pp 147–161. http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER02/alencar.pdf. Accessed 30 Jan 2020
3. Alexaki S, Christophides V, Karvounarakis G, Plexousakis D, Tolle K (2001) The ICS-FORTH RDFSuite: managing voluminous RDF description bases. In: SemWeb
4. Allen JF (1981) An interval-based representation of temporal knowledge. In: Proceedings of the 7th international joint conference on artificial intelligence, IJCAI '81, Vancouver, BC, Canada, August 24–28, 1981, pp 221–226
5. Atkinson C, Kühne T (2008) Reducing accidental complexity in domain models. Softw Syst Model 7(3):345–359
6. Bereta K, Smeros P, Koubarakis M (2013) Representation and querying of valid time of triples in linked geospatial data. In: The Semantic Web: semantics and big data, 10th international conference, ESWC 2013, Montpellier, France, May 26–30, 2013. Proceedings, pp 259–274
7. Bernstein PA, Halevy AY, Pottinger R (2000) A vision of management of complex models. SIGMOD Rec 29(4):55–63. https://doi.org/10.1145/369275.369289
8. Bernstein PA, Melnik S (2007) Model management 2.0: manipulating richer mappings. In: Proceedings of the ACM SIGMOD international conference on management of data, Beijing, China, June 12–14, 2007, pp 1–12. https://doi.org/10.1145/1247480.1247482
9. Bowen KA, Kowalski R (1983) Amalgamating language and metalanguage in logic programming. In: Clark K, Tarnlund SA (eds) Logic programming. Academic Press, Cambridge
10. Brandt SC, Morbach J, Miatidis M, Theißen M, Jarke M, Marquardt W (2008) An ontology-based approach to knowledge management in design processes. Comput Chem Eng 32(1–2):320–342. https://doi.org/10.1016/j.compchemeng.2007.04.013
11. Buchheit M, Jeusfeld MA, Nutt W, Staudt M (1994) Subsumption between queries to object-oriented databases. In: Advances in database technology—EDBT'94. 4th International conference on extending database technology, Cambridge, UK, March 28–31, 1994, Proceedings, pp 15–22. https://doi.org/10.1007/3-540-57818-8_37
12. Chaudhri VK, Hadzilacos V (1995) Safe locking policies for dynamic databases. In: Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, May 22–25, 1995, San Jose, CA, USA, pp 233–244
13. Chaudhri VK, Hadzilacos V, Mylopoulos J (1992) Concurrency control for knowledge bases. In: Proceedings of the 3rd international conference on principles of knowledge representation and reasoning (KR'92). Cambridge, MA, October 25–29, 1992, pp 762–773
14. Chaudhri VK, Jurisica I, Koubarakis M, Plexousakis D, Topalaglou T (2009) The KBMS project and beyond. In: Conceptual modeling: foundations and applications—essays in honor of John Mylopoulos, pp 466–482
15. Chen W, Kifer M, Warren DS (1989) HiLog: a first-order semantics for higher-order logic programming constructs. In: Logic programming, Proceedings of the North American conference 1989, Cleveland, OH, USA, October 16–20, 1989. 2 Volumes, pp 1090–1114
16. Chen W, Kifer M, Warren DS (1993) HiLog: a foundation for higher-order logic programming. J Log Program 15(3):187–230
17. Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. International series in software engineering, vol 5. Springer, Berlin. https://doi.org/10.1007/978-1-4615-5269-7
18. Constantopoulos P (1994) Cultural documentation: the CLIO system. Technical report FORTH-ICS-TR115-1994, Institute of Computer Science, Foundation of Research and Technology, Hellas
19. Constantopoulos P, Doerr M (1994) The semantic index system: a brief presentation. Institute of Computer Science, Foundation for Research and Technology, Hellas. http://www.ics.forth.gr/_pdf/brochures/SIS_descr_diafimistiko.WWW.pdf
20. Constantopoulos P, Doerr M, Vassiliou Y (1993) Repositories for software reuse: the software information base. In: Information system development process, Proceedings of the IFIP WG8.1 working conference on information system development process, Como, Italy, 1–3 September, 1993, pp 285–307
21. Constantopoulos P, Jarke M, Mylopoulos J, Vassiliou Y (1995) The software information base: a server for reuse. VLDB J 4(1):1–43
22. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20(1–2):3–50. https://doi.org/10.1016/0167-6423(93)90021-G
23. De Giacomo G, Lenzerini M, Rosati R (2011) Higher-order description logics for domain metamodeling. In: Proceedings of the twenty-fifth AAAI conference on artificial intelligence, AAAI 2011, San Francisco, CA, USA, August 7–11, 2011
24. Doerr M (2003) The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. AI Mag 24(3):75–92
25. Eberlein A (2009) Conceptual modeling in telecommunications service design. In: Jeusfeld MA, Jarke M, Mylopoulos J (eds) Metamodeling for method engineering. MIT Press, Cambridge, pp 169–231
26. Eberlein APG, Halsall F (1997) Telecommunications service development: a design methodology and its intelligent support. Eng Appl Artif Intell 10(6):647–663. https://doi.org/10.1016/S0952-1976(97)00024-9

27. Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theor Comput Sci 336(1):89–124. https://doi.org/10.1016/j.tcs.2004.10.033
28. Finkelstein A, Stevens R (1997) Requirements traceability. In: 3rd IEEE international symposium on requirements engineering (RE'97), January 5–8, 1997, Annapolis, MD, USA, p 265
29. Gans G, Jarke M, Kethers S, Lakemeyer G (2003) Continuous requirements management for organisation networks: a (dis)trust-based approach. Requir Eng 8(1):4–22. https://doi.org/10.1007/s00766-002-0163-8
30. Georgiannakis G (1993) A storage and memory management mechanism for objects in Telos. Master's thesis, University of Crete and ICS-FORTH, Greece
31. Giannoulis C, Petit M, Zdravkovic J (2011) Modeling competition-driven business strategy for business IT alignment. In: Advanced information systems engineering workshops—CAiSE 2011 international workshops, London, UK, June 20–24, 2011. Proceedings, pp 16–28
32. Greenspan SJ, Mylopoulos J, Borgida A (1982) Capturing more world knowledge in the requirements specification. In: Proceedings, 6th international conference on software engineering, Tokyo, Japan, September 13–16, 1982, pp 225–235
33. Greenspan SJ, Mylopoulos J, Borgida A (1994) On formal requirements modeling languages: RML revisited. In: Proceedings of the 16th international conference on software engineering, Sorrento, Italy, May 16–21, 1994, pp 135–147
34. Gutiérrez C, Hurtado CA, Vaisman AA (2005) Temporal RDF. In: The semantic web: research and applications, Second European semantic web conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1, 2005, Proceedings, pp 93–107
35. Hartig O (2017) Foundations of RDF⋆ and SPARQL⋆ (an alternative approach to statement-level metadata in RDF). In: Proceedings of the 11th Alberto Mendelzon international workshop on foundations of data management and the web, Montevideo, Uruguay, June 7–9, 2017
36. Haumer P, Jarke M, Pohl K, Heymans P (1999) Bridging the gap between past and future in RE: a scenario-based approach. In: 4th IEEE international symposium on requirements engineering (RE '99), 7–11 June 1999, Limerick, Ireland, pp 66–73. https://doi.org/10.1109/ISRE.1999.777986
37. Hernández D, Hogan A, Krötzsch M (2015) Reifying RDF: what works well with wikidata? In: Proceedings of the 11th international workshop on Scalable semantic web knowledge base systems co-located with 14th international semantic web conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015, pp 32–47
38. Hewitt C, Attardi G, Simi M (1980) Knowledge embedding in the description system omega. In: AAAI, pp 157–164
39. Hoffart J, Suchanek FM, Berberich K, Lewis-Kelham E, de Melo G, Weikum G (2011) YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In: Proceedings of the 20th international conference on world wide web, WWW 2011, Hyderabad, India, March 28–April 1, 2011 (Companion Volume), pp 229–232
40. Hoffart J, Suchanek FM, Berberich K, Weikum G (2013) YAGO2: a spatially and temporally enhanced knowledge base from wikipedia. Artif Intell 194:28–61
41. Jarke M (1993) Database application engineering with DAIDA. Research reports ESPRIT. Springer, Berlin. https://doi.org/10.1007/978-3-642-84875-9
42. Jarke M, Bubenko JA, Rolland C, Sutcliffe AG, Vassiliou Y (1993) Theories underlying requirements engineering: an overview of NATURE at genesis. In: Proceedings of IEEE international symposium on requirements engineering, RE 1993, San Diego, CA, USA, January 4–6, 1993, pp 19–31. https://doi.org/10.1109/ISRE.1993.324840
43. Jarke M, Gallersdörfer R, Jeusfeld MA, Staudt M (1995) ConceptBase—a deductive object base for meta data management. J Intell Inf Syst 4(2):167–192
44. Jarke M, Jeusfeld MA (1989) Rule representation and management in ConceptBase. SIGMOD Rec 18(3):46–51
45. Jarke M, Jeusfeld MA, Rose T (1990) A software process data model for knowledge engineering in information systems. Inf Syst 15(1):85–116
46. Jarke M, Lenzerini M, Vassiliou Y, Vassiliadis P (2003) Fundamentals of data warehouses, 2nd ed. Springer. http://www.worldcat.org/oclc/49824734. Accessed 30 Jan 2020
47. Jarke M, Mylopoulos J, Schmidt JW, Vassiliou Y (1992) DAIDA: an environment for evolving information systems. ACM Trans Inf Syst 10(1):1–50
48. Jarke M, Rolland C, Sutcliffe A, Dömges R (1999) The nature of requirements engineering. Shaker Verlag, Aachen
49. Jarke M, Rose T (1988) Managing knowledge about information system evolution. In: Proceedings of the 1988 ACM SIGMOD international conference on management of data, Chicago, IL, June 1–3, 1988, pp 303–311
50. Jeusfeld M, Jarke M, Mylopoulos J (eds) (2009) Metamodeling for method engineering. MIT Press, Cambridge
51. Jeusfeld MA (1992) Änderungskontrolle in deduktiven Datenbanken. Ph.D. thesis, University of Passau, Germany
52. Jeusfeld MA (2009) Metamodeling and method engineering with ConceptBase. In: Jeusfeld MA, Jarke M, Mylopoulos J (eds) Metamodeling for method engineering. MIT Press, Cambridge, pp 89–168
53. Semcheck Jeusfeld MA (2016) Checking constraints for multi-perspective modeling languages. In: Karagiannis D, Mayr HC, Mylopoulos J (eds) Domain-specific conceptual modeling, concepts, methods and tools. Springer, Berlin, pp 31–53. https://doi.org/10.1007/978-3-319-39417-6_2
54. Jeusfeld MA, Jarke M, Staudt M, Quix C, List T (1999) Application experience with a repository system for information systems development. In: EMISA, Teubner-Reihe Wirtschaftsinformatik, pp 147–174. Teubner, Stuttgart, Leipzig. https://doi.org/10.1007/978-3-322-84795-9_9
55. Jeusfeld MA, Neumayr B (2016) DeepTelos: multi-level modeling with most general instances. In: Conceptual modeling—35th international conference, ER 2016, Gifu, Japan, November 14–17, 2016, Proceedings, pp 198–211
56. Karagiannis D, Kühn H (2002) Metamodelling platforms. In: E-commerce and web technologies, third international conference, EC-Web 2002, Aix-en-Provence, France, September 2–6, 2002, Proceedings, p 182
57. Karagiannis D, Mayr HC, Mylopoulos J (eds) (2016) Domain-specific conceptual modeling, concepts, methods and tools. Springer, Berlin. https://doi.org/10.1007/978-3-319-39417-6
58. Kelly S, Smolander K (1996) Evolution and issues in metacase. Inf Softw Technol 38(4):261–266. https://doi.org/10.1016/0950-5849(95)01051-3
59. Kensche D, Quix C, Chatti MA, Jarke M (2007) Gerome: a generic role based metamodel for model management. J Data Semant 8:82–117. https://doi.org/10.1007/978-3-540-70664-9_4
60. Kensche D, Quix C, Li X, Li Y, Jarke M (2009) Generic schema mappings for composition and query answering. Data Knowl Eng 68(7):599–621. https://doi.org/10.1016/j.datak.2009.02.006
61. Kethers S, von Buol B, Jarke M, Repges R (1998) Lessons learned from co-operative terminology work in the medical domain. In: MEDINFO '98—9th world congress on medical informatics, Seoul, South Korea, August 14–21, 1998, pp 628–632
62. Kifer M (1995) Deductive and object data languages: a quest for integration. In: Deductive and object-oriented databases, fourth

international conference, DOOD'95, Singapore, December 4–7, 1995, Proceedings, pp 187–212
63. Kifer M, Lausen G (1989) F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In: Proceedings of the 1989 ACM SIGMOD international conference on management of data, Portland, Oregon, May 31–June 2, 1989, pp 134–146
64. Kifer M, Lausen G, Wu J (1995) Logical foundations of object-oriented and frame-based languages. J ACM 42(4):741–843
65. Koubarakis M (1993) Representation and querying in temporal databases: the power of temporal constraints. In: Proceedings of the ninth international conference on data engineering, April 19–23, 1993, Vienna, Austria, pp 327–334
66. Koubarakis M (1994) Database models for infinite and indefinite temporal information. Inf Syst 19(2):141–173
67. Koubarakis M (1997) The complexity of query evaluation in indefinite temporal constraint databases. Theor Comput Sci 171(1–2):25–60
68. Koubarakis M (2006) Temporal CSPs. In: Rossi F, van Beek P, Walsh T (eds) Handbook of constraint programming, foundations of artificial intelligence, vol 2. Elsevier, Amsterdam, pp 665–697
69. Koubarakis M, Kyzirakos K (2010) Modeling and querying metadata in the semantic sensor web: the model stRDF and the query language stSPARQL. In: The semantic web: research and applications, 7th Extended semantic web conference, ESWC 2010, Heraklion, Crete, Greece, May 30–June 3, 2010, Proceedings, Part I, pp 425–439
70. Koubarakis M, Mylopoulos J, Stanley M, Borgida A (1989) Telos: features and formalization. Technical report KRR-TR-89-4, Department of Computer Science, University of Toronto
71. Kramer BM, Chaudhri VK, Koubarakis M, Topaloglou T, Wang H, Mylopoulos J (1991) Implementing Telos. SIGART Bull 2(3):77–83
72. Kubincová P, Kluka J, Homola M (2016) Expressive description logic with instantiation metamodelling. In: Principles of knowledge representation and reasoning: proceedings of the fifteenth international conference, KR 2016, Cape Town, South Africa, April 25–29, 2016, pp 569–572
73. Kyzirakos K, Karpathiotakis M, Koubarakis M (2012) Strabon: a semantic geospatial DBMS. In: The semantic web—ISWC 2012—11th international semantic web conference, Boston, MA, USA, November 11–15, 2012, proceedings, Part I, pp 295–311
74. Ladkin PB (1987) The completeness of a natural system for reasoning with time intervals. In: Proceedings of the 10th international joint conference on artificial intelligence. Milan, Italy, August 23–28, 1987, pp 462–465
75. Lenat DB, Guha RV (1991) The evolution of CycL, the Cyc representation language. SIGART Bull 2(3):84–87
76. Lenzerini M (2002) Data integration: a theoretical perspective. In: Proceedings of the twenty-first ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, June 3–5, Madison, WI, USA, pp 233–246. https://doi.org/10.1145/543613.543644
77. Lenzerini M, Lepore L, Poggi A (2016) Answering metaqueries over Hi (OWL 2 QL) ontologies. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, pp 1174–1180
78. Lenzerini M, Lepore L, Poggi A (2016) A higher-order semantics for metaquerying in OWL 2 QL. In: Principles of knowledge representation and reasoning: proceedings of the fifteenth international conference, KR 2016, Cape Town, South Africa, April 25–29, 2016, pp 577–580
79. Levesque HJ (1984) Foundations of a functional approach to knowledge representation. Artif Intell 23(2):155–212
80. Levesque HJ, Mylopoulos J (1977) An overview of a procedural approach to semantic networks. In: Proceedings of the 5th international joint conference on artificial intelligence. Cambridge, MA, USA, August 22–25, 1977, p 283
81. Li X, Quix C, Kensche D, Geisler S (2010) Automatic schema merging using mapping constraints among incomplete sources. In: Proceedings of the 19th ACM conference on information and knowledge management, CIKM 2010, Toronto, ON, Canada, October 26–30, 2010, pp 299–308. https://doi.org/10.1145/1871437.1871479
82. Ludwig SA, Thompson C, Amundson K (2009) Performance analysis of a deductive database with a Semantic Web reasoning engine: ConceptBase and Racer. In: Proceedings of the 21st international conference on software engineering and knowledge engineering (SEKE'2009), Boston, MA, USA, July 1–3, 2009, pp 688–693
83. Maiden NAM, Spanoudakis G, Nissen HW (1996) Multi-perspective requirements engineering within NATURE. Requir Eng 1(3):157–169. https://doi.org/10.1007/BF01236425
84. Minker J (ed) (1988) Foundations of deductive databases and logic programming. Morgan Kaufmann, Burlington
85. Minker J, Seipel D, Zaniolo C (2014) Logic and databases: a history of deductive databases. In: Boyer RS, Moore JS (eds) Computational logic. Academic Press, Cambridge, pp 571–627
86. Minsky M (1974) A framework for representing knowledge. Technical report Memo n.306, MIT AI Laboratory
87. Motik B (2005) On the properties of metamodeling in OWL. In: The Semantic Web—ISWC 2005, 4th international semantic web conference, ISWC 2005, Galway, Ireland, November 6–10, 2005, Proceedings, pp 548–562
88. Mylopoulos J, Bernstein PA, Wong HKT (1980) A language facility for designing database-intensive applications. ACM Trans Database Syst 5(2):185–207
89. Mylopoulos J, Borgida A, Jarke M, Koubarakis M (1990) Telos: representing knowledge about information systems. ACM Trans Inf Syst 8(4):325–362
90. Mylopoulos J, Chaudhri VK, Plexousakis D, Shrufi A, Topaloglou T (1996) Building knowledge base management systems. VLDB J 5(4):238–263
91. Nejdl W, Dhraief H, Wolpers M (2001) O-Telos-RDF: a resource description format with enhanced metamodeling functionalities based on O-Telos. In: Proceedings of the K-CAP 2001 workshop on knowledge markup and semantic annotation, Victoria, BC, Canada, October 21, 2001
92. Nejdl W, Wolf B, Qu C, Decker S, Sintek M, Naeve A, Nilsson M, Palmér M, Risch T (2002) EDUTELLA: a P2P networking infrastructure based on RDF. In: Proceedings of the eleventh international world wide web conference, WWW 2002, May 7–11, 2002, Honolulu, Hawaii, pp 604–615
93. Nguyen V, Bodenreider O, Sheth AP (2014) Don't like RDF reification?: Making statements about statements using singleton property. In: 23rd International world wide web conference, WWW '14, Seoul, Republic of Korea, April 7–11, 2014, pp 759–770
94. Nicolaescu P, Rosenstengel M, Derntl M, Klamma R, Jarke M (2016) View-based near real-time collaborative modeling for information systems engineering. In: Advanced information systems engineering—28th international conference, CAiSE 2016, Ljubljana, Slovenia, June 13–17, 2016. Proceedings, pp 3–17. https://doi.org/10.1007/978-3-319-39696-5_1
95. Nikolaou C, Koubarakis M (2016) Querying incomplete information in RDF with SPARQL. Artif Intell 237:138–171
96. Nissen HW, Jeusfeld MA, Jarke M, Zemanek GV, Huber H (1996) Managing multiple requirements perspectives with metamodels. IEEE Softw 13(2):37–48

97. Nissen HW, Schmitz D, Jarke M, Rose T, Drews P, Hesseler FJ, Reke M (2009) Evolution in domain model-based requirements engineering for control systems development. In: RE 2009, 17th IEEE international requirements engineering conference, Atlanta, GA, USA, August 31–September 4, 2009, pp 323–328
98. Ntantouris K (1993) Programmatic query interface and query processing for the Telos language. Master's thesis, University of Crete and ICS-FORTH, Greece
99. Oldman D (2012) The British Museum, CIDOC CRM and the shaping of knowledge. http://www.oldman.me.uk/blog/. Accessed 30 Jan 2020
100. Pan JZ, Horrocks I (2001) Metamodeling architecture of web ontology languages. In: Proceedings of SWWS'01, the first semantic web working symposium, Stanford University, California, USA, July 30–August 1, 2001, pp 131–149
101. Pan JZ, Horrocks I (2003) RDFS(FA) and RDF MT: two semantics for RDFS. In: The semantic web—ISWC 2003, second international semantic web conference, Sanibel Island, FL, USA, October 20–23, 2003, Proceedings, pp 30–46
102. Pan JZ, Horrocks I (2007) RDFS(FA): connecting RDF(S) and OWL DL. IEEE Trans Knowl Data Eng 19(2):192–206
103. Pan JZ, Horrocks I, Schreiber G (2005) OWL FA: a metamodeling extension of OWL DL. In: Proceedings of the OWLED*05 workshop on OWL: experiences and directions, Galway, Ireland, November 11–12, 2005
104. Perry M, Jain P, Sheth AP (2011) SPARQL-ST: extending SPARQL to support spatiotemporal queries. In: Ashish N, Sheth AP (eds) geospatial semantics and the semantic web–foundations, algorithms, and applications, Semantic web and beyond: computing for human experience, vol 12. Springer, Berlin, pp 61–86
105. Petit M, Dubois E (1997) Defining an ontology for the formal requirements engineering of manufacturing systems. Springer, Berlin, pp 378–387
106. Plexousakis D (1993) Semantical and ontological consideration in Telos: a language for knowledge representation. Comput Intell 9:41–72
107. Plexousakis D (1995) Compilation and simplification of temporal integrity constraints. In: Rules in database systems, second international workshop, RIDS '95, Glyfada, Athens, Greece, September 25–27, 1995, Proceedings, pp 260–276
108. Plexousakis D, Mylopoulos J (1996) Accomodating integrity constraints during database design. In: Advances in database technology—EDBT'96, 5th international conference on extending database technology, Avignon, France, March 25–29, 1996, Proceedings, pp 497–513
109. Quillian MR (1966) Semantic memory. Technical report AFCRL-66-189, Bolt Beranek and Newman Inc
110. Quix C, Pascan M, Roy P, Kensche D (2010) Semantic matching of ontologies. In: Proceedings of the 5th international workshop on ontology matching (OM-2010), Shanghai, China, November 7, 2010. http://ceur-ws.org/Vol-689/om2010_poster10.pdf. Accessed 30 Jan 2020
111. Ram S, Liu J (2006) Understanding the semantics of data provenance to support active conceptual modeling. In: Active conceptual modeling of learning, Lecture notes in computer science, vol 4512, pp 17–29. Springer
112. Ramesh B, Jarke M (2001) Toward reference models of requirements traceability. IEEE Trans Software Eng 27(1):58–93
113. Reiter R (1988) On integrity constraints. In: Proceedings of the 2nd conference on theoretical aspects of reasoning about knowledge, Pacific Grove, CA, March 1988, pp 97–111
114. Robinson WN (2009) Monitoring requirements development with goals. In: Jeusfeld MA, Jarke M, Mylopoulos J (eds) Metamodeling for method engineering. MIT Press, Cambridge, pp 257–295
115. Robinson WN, Pawlowski SD (1998) Surfacing root requirements interactions from inquiry cycle requirements documents. In: 3rd International conference on requirements engineering (ICRE '98), putting requirements engineering to practice, April 6–10, 1998, Colorado Springs, CO, USA, Proceedings, pp 82–89
116. Robinson WN, Pawlowski SD (1999) Managing requirements inconsistency with development goal monitors. IEEE Trans Softw Eng 25(6):816–835
117. Ross DT (1975) Structured analysis (SA): a language for communicating ideas. IEEE Trans Softw Eng SE–3:16–34
118. Shrufi A (1994) Performance of clustering policies in object bases. In: Proceedings of the third international conference on information and knowledge management (CIKM'94), Gaithersburg, MD, November 29–December 2, 1994, pp 80–87
119. Shrufi A, Topaloglou T (1995) Query processing for knowledge bases using join indices. In: CIKM '95, Proceedings of the 1995 international conference on information and knowledge management, November 28–December 2, 1995, Baltimore, MD, USA, pp 158–166
120. Snodgrass RT, Ahn I (1985) A taxonomy of time in databases. In: Proceedings of the 1985 ACM SIGMOD international conference on management of data, Austin, TX, May 28–31, 1985, pp 236–246
121. Sripada SM (1988) A logical framework for temporal deductive databases. In: Fourteenth international conference on very large data bases, August 29–September 1, 1988, Los Angeles, CA, USA, Proceedings, pp 171–182
122. Staudt M, Jarke M (1996) Incremental maintenance of externally materialized views. In: VLDB'96, Proceedings of 22th international conference on very large data bases, September 3–6, 1996, Mumbai (Bombay), India, pp 75–86
123. Stemple D, Simon E, Mazumdae S, Jarke M (1990) Assuring database integrity. J Database Adm 1:12–26
124. Tansel AU, Clifford J, Gadia SK, Segev A, Snodgrass RT (eds) (1993) Temporal databases: theory, design, and implementation. Benjamin/Cummings, San Francisco
125. Topaloglou T (1993) Storage management for knowledge bases. In: CIKM 93, Proceedings of the second international conference on information and knowledge management, Washington, DC, USA, November 1–5, 1993, pp 95–104
126. Topaloglou T, Illarramendi A, Sbattella L (1992) Query optimization for KBMSs: temporal, syntactic and semantic transformations. In: Proceedings of the eighth international conference on data engineering, February 3–7, 1992, Tempe, Arizona, pp 310–319
127. Topaloglou T, Mylopoulos J (1996) Representing partial spatial information in databases. In: Conceptual modeling—ER'96, 15th international conference on conceptual modeling, Cottbus, Germany, October 7–10, 1996, Proceedings, pp 325–340
128. Tran T, Haase P, Motik B, Grau BC, Horrocks I (2008) Metalevel information in ontology-based applications. In: Proceedings of the twenty-third AAAI conference on artificial intelligence, AAAI 2008, Chicago, IL, USA, July 13–17, 2008, pp 1237–1242
129. Vegetti M, Leone HP, Henning GP (2011) PRONTO: an ontology for comprehensive and consistent representation of product information. Eng Appl AI 24(8):1305–1327
130. Weyhrauch RW (1980) Prolegomena to a theory of mechanized formal reasoning. Artif Intell 13(1–2):133–170
131. Yu E, Giogini P, Maiden N, Mylopoulos J (2011) Social modelling for requirements engineering. MIT Press, Cambridge
132. Yu ESK (1997) Towards modeling and reasoning support for early-phase requirements engineering. In: 3rd IEEE international symposium on requirements engineering (RE'97), January 5–8, 1997, Annapolis, MD, USA, pp 226–235

133. Yu ESK, Mylopoulos J (1994) From E-R to "a-r" - modelling strategic actor relationships for business process reengineering. In: Entity-relationship approach—ER'94, business modelling and re-engineering, 13th international conference on the entity-relationship approach, Manchester, UK, December 13–16, 1994, Proceedings, pp 548–565

134. Zachman JA (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292