1

An Overview of Modularity

Christine Parent[1] and Stefano Spaccapietra[2]

[1] HEC ISI, Université de Lausanne, Switzerland
christine.parent@epfl.ch
[2] Database Laboratory, Ecole Polytechnique Fédérale de Lausanne, Switzerland
stefano.spaccapietra@epfl.ch

Summary. Modularization is a familiar concept in IT, widely used in e.g. software development. It has been somehow neglected in knowledge management. Database research has only marginally addressed the topic, mainly for the development of cooperative database systems. Instead, research on ontologies, seen as the knowledge provider for the semantic web, has significantly invested on modularization techniques, seen as a key capability in the current efforts towards scalable solutions that will enable ontologies to grow to the huge size that we can foresee in real world future applications. Many different approaches exist to tackle ontology modularization, corresponding to different goals or building on different initial hypotheses. This chapter aims at clarifying the vision of the domain by providing a detailed yet generic characterization of the issues and solutions related to the various facets of modularization.

1.1 Introduction

Complexity is an almost pervasive feature of modern life. Computerized knowledge management is no exception. We, both as individuals and as societies, increasingly rely on knowledge stored in computers worldwide to acquire the pieces of information we need to govern our personal behavior. From politicians to technical directors, managers in charge of societal problems (e.g. environmental issues) similarly rely on accurate extraction and synthesis of relevant knowledge. Unfortunately, knowledge repositories have grown beyond our capacity to apprehend their content. We are constantly exposed to knowledge overdoses, and one of the critical success factors nowadays is the capability to get rid of all knowledge that is unnecessary to the task at hand, so that the knowledge to be considered downsizes to a manageable amount. Accordingly, the research community is increasing its effort towards the specification of frameworks and techniques that may help in downsizing knowledge.

Modularization is one of the approaches to achieve such a result. In its most generic meaning, it denotes the possibility to perceive a large knowledge repository (be it an ontology or a database) as a set of modules, i.e. smaller repositories that, in some way, are parts of and compose the whole thing. Modularization materializes the long-established complexity management technique known as divide

H. Stuckenschmidt et al. (Eds.): Modular Ontologies, LNCS 5445, pp. 5–23, 2009.

and conquer. It is routinely used in various areas of computer science, such as algorithms, software engineering, and programming languages. In software engineering, for example, module is one of the terms used to denote a software component that is designed to perform a given task and is intended to interact with other modules within a larger software architecture. In programming languages, module sometimes denotes an encapsulation of some data. Easiness of understanding and potential for reusing are among the main claimed benefits of these approaches.

In the knowledge management domain, modularization opposes the initial and traditional vision of databases as an integrated and monolithic collection of strongly interrelated data. This explains that the database community basically discarded modularization. Even federated databases were seen as a way to build larger databases, not as a way to address complexity. Only few researchers (e.g., [20] [17]) addressed for example the issue raised by the complexity of handling database schemas having a very large number of object and relationship types. The work in [20] proposes design techniques supporting multiple levels of details (i.e. more or less detailed representations of the database schema), while the work in [17] develops a design methodology built on the idea of manually identified design units, such that each unit covers a part of the application domain.

In contrast, concerns about scalability and interoperability of ontologies have generated a significant interest in modularization from the semantic web community. Ontologies are the knowledge repositories that enable meaningful inter-agent communication in semantic web frameworks. To support the huge number of services that will be developed in the semantic web, ontologies will grow larger and larger to accommodate as much knowledge as possible. The universal, all-encompassing ontology is the dream behind this trend. In more realistic approaches, ontologies are targeted to cover a specific domain. But even these domain ontologies may grow in size beyond human mental capacity and pose a critical challenge to reasoning tools. An example is the ProPreo (Proteomics data and process provenance) ontology, which in 2006 contained 18.6 million assertions for 390 classes and 3.1 million instances [16]. Moreover, the domain covered by an ontology may be expanded to be of use to a wider community and to promote interdisciplinary research. For example, an ontology for biology (e.g. the Gene ontology [18]) may eventually combine with an ontology for medicine to provide for better interoperation of specialists in the two domains. A rising number of instances may be the growing factor for an ontology used to cover an increasing number of similar applications in a given domain. For example, ontologies for tourism will grow beyond limits if they are meant to hold instances about any possible tourist destination worldwide. There clearly is a scalability issue, and modularization is an appealing solution as it aims at replacing oversized ontologies with smaller ones. The issue originates in both growth in concepts and growth in instances, although the two phenomena do not raise exactly the same issues, and may call for different solutions.

Conversely, there is also a definite need from applications to gather knowledge from several, not just one, ontological sources. It is known that, when knowledge is distributed, the idea to collect all knowledge into a single repository (i.e. the integration approach) is very difficult to implement, because of semantic heterogeneity

calling for human processing. Instead, implementing an interoperability framework that supports meaningful knowledge exchange among the sources is an easier and more efficient approach. The definition of such a distributed collaborative paradigm that enables collaboration among existing ontologies (now seen as modules of the larger virtual ontology that is built by the collaboration and corresponds to application requirements) is a challenging research stream that comes under the umbrella of modularization and has already generated several competing approaches.

The chapter hereinafter continues with the overview of modularity ideas. The next section discusses goals that may be assigned to modularization. A given goal may influence the way the concept of module is defined. Section 1.3 focuses on the concept of module per se. Section 1.4 looks at the strategies that may be used to produce modules using semantic or syntactic criteria, with human-driven or automatic approaches. Section 1.5 discusses the correctness criteria that should characterize modular ontologies. Section 1.6 analyzes issues in composing existing ontologies as modules for larger repositories. Module interconnection, essential to distributed reasoning, is discussed in Section 1.7. Section 1.8 introduces multi-perception features. Section 1.9 concludes the chapter.

1.2 Goals of Ontology Modularization

Modularization in itself is a generic concept that is intuitively understood as referring to a situation where simultaneously a thing (e.g. an ontology) exists as a whole but can also be seen as a set of parts (the modules). How modularization is approached and put into practice, as well as what are the advantages and disadvantages that can be expected from modularization greatly depend on the goals that are pursued through modularization. This section briefly reviews some frequently quoted possible goals for modularization of ontologies. A complementary discussion, in Chapter 3 of this book, focuses on measurable goals that can be used as evaluation criteria to assess the relative quality of modularization techniques versus expected benefits.

Scalability for querying data and reasoning on ontologies

This is an all-embracing goal, which mainly sees modularization as a way to keep performance of ontology services at an acceptable level. Performance concerns may be related to query processing techniques, reasoning engines, and ontology modeling and visualization tools. Database systems have basically solved performance issues by developing efficient storage and querying optimizers (e.g. dynamic hashing and multidimensional indexing) to provide fast retrieval and update of data. As they do little reasoning, this task is not problematic for DBMS. Instead, for ontological reasoners the complexity of their task is directly related to the number of axioms and facts they have to explore before coming to a conclusion. While different reasoners are available, supporting different formalisms from RDF to OWL, they are known to perform well on small-scale ontologies, with performances degrading rapidly as the size of the ontology (in particular, the number of instances) increases. Keeping

ontologies smaller is one way to avoid the performance loss, and modularization is a way to replace an ontology that tends to become oversized by smaller subsets. Modularization fulfills the performance goal if, whenever a query has be evaluated or an inference performed, this can be done by looking at just one module, rather than exploring the whole ontology. But if most queries call for distributed reasoning to exploit knowledge from several modules, it remains to be demonstrated that the overall time for coming up to a result is less than the time required for the same task executed against a single ontology holding all the knowledge kept by the modules in the network. Thus, the driving criterion for modularization aiming at performance is to localize the search space for information retrieval within the limits of a module. Implementing a good distribution of knowledge into modules requires knowledge about the search queries that are expected. This type of knowledge can be extracted a posteriori from observing information queries over some period of time. Predicting this knowledge a priori would be more effective, but is difficult to achieve.

Scalability for evolution and maintenance

Achieving this goal is an open issue for both ontologies and databases. Here, the driving criterion for modularization is to localize the impact of updating the knowledge repository within the limits of a module. Implementing a good knowledge distribution requires an understanding of how updates propagate within the repository. It also requires knowledge on the steadiness of the information it contains. Steadiness here is meant to denote the unlikeliness of an evolution. A possible factor for steadiness is the confidence level attached to information. How confidence levels, and more generically steadiness indicators, are acquired remains an open issue for research.

Complexity management

While scalability usually refers to system performance in processing user queries, performing reasoning tasks, and visualizing results, a similar issue can be raised regarding the design of the knowledge repository. The larger the repository is, in terms of objects and relationships or in terms of axioms and rules, the more difficult is ensuring the quality and accurateness of the design, especially if the designers are humans (as it is still the case). We refer here to the issue of making the design task intellectually affordable for designers, thus creating the best conditions leading to a semantically correct design that fulfills the informational requirements assigned to the ontology/database at hand. Quoting Stuckenschmidt and Klein [15], "ontologies that contain thousands of concepts cannot be created and maintained by a single person". The best way to help designers is to reduce the size of the design problem. That is what modularization achieves. Let designers design modules of a size they can apprehend, and later either integrate these modules into the final repository or build the relationships among modules that support interoperability. This is a typical application of the divide-and-conquer principle.

Notice that the facilities provided by DL reasoners to check the consistency of specifications address a different issue: correctness. These facilities are essential to

guarantee the quality of a given design, but they only lift a specific problem from the concerns of the designer. They do not make the design task easier.

Understandability

Another challenge is to be able to understand the content of an ontology (and of the schema of a database), an obvious prerequisite to the ability to use them. This is very similar to the previous challenge, but has to do with the usage phase rather than the design phase. Whether the content is shown in visual or textual format, understanding is easier if the repository is small, for example just a module. Smaller repositories are undoubtedly preferable if the user is a human being (as is visual versus textual representation). We believe intelligent agents navigating through the Web-services space also benefit from smaller repositories. The agent might have to explore (browse) the schema or ontology rather than just querying it. Browsing a few nodes is faster than browsing a huge number of nodes. Size, however, is not the only criterion that influences understandability. The way it is structured contributes to improving or decreasing understandability, as it has been extensively shown in the database modeling domain (i.e., a relational schema is much harder to understand than its equivalent entity-relationship or UML schema).

Context-awareness and Personalization

Context is a pervasive concept that also applies to knowledge repositories, and personalization is a specific kind of context use driven by the user profile. Efforts towards context-dependent databases and ontologies are being pursued. Context-awareness means that the database/ontology system knows that different subsets of its content are relevant for different contexts. This implies that the creation of knowledge, its storage and usage have to take context into account. What exactly makes up a context is still an open issue and there is no universal answer to the question. Contexts are themselves context-dependent, i.e. they convey the specific requirements of the applications using the contextual knowledge. One particular kind of context is ownership of information, known to be an important factor to be taken into account when organizing distributed cooperative systems. This may also apply to ontologies, although most of them are seen as publicly available resources. Ownership in these cases provides the rationale for building a modular ontology. Ownership information can also be attached to existing ontologies in view of integrating them into a modular ontology where modularity is based on personalization.

Reuse

Reuse is a well-know goal in software engineering. Reuse is most naturally seen as an essential motivation for approaches aiming at building a broader, more generic repository from existing, more specialized repositories. However, it may also apply to the inverse approaches aiming at splitting a repository into smaller modules. In this case, the decomposition criterion would be based on the expected reusability

of a module (e.g., how well can the module fill purposes of various applications?). Reusability emphasizes the need for rich mechanisms to describe modules, in a way that maximizes the chances for modules to be understood, selected and used by other services and applications.

1.3 The Essence of Modules

Ontologies and databases are not arbitrary collections of concepts/objects, relations, properties, axioms and instances. The gathering and definition of their elements fulfills a purpose, be it to make available to potential users the knowledge and terms covering some domain or to represent a subset of the real world that is of interest to an organization and its applications. Similarly, the process of decomposing a (database or ontology) repository into modules has to rely on some meaningful principles to produce modules that make sense. Each module is expected to show a similar unit of purpose, gluing together those elements from the global repository that participate to a given goal (which may be seen as a sub-goal of the goal of the global repository). For example, an ontology module would represent an agreed conceptualization of a sub-domain of the domain of the ontology [3]. Such modules make sense for, and can be separately used by the community of users only/mainly interested in the sub-domain rather than the whole domain covered by the overall ontology. For example, an enterprise database or ontology may include modules for payroll, for customer support, for accounting, for marketing, etc. An ontology for historians may have modules for chronological subsets (e.g., pre-history, ancient, medieval, renaissance, and contemporary periods) and at the same time have modules by geographical area (e.g., European, American, Asian, Middle-East, African, Oceanic history).

For knowledge management consistency, a database module shall be a (smaller) database, and an ontology module shall be a (smaller) ontology. The benefit is that the same software system can handle the whole repository and its modules. From a pragmatic viewpoint, modules of the same database or ontology are most likely defined and used according to the same model and formalism (e.g., they all follow OWL-DL specifications or they all are relational databases), but this is not a mandatory feature. The same rationale applies when a module is built by extracting some content from the ontology/database, without running a global decomposition process.

The potential specificity of modules, making them different from the whole, is that a module is aware of being a subset of a broader knowledge organization, and therefore knows it may interact with other modules to get additional knowledge in a sub-domain closely related to its own sub-domain. Thus, on the one hand (see Part II of this book) each module interacts with the whole in a part/whole relationship (leading to a focus on extraction techniques), while on the other hand (see Part III of this book) modules may have a collective societal behavior, where the focus is on cooperative techniques. Using an arithmetic metaphor, we can summarize this as:

module = a (smaller) ontology + inter-modules links

The composition/interrelation duality supports two different contexts that people associate with the idea of modularization. In one context modularization refers to the process of turning several self-standing repositories into either a collection of interrelated and interoperating modules that together form a broader repository (cooperative or distributed approaches), or a single integrated broader repository (integration approaches). In this context the modules are simply the existing repositories. The issue is not how to delimit them, but to enable their interoperation or integration. In the other context, modularization refers to the process of creating one or more modules from an existing repository. Many different ways to do that have been proposed. The next section discusses the issue.

1.4 Modularization Strategies

Defining the strategy and assumptions that rule how knowledge in a repository is distributed into modules is a fundamental task in approaching modularization. Strategies may rely on semantic or syntactic criteria, informal or formal specifications, vary in degree of automation and vary in terms of targeted goal. The latter includes the issue whether the coexisting modules created from a given ontology have to be disjoint or may overlap. This section first discusses disjointedness, then moves to examining the different strategies.

Disjoint or overlapping modules

One of the basic alternatives in fixing a modularization strategy is whether the strategy should enforce disjointedness among the modules of an ontology, or allow them to overlap. The main advantage of disjointedness is easier consistency management. Contrary to overlapping modules, disjoint modules cannot contradict each other. Hence a set of consistent disjoint modules is always consistent. Enforcing disjointedness entails that distribution of knowledge into the modules has to be controlled by the system. The system may have complete control, meaning that it automatically performs distribution of knowledge using some partitioning algorithm. Alternatively, users may somehow fix the distribution, in which case two options exist to enforce disjointedness. One option lets the system check that modules are kept disjoint by users (disjointedness is explicitly enforced), rejecting knowledge insertions that violate disjointedness. In the other option whenever the user allocates a given piece of knowledge to multiple modules, the system creates as many copies as needed to store one in each targeted module and does not retain the fact that copies exist (disjointedness is implicitly assumed). Otherwise stated, the system iterates the insertion but ignores the resulting duplication (each copy is considered as a separate and unique piece of knowledge).

On the other hand, the advantage of overlapping is more flexibility in knowledge distribution decisions and more functionalities in using modules.Assume, for example, that a modular ontology about *History* is required. Possible targeted modules could include a module about *Middle Age* and another one about *Italy*. Clearly, a

number of concepts, e.g. the *History of Medieval Italy* concept, will be relevant for both these modules. Each of these concepts is candidate for allocation to multiple modules. In a disjointedness-oriented strategy, one may simply create as many copies of a concept as needed to allocate one copy to each candidate module, and then forget about the duplication, i.e. consider each copy as a separate piece of knowledge, independent from the other copies of the same concept. In an overlapping-oriented strategy, concepts may be directly allocated to multiple modules (without being duplicated) and the system keeps awareness of this multiplicity. Using this awareness the system can let users navigate from one module to another, i.e. from one instance of a concept in one module to another instance of the same concept in another module.

Issues related to such knowledge overlapping form a research domain per se. Practically overlapping modules may be implemented in two different ways. The modules may be created to share the same interpretation domain, as that may be the case of ontologies and databases created as modular since the beginning, like Mads (see Chapter 5 in this book). Alternatively, each module may have its own interpretation domain, and these domains be linked, two by two, by binary relations, like in distributed ontologies (see Chapter 12 in this book). Remark that when real world entities are described at the same level of granularity in two modules, the binary relation should reduce to an injective function. A "full" binary relation allows mapping an instance (object) of a module to a set of instances (objects) of the other module, which is very different from a simple overlap.

Semantics-Driven Strategies

The most intuitive approach to modularization is to let it be driven by the semantics of the application domain and the way the application perceives identifiable subdomains which could stand by themselves and handle a subset of the application requirements. This has been stated as a "semantic correctness" requirement for modularization: "A module should make sense" for the users [7].

Semantic interpretation of knowledge relies on human appreciation, i.e. on the expertise (knowledge of the domain covered by the ontology and knowledge of application requirements for a database) of the persons in charge of creating and updating the content of the repository. The role of the system is usually limited to recording the allocation of knowledge items to the modules, ensuring that every single content component (e.g., concept, role, axiom, and instance for ontologies) is allocated to at least one module (assuming the repository is fully distributed over its modules). Modular ontology approaches such as Cyc and Mads (see Chapter 5 in this book) follow this path. In modular ontologies the specification of a new piece of knowledge includes its allocation to modules. We can say these are manual strategies, as modularization is crafted piece by piece by users. Cyc nevertheless offers tools for automatic placement of a new assertion in the most relevant module, as determined by examining the concepts in the assertion. This assistance to users may help in avoiding erroneous distribution of knowledge, such as allocating a concept to a module and its properties to another module. Some proposals address this concern

through the definition of semantic units that group knowledge into atomic pieces whose components are likely to be kept together when defining the distribution into modules. For example, Rector [12] proposes a semantic reorganization ("normalization") of the hierarchy of concepts in order to lead to a more semantic partition of the ontology. This normalization is proposed as a preliminary to the extraction of modules from the ontology.

Some approaches, where ontology and module creation are desynchronized, are also driven by semantics and aim at bringing more automation into the module elaboration process. It is the case of most methods that extract modules out of a classic, non modular, ontology. The idea of these semi-automatic approaches is simply to compute the desired modules from initial specifications provided by users. The human designer role is to specify the desired result and the system role is to actually build the result. A simple and quite traditional way to specify the desired module is by writing a query that selects a subset of the ontology. The evaluation of the query produces the module, extracted from the ontology [21] [9]. The newly extracted module is materialized, and is independent from the ontology.

Structure-Driven Strategies

Other modularization strategies purposely avoid considering semantic aspects. They look at an ontology as a data structure consisting of a graph of interconnected nodes (whatever their semantics) and use graph decomposition algorithms to create subsets (the modules) based on structural properties that a subset has to comply with. For these strategies, a module is a set of concepts (and roles) that are more tightly inter-related together than to other concepts that are not in the set. What tightly interrelated exactly means varies from one proposal to another one. Hence different strategies belong to this category, each one characterized by the specific structural criteria they use for the decomposition. A significant example is Chapter 7 in this book, which presents in detail a structure-based algorithm that produces a partition of an ontology into modules. Such semantic unawareness allows the algorithm to run automatically without relying on human input. This is essential whenever the goal is to allow computerized agents to produce ontological modules without calling for human interaction.

However, these algorithms can also be tuned by initially specifying the application requirements that may influence the decomposition criteria. Typically, the module designer identifies within the ontology a small set of kernel elements (i.e., the concepts and roles most important to the new module) and in a second step the system iteratively builds a module from the kernel elements by adding other elements attached to the selected ones. This kind of "growing" algorithm needs some criterion to determine how far it should go in looking for attached elements (e.g., using something like a threshold for distance between the kernel elements and the other elements). Moreover, a strategy has to be defined to instruct the system on what to do whenever two related elements are allocated to different modules. Strategies presented in Chapters 8 and 9 build on the concept of kernel elements and associated structural criteria to automatically extract modules from an ontology.

In the database domain, fully automatic decomposition has been investigated, for example, to allocate data within a distributed database framework based on load balancing algorithms, i.e. a criterion driven by system performance concerns. The algorithms produce a data distribution schema that shows which data fragments have been defined and where they are stored [4].

A different category that may also be classified as structure-driven groups the strategies to build a modularized ontology from existing ontologies considered as modules of the targeted all-embracing ontology. The structural criterion on which modules are defined is simply to take the existing pieces as modules and interconnect them to form the new ontology. One could say that modules pre-date the ontology, while the other strategies derive modules from the ontology. Representative of these strategies are distributed ontologies (as discussed in Part III of this book) and federated databases [13]. Strategies differ from each other on the techniques they use to interrelate the module (see Section 1.7 hereinafter).

Machine Learning Strategies

An alternative to human-driven modularization, also not needing an explicit modularity criterion, is computed modularization based on some machine learning knowledge acquisition process. For example, an analysis of the queries that are addressed to the repository and of the paths within the repository that are used to respond to queries may be used, considering the frequency of paths and their overlapping, to determine the clustering rule that produces the optimal decomposition. We use the term machine learning in a very loose sense including any one of the many available techniques that a system may use to infer knowledge from any data log. Data mining and clustering analysis are candidate techniques. These strategies differ from previously discussed strategies in the sense that the knowledge distribution they produce is purely based on observing interactions that are relevant for knowledge management, not on a human or human-driven dedicated specification (as in semantics-driven strategies), nor on structural properties of the ontology (as in structure-driven strategies). To the best of our knowledge, machine learning strategies have not yet been explored for ontology modularization.

Monitoring Modularization and Making it Evolve

At last, we mention here the problem of updating the distribution of knowledge into the modules, irrespectively of the strategy used to create them in the first round. Whatever the modularization strategy, it is important to evaluate and monitor the reliability and efficiency of the knowledge services provided by modules. Knowledge distributions that, once in use, prove to be unsatisfactory or inefficient shall be adjusted or corrected based on the outcome of monitoring tasks. The outcome may for example identify a subset of a module that is frequently needed by another module, thus suggesting that this subset should more properly be allocated to the other module. Similarly, unused parts of modules may simply be deleted or transferred to an archival module. Whenever queries to a module always require additional knowledge

from another module, merging of the two modules might be advisable. Conversely, if query patterns show independent usage of separate parts of a module, a splitting of the module is worth considering.

1.5 Module Correctness

A very important concern when decomposing a repository is the correctness of the modularization. A basic requirement for a modular ontology to be correct is that each module is correct.

To evaluate correctness, first we have to define what the term means. An obvious correctness criterion is syntactic correctness, that is to say the knowledge specifications follow the rules of the language and data model they rely on. Enforcing this kind of correctness is usually done by the underlying tools, e.g. ontology editor tools and schema design tools for databases. At the instance level, checking the consistency between instances and specifications is done by the reasoners and the DBMS.

In the previous section we quoted the definition by Grau et al. of semantic correctness as referring to the fact that a module should make sense. Obviously there is no way for a system to check such semantic correctness, in particular in the ontology world where ontologies (unlike most of databases) are not confined to hold the knowledge required by a specific application.

The same authors also defined the logical correctness concept, where logical refers to the use of logic. Their definition states that a module is logically correct if and only if, as long as it involves only the vocabulary of the module, the knowledge that can be inferred by the module is the same as the one that could be inferred by the initial ontology (see Chapter 6 in this book).

The links that interrelate the modules of an ontology must be taken into account when computing the inferences generated by a module. This means that knowledge targeted by the links should exist (no pending references) and be consistent with the knowledge inside the module, i.e. augmenting the knowledge within the module with the knowledge external to the module should not result in unsatisfiability. However, recent work shows that a certain level of localized inconsistency between external and internal knowledge can be allowed in the sense that queries that do not involve the inconsistent subset can be evaluated with no problem [2].

In case an existing ontology is split into modules by partitioning, the collection of modules should satisfy specific correctness criteria that assess the preservation of information in the partitioning process. Information preserving may be defined as the fact that the result of any query addressed to the collection is logically the same as the result of the query addressed to the original ontology. This is sometimes referred to as the fact that the whole ontology is a conservative extension of each one of its modules. Chapters 2 and 6 in this book build on this conservative extension idea to formally define properties of modules generated using alternative modularization schemes. This allows ontology designers to choose a modularization approach that best matches the properties they want to be achieved.

Information preserving can also be statically defined as the fact that, when recomposing the original ontology from the modules (using some composition rules), what is obtained is equivalent to the original piece, nothing less, and nothing more[1]. Depending on the modularization rules used, it may be possible to guarantee that, if rules are obeyed, the modularization they generate is information preserving. If information loss cannot be avoided, an estimation of the information loss can be a very useful add-on for the query answering techniques [8].

1.6 Ontology Composition

The idea to collect data/knowledge from several existing repositories to form a new, real or virtual repository designed for new applications has arisen as soon as computer networks have become operational. Current economic trends have turned the idea into a necessity. In the database domain the idea has become very popular in the research community in the 80'ies and 90'ies under the label "federated databases" [13], denoting new databases built by integrating sets of existing databases. The main research issue from the semantic viewpoint has been data and schema integration, i.e. how to build a homogeneous description (the integrated schema) of the whole set of data available in the existing databases, despite their syntactic and semantic heterogeneity [10][14]. Later, the trend turned to alternative solutions that would support integrated data management without needing an integrated schema. These approaches are frequently denoted as cooperative systems. Eventually, new cooperation paradigms have emerged in the framework of peer-to-peer systems[2]. As the name says, in these frameworks the component database systems (the peers) directly interact with each other whenever they need information they do not have. Different strategies use different heuristics for a peer to determine which other peers to contact, given that no central description is available to tell which peer holds the desired information. The latest trend build on memorizing the results of these exchanges to gradually build in each peer knowledge of where relevant information has been found. This trend is known as emergent semantics approaches [1][5].

In the ontology world, the same two approaches exist: Integration of existing ontologies into a global, classic, non modular, ontology is similar to the federated databases approach. On the other hand, distributed ontologies is similar to cooperative databases systems. A distributed ontology is a broader global ontology, covering a larger domain, which is built by somehow "composing" a set of existing ontologies. In this setting, from the modularization viewpoint the existing ontologies are pre-existing modules used to build a modular ontology. The goal is not to create a single big ontology (an effort that would probably fail for the same heterogeneity issues that made data integration into federated databases a dream rather than a reality) but to target a new modular and cooperative ontology, where the modules (existing

[1] This is analogous to information preserving rules in databases for the decomposition of a relational relation into a set of relations of higher normal form (known as the normalization process).

[2] `http://en.wikipedia.org/wiki/Peer-to-peer`

ontologies) are kept unchanged and acquisition of new knowledge is done within the modules rather than at the global ontology level. Part III in this book presents the best established composition approaches.

The process of composing the modules consists in identifying duplicated and complementary knowledge among modules, and then establishing appropriate links in between the identified pieces of knowledge. Duplicate knowledge can be linked using a multi-instantiation link, i.e. a link whose semantics says that the two linked elements contain each one a subset of instances that represent the same set of real world entities; or it can be linked using a "same as" role, whose semantics says that two different instances are in fact two representations of the same real world entity. Complementary knowledge is linked using inter-module roles, i.e. roles whose domain is in one module and whose range is in another module. For example, a Person concept in module M_i can be linked by a role to the Car concept in module M_j assuming that the intended semantics is that a person may own a car.

It is also possible to devise strategies to compose existing ontologies where some rules are used to redistribute or reallocate the content of modules within the global ontology. For example, the strategy could instruct the ontology system to remove duplicate content and replace it by inter-module links, in an effort to reduce the individual or the cumulative size of the modules. However, this would be against the autonomy of the modules, and should therefore applied with care, if ever.

As in federated databases, a typical issue in module composition/integration arises from the syntactic and semantic heterogeneity that may exist among multiple descriptions of the same concepts and roles in the different modules. Syntactic differences include heterogeneity of the underlying formalisms (e.g., one module uses a paradigm from logics while another module uses a conceptual data modeling paradigm), and of the languages used to define them (one module uses RDF, another one uses OWL-DL). Differences in representation of shared elements arise whenever the sub-domains covered by the modules are not mutually disjoint, a situation that is likely to be the norm rather than the exception. For example, one module holds road network knowledge while another one holds knowledge on roads as part of a public transport system, resulting in different representation requirements of roads in the two modules.

The whole ontology matching process has been and still is extensively discussed, e.g. in [6]. Part of the issue is the detection and processing of the semantic differences that otherwise obscure knowledge sharing. Indeed, corresponding elements, i.e. elements that at least partly represent the same set of real world entities or links, may differ at the terminological level (different names), descriptive level (different sets of properties), structural level (different constructs of the model are used to represent the elements), and semantic level (the elements describe different sets of real world entities or links). The process to detect the correspondences between the elements is referred to in the integration literature as similarity search or correspondence discovery.

When the goal is to create a global repository (as in federated databases) with a single description (the integrated schema), differences between the source descriptions have to be reconciled to enable generating the integrated element. This

reconciliation process is called conflict resolution. Similar reconciliation is nowadays proposed for integration of multiple ontologies into a single ontology. Integration research is not discussed in this book as its goal is opposite to modularization. Instead, distributed ontology approaches fully qualify as modularization techniques, and are reported in detail in Part III of this book. In distributed ontologies the only task is to put an inter-modules link between the corresponding elements. Terminological and descriptive differences do not matter. But the inter-modules links should support structural and semantic differences. They should allow to link elements of different kinds and they should express various kinds of semantic differences, i.e. the represented sets of real world entities/links are the same, or one is the subset of the other, or the two sets are overlapping, or the two sets are disjoint but related.

1.7 Links among Modules

In a modular ontology, each module is assumed to provide ontological services related to a given sub-domain of the global domain covered by the ontology as a whole. Basic experience in knowledge organization shows that very frequently it is impossible to partition a domain into disjoint sub-domains. Whatever the modularization principle is, some concepts inevitably end up belonging to multiple sub-domains. Most frequently, the way a concept is characterized changes from sub-domain to sub-domain. Usually, some characteristics, referred to as inherent or essential to the concept, are the same whatever the sub-domain, and many others, referred to as incidental to the concept, are specific to the sub-domain they relate to. This means that complementary and duplicated knowledge about a concept exists in different modules. Users of a module may be interested in accessing the entirety of the knowledge about a concept available within the modular ontology. Similarly, given the availability of two modules, users of one module may be interested in connecting concepts of this module to other concepts in the other module whenever they see a meaningful and useful connection between the two. To gain direct access to knowledge external to a module, inter-module links are established. They create a navigational path from module to module that makes the knowledge in the target module available to users of the initial module.

Inter-module links may be seen as an integral part of the module they stem from, or as a construct that is superimposed onto the modules and external to them. In the first case a module per se is an incomplete ontology given that its links to external elements appear as pending links when the module is considered in isolation, for instance for consistency checks. In the second case, links can be considered as externally stored, possibly in a separate dedicated repository, and each module taken in isolation is an autonomous ontology. This solution has for example been adopted in distributed database systems, where tables with pairs of oids have been used to store the instance-level correspondence between related objects in different databases. Managing links externally to modules allows creating different ontologies from the same set of modules by defining alternative sets of links.

Inter-module links may be of different types: relationships or multi-instantiation links. Inter-module relationships may relate two concepts of two different modules, concepts that describe different sets of real world entities that are related by a relationship. For instance, one module may describe cars and another one models of cars. An inter-module relationship or role, e.g. hasModel, can be defined between the two concepts. Its instantiation will have also to be defined. Chapter 11 presents such a solution based on the $\mathcal{E}$–connections technique which is a method for combining logical languages and in particular OWL-DL ontologies. The ontologies are related by inter-module roles that link concepts of two different modules. The method supports an extended version of the Tableau algorithm for reasoning on distributed ontologies of kind $\mathcal{E}$-connections.

Inter-module multi-instantiation links are links that relate two elements that belong to different modules and describe, at least partially, the same set of real world entities or links. The most usual kind of inter-module multi-instantiation links is the is-a link between two concepts, for example Car in one module and Vehicle in another one. C-OWL calls this kind of link, bridge rules. More generally, an inter-module multi-instantiation link may express sub-set (is-a), equivalence or overlapping. Chapters 5 and 12 support this kind of links. Notice that the linked elements may be two concepts, two roles, a concept and a data-type, and a composition of roles and a role.

In most distributed ontology approaches, inter-module multi-instantiation links relate elements of the same kind, i.e. two concepts, two roles or two individuals. But more complex links are needed, for instance a role may correspond to a composition of roles (e.g. *isUncle* corresponds to *isParent.isBrother*), or a concept may correspond to the union of several concepts (e.g. *Parent* corresponds to $Father \cup Mother$).

As in distributed ontologies each module is an autonomous ontology, it has its own interpretation domain, which is disjoint from all the other ones. Therefore, inter-modules links of kind is-a, or more generally multi-instantiation links, require a mapping between the interpretation domains of any two linked modules. In most of the approaches the mapping is defined by a binary relation. That allows to assert that an instance of one module corresponds to any number of instances of the other module. Consequently, the semantics of an inter-module is-a link depends on the underlying binary relation. For instance in Chapter 12, a bridge rule $O1{:}C1 \xrightarrow{\sqsupseteq} O2{:}C2$ does not always say that any instance of *C1* corresponds to an instance of *C2*. If the binary relation leaves out some *C1* instances, that would not be the case: The bridge rule constrains only the instances of *C1* and *C2* that participate in the binary relation.

When a distributed ontology contains inter-modules multi-instantiation links, an important question is the consistency of the related modules. As already discussed in Section 1.5, knowledge sharing by different modules is prone to inconsistencies (unless a tight control and coordination of the evolution of the modules are enforced). This calls for specific consistency-preserving mechanisms. A distributed ontology will be consistent only if each component ontology is itself consistent and all the

multi-instantiation links are consistent with the component ontologies they are linking. For more general distributed reasoning issues see Chapter 12 in this book.

Another facet of the consistency issue about links is how up-to-date they are. Modules, as ontologies in general, are expected to evolve in a desynchronized and decentralized way. If a link has been identified and described at time t, can we guarantee that it is still valid for use at some later time $t' > t$? For example, assume a link specifying that a concept A in module M_i is equivalent to a concept B in module M_j. The evaluation of a query to M_i that references A may need to traverse the link between M_i and M_j to find out what M_j knows about B. If, after the link has been established, a role r(B, D) is added to M_j which somehow constrains B, it may be that the new constraint results in the fact that the equivalence does not hold anymore. To prevent inconsistencies, either an update propagation mechanism is developed to keep all links up-to-date anytime, or links have to be invalidated whenever the semantics of one of the elements they link changes.

1.8 Contextual Modules

As discussed in section 2, irrespectively of issues related to the size and complexity of knowledge repositories, modularization may be used as a means to achieve contextualization and personalization of knowledge services. Modules in this perspective are not conceptualizations of different sub-domains but different conceptualizations of the same domain, each one tailored for a specific usage context. Each conceptualization stems from a specific perspective on and interpretation of which knowledge is relevant and how it has to be represented according to many parameters, including the observer's background and preferences, the purpose assigned to its representation, the temporal and spatial framework of the observation. For example, should the spatial context be North America and the temporal context be before the first arrival of Europeans, the concept of *horse* would not be part of the relevant knowledge as the concept was not known to American Indians at that time. The concept become relevant with the arrival of Europeans, and denotes a kind of quadruped. For a more recent temporal context, the term would also denote heroin (in slang). Later, the same term would also denote persons smuggling drugs. Notice that recording synonyms and other lexical relationship among words, as done in terminological ontologies (e.g. Wordnet), does not suffice to convey the complexity of contextual influence on knowledge description. Context-dependent modules for the same ontological domain may provide the means to achieve context-aware information services. These modules may be independent from each other, but most likely they will have interdependencies to the extent they share some common knowledge. In particular, one module may include another module [5]. For example, all knowledge about a university system according to a "Professor" context may also belong by definition to a more generic "Academic" context for representing the same university system. We would then say that the Academic context includes the Professor context.

This approach to modularization suggests that the contexts of interest have been defined first, so that when entering new knowledge into the repository the context(s) to which it belongs can be readily identified. Existing context-awareness proposals are built on this hypothesis [5][19][11]. They are discussed in Chapter 5 of this book. A posteriori allocation to contexts is possible but is likely to call for manual annotation, usually a non-affordable task.

1.9 Conclusion

Managing large ontologies is a serious challenge for ontology designers, reasoners and users. One generic strategy to deal with the problem is modularization, which aims at replacing a huge ontology by a collection of smaller component ontologies, called modules, that can be manipulated independently from each other and are nevertheless capable of collaborating in providing the same service as the whole initial ontology. Although knowledge modularization is mainly a relatively new research domain, it has attracted the attention of several research groups and workshops trying to deal with large and distributed ontologies. Many research proposals have therefore been documented in the literature. Tendencies have appeared and enable an organized survey of current achievements, which is the subject of this book. Thus, most significant achievements related to the topic are described in detail within the book.

As a preamble, this initial chapter developed a global analysis of modularization issues, caring to take into account the various facets and perceptions of ontology modularization and suggesting possible answers to the many open problems. In particular, we identified a composition versus a decomposition approach to modularization. In the former, a set of existing source ontologies are apprehended as modules of a larger ontology that is built from the preexisting sources using some integration or cooperation technique. In the latter, it is the global ontology that pre-exists, and modularization is seen as the process of producing a consistent set of sub-ontologies, the modules, using some decomposition technique. Beyond this major split within the set of approaches that deal with modularization, we identified a number of issues, from the precise definition of the module concept to how different modules can be interconnected to benefit from complementarities of their semantic content. This generic analysis is further explored in following chapters 2 to 4 in this Part I of the book, focusing on formal properties of modularization and evaluation criteria for modularization techniques, as well as the variety of knowledge importing techniques that can be used to transfer knowledge between modules.

We have focused on identifying and showing alternative approaches, with their underlying assumptions as well as with their specific goals. The need for modularization does indeed emerge from different contexts, characterized by different requirements. A multiplicity of solutions is required to cover all potential useful contexts.

However, neither this chapter nor the book exhaust the possible research avenues that remain open for the future. For example, one direction for future work is to focus on using fuzzy techniques to support the possibility to attach different confidence

degrees to the mappings between ontological modules, thus leading to some form of fuzzy modular ontology.

References

1. Aberer, K., et al.: Emergent semantics principles and issues. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 25–38. Springer, Heidelberg (2004)
2. Bouquet, P., Giunchiglia, F., van Harmelen, A., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. Journal of Web Semantics 1(4), 325–343 (2004)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-lite: Tractable description logics for ontologies. In: Proceedings of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005) (2005)
4. Ceri, S., Pelagatti, G.: Distributed Databases - Principles and Systems. McGraw-Hill, New York (1984)
5. Cudre-Mauroux, P., Aberer, K. (eds.): Viewpoints on emergent semantics. Journal on Data Semantics VI, 1–27 (2006)
6. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
7. Grau, B.C., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006) (2006)
8. Jarrar, M.: Modularization and automatic composition of object role modeling (orm) schemes. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 613–625. Springer, Heidelberg (2005)
9. Magkanaraki, A., Tannen, V., Christophides, V., Plexousakis, D.: Viewing the semantic web through RVL lenses. Journal of Web Semantics 1(4), 359–375 (2004)
10. Parent, C., Spaccapietra, S.: Database integration: an overview of issues and approaches. Communications of the ACM 41(5), 166–178 (1998)
11. Parent, C., Spaccapietra, S., Zimányi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach. Springer, Heidelberg (2006)
12. Rector, A.: Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In: Proceedings of the K-CAP 2003 Conference, pp. 121–128 (2003)
13. Sheth, A., Larson, J.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys 22(3), 183–236 (1990)
14. Sheth, A., Kashyap, V.: So far (schematically) yet so near (semantically). IFIP Trans. A-25 41(5), 166–178 (1998)
15. Stuckenschmidt, H., Klein, M.: Modularization of ontologies, WonderWeb: Ontology infrastructure for the semantic web. Deliverable 21 v.0.6 (May 14, 2003)
16. Sahoo, S.S., Thomas, C., Sheth, A., York, W.S., Tartir, S.: Knowledge modeling and its application in life sciences: A tale of two ontologies. In: Proceedings of the K-CAP 2003 Conference (2006)
17. Thalheim, B.: Engineering database component ware. In: Draheim, D., Weber, G. (eds.) TEAA 2006. LNCS, vol. 4473, pp. 1–15. Springer, Heidelberg (2007)
18. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. Nature Genetics 25, 25–29 (2000)

19. Taylor, M.E., Matuszek, C., Klimt, B., Witbrock, M.: Autonomous classification of knowledge into an ontology. In: Proceedings of the Twentieth International FLAIRS Conference (2007)
20. Teorey, T.J., Wei, G., Bolton, D.L., Koenig, J.A.: ER model clustering as an aid for user communication and documentation in database design. Communications of the ACM 32(8), 975–987 (1989)
21. Volz, R., Oberle, D., Studer, R.: Views for light-weight web ontologies. In: Proceedings of the ACM Symposium on Applied Computing (SAC), 2003, pp. 1168–1173. ACM, New York (2003)