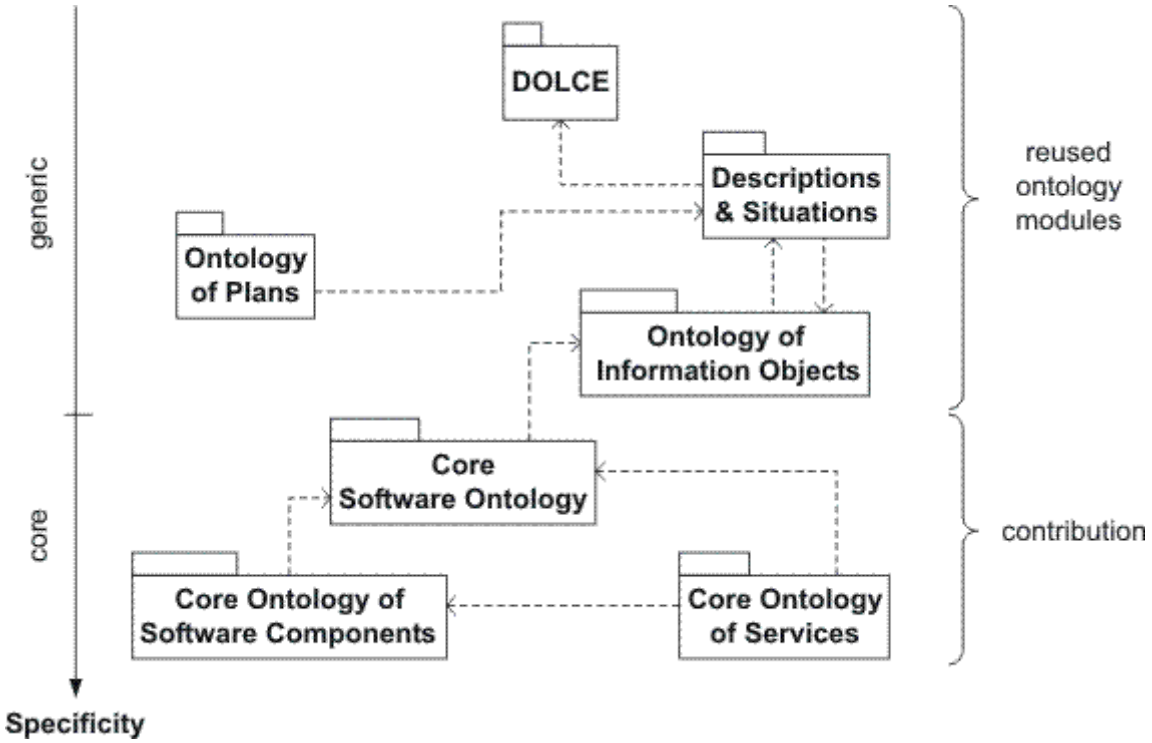
Core Software Ontology
Core Ontology of Software Components
Core Ontology of Services

- **Overview**



The figure provides an overview of the reused ontology modules and the modules we contribute. Besides *DOLCE*, we also need theories for contextualization, for plans and for information objects. *Descriptions & Situations*, the *Ontology of Plans* and the *Ontology of Information Objects* realize such theories and come in the form of ontology modules. Our contributed ontologies are the Core Software Ontology and the Core Ontologies of Software Components and Services.

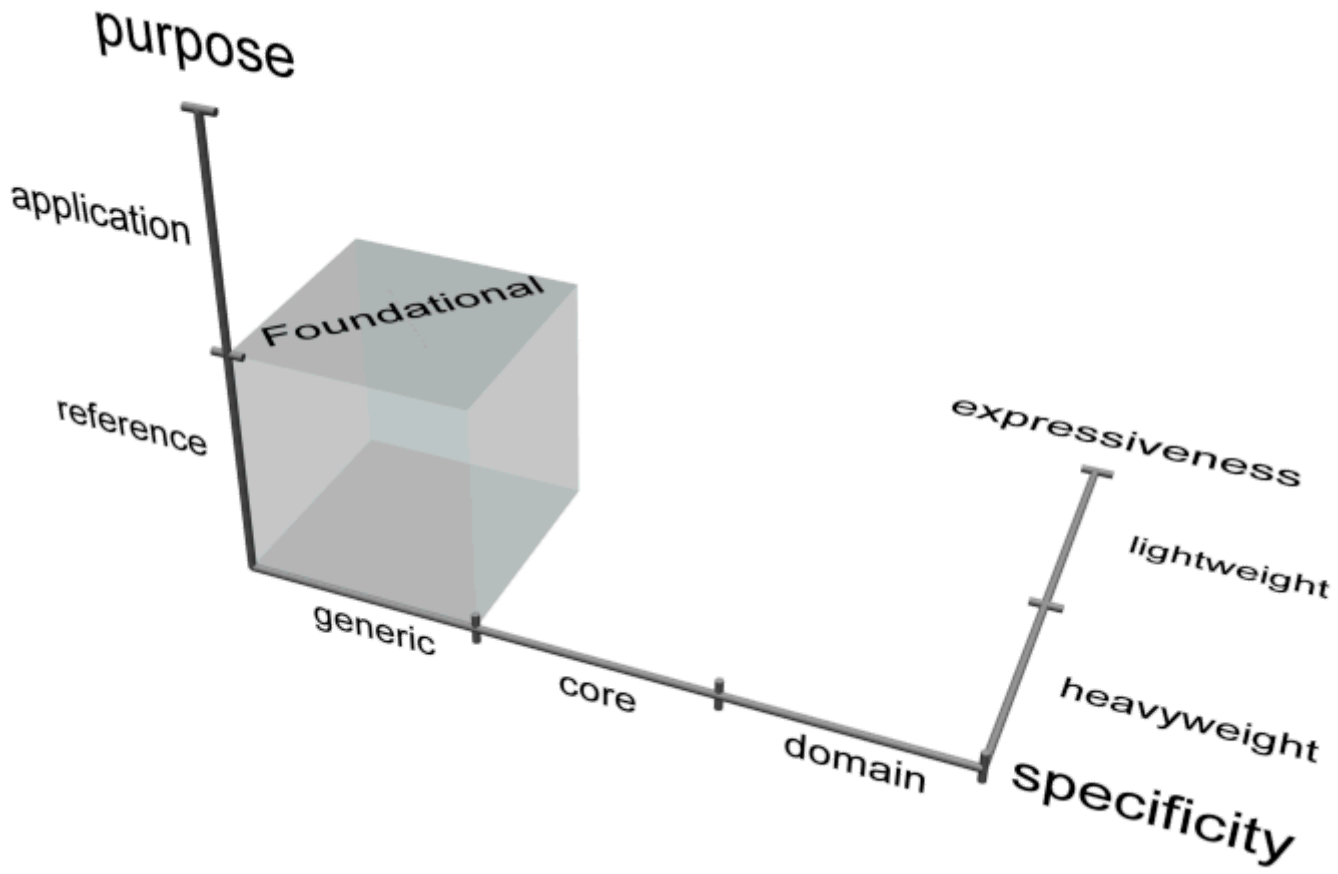
The *Core Ontology of Software* formalizes the most fundamental concepts which are required to model both software components and Web services. This includes concepts such as software, data, users, access rights or interfaces. Such concepts are formalized by reusing the modelling

basis of DOLCE, Descriptions & Situations, Ontology of Plans and the Ontology of Information Objects. We separated the fundamental concepts in the Core Software Ontology to facilitate reuse.

The *Core Ontology of Software Components* is based on the Core Software Ontology to formalize our understanding of the term software component. It requires special attention as there is a variety of interpretations that leads to ambiguity. We also put libraries and licenses in this core ontology and define a component profile that aggregates all relevant aspects of a component. It is expected that this grouping makes browsing and querying for developers more convenient. The component profile is envisioned to act as the central information source for software components rather than having bits and pieces all over the place. The *Core Ontology of Web Services* is based on the Core Software Ontology. It formalizes our understanding of the term Web service and introduces the notion of service profiles.

The *KAON SERVER ontology* (not shown in the figure) reuses the Core Ontology of Software Components to specialize and extend its concepts and associations such that they reflect the idiosyncracies of the KAON SERVER. The KAON SERVER is an ontology-based application server and can be obtained at http://kaon.semanticweb.org/server. We introduce MBeans as a special kind software components and by defining additional kinds of profiles. Note that component providers can further extend the ontology by introducing customized profiles to categorize specific components, e.g., ontology stores.

- **Classification**



The classification allows clarifying how the ontologies were built. We started by analysing whether the quality of existing ontologies suffice for our goals of having *core*, *reference* and *heavyweight* ontologies. As this wasn't the case, we decided to build the ontologies anew by choosing a suitable foundational ontology according to ontological choices (descriptive vs. revisionary, multiplicative vs. reductionist, actualism vs. possibilism, endurantism vs. perdurantism as well as several extrinsic properties). Eventually, we took DOLCE and some its modules (see above) as foundation (*generic*, *reference* and *heavyweight*). DOLCE is used as a starting point to arrive at our contributed ontologies, viz., the Core Software Ontology and the Core Ontologies of Software Components and Services, which are classified as.

In order to reuse one of our core ontologies in a specific setting, the following three steps have to be taken: (*i*) specialize the core concepts and associations to reflect the idiosyncracies of the platform. For example, we have to introduce EnterpriseBean as a special kind of COSC:SoftwareComponent in a J2EE-based platform. The result of this step is a *domain*, *reference* and *heavyweight* version of our core ontologies. Step (*ii*) removes concepts and associations that have been introduced merely for reference purposes. As an example, it is unlikely and not required to model particular Computational Objects or Computational Activities for the reasoning at run time. Abstract concepts, i.e., ones which are not instantiated, are of no use in a running system and can be removed. The result is a *domain*, *application* and *heavyweight* version. Finally, step (*iii*) requires a decision for an executable ontology language that can be reasoned with at run time. Accordingly, the

axiomatization has to be adapted to this language. This might be a description logic such as OWL-DL which is less expressive than modal logic S5. The result of this step is a *domain*, *application* and *lightweight* version of the core ontologies.

Consequently, we offer the axiomatization of the *core*, *reference* and *heavyweight* axiomatizations on paper in modal logic S5. The *domain*, *application* and *lightweight* versions are available in OWL-DL and KAON.

- Classification according to Purpose
  - Application Ontology
    Used during run time of a specific application implementing an ontology putting constraints on the axiomatization for the terminological service, i.e., the reasoner. The typical trade-off between expressiveness and decidability requires a limited representation formalism. In a description logics based application this would coincide with the TBox. Application ontologies may also describe specific worlds (semantic descriptions, knowledge base, metadata, semantic metadata or simply instances) In a description logics based application this would coincide with the ABox.
  - Reference Ontology
    Used during development time of applications for mutual understanding and explanation between (human or artificial) agents belonging to different communities, for establishing consensus in a community that needs to adopt a new term or simply for explaining the meaning of a term to somebody new to the community. Although parts of the reference ontology can be formalized in a TBox as well, description logics are usually not expressive enough for reference purposes.
- Classification according to Expressiveness
  - Heavyweight Ontology
    Heavyweight ontologies are extensively axiomatized and thus represent ontological commitment explicitly. The purpose of the axiomatization is to exclude terminological and conceptual ambiguities, due to unintended interpretations. Every heavyweight ontology can have a lightweight version. Many domain ontologies are heavyweight because they should support heavy reasoning (e.g., for integrating database schemata, or to drive complex corporate applications). As with all dimensions, the borderline between light and heavy weight is not clearly delimited.
  - Lightweight Ontology
    Lightweight ontologies are simple taxonomic structures of primitive or composite terms together with associated definitions. They are hardly axiomatized as the intended meaning of the terms used by the community is more or less known in advance by all members, and the ontology can be limited to those structural relationships among terms that are considered as relevant.
- Classification according to Specificity
  - Generic Ontology
    The concepts defined by this layer are considered to be generic across many fields. Typically, generic ontologies (synonyms are "upper level" or "top-level" ontology) define concepts such as state, event, process, action, component etc.
  - Core Ontology
    Core ontologies define concepts which are generic across a set of domains. Therefore, they are situated in between the two extremes of generic and domain ontologies. The borderline between generic and core ontologies is not clearly defined because there is no exhaustive enumeration of fields and their conceptualizations. However, the distinction is intuitively meaningful and useful for building libraries.
  - Domain Ontology
    Domain ontologies express conceptualizations that are specific for a specific universe of discourse. The concepts in domain ontologies are often defined as specializations of concepts in the generic and core ontologies. The borderline between core and

domain ontologies is not clearly defined because core ontologies intend to be generic within a domain. Thus, it is usually hard to make a clear cut between generic and core as well as between core and domain ontologies. A concept such as software component would be placed in a core ontology for application servers for reuse in every possible domain ontology we can think of. However, a concept such as enterprise bean might only be relevant in a specific J2EE setting.

- ## Download

Module	Namespace	.owl File	.kaon File
DOLCE	http://www.loa-cnr.it/ontologies/DOLCE-Lite#	DOLCE-Lite_397.owl	dolce.kaon
Descriptions & Situations	http://www.loa-cnr.it/ontologies/ExtendedDnS#	ExtDnS_397.owl	dolce.kaon
Ontology of Information Objects	http://www.loa-cnr.it/ontologies/ExtendedDnS#	Information_397.owl	dolce.kaon
Ontology of Plans	http://www.loa-cnr.it/ontologies/Plans#	Plans_397.owl	oop.kaon
Core Software Ontology	http://cos.ontoware.org/cso#	cso.owl	cso.kaon
Core Ontology of Software Components	http://cos.ontoware.org/cosc#	cosc.owl	cosc.kaon
Core Ontology of Services	http://cos.ontoware.org/cos#	cows.owl	cows.kaon
KAON SERVER Ontology	http://kaon.semanticweb.org/server#	server.owl	server.kaon

You may download the files directly from the Ontoware repository. The package diagram at the very top of this page shows the inclusion graph of ontology modules. In KAON, modularization is realized by a relative `kaon:includes`, i.e., place all the files in the same directory and open `server.kaon` to obtain the whole ontology. In OWL-DL, modularization is realized by `owl:imports`, the links to the imported ontology are http-references (url's see table). That means you have to be online for a successful resolution of the `owl:imports` statement modules. You have to open `server.owl` to obtain the whole stack of ontology modules in your editor. The OWL-DL version of DOLCE and its modules is available at LOA website. The OWL files were edited with Protege 3.1.

- ## Documentation

  - Read the whole story in Semantic Management of Middleware by Daniel Oberle appeared in the book series "The Semantic Web and Beyond" (published by Springer and edited by Amit Sheth). The book includes a detailed analysis of the shortcomings of existing ontologies, a careful selection of the appropriate foundational ontology according to specific ontological choices, an introduction of the reused ontology modules, extensive axiomatization of the contributed modules and a detailed discussion of how to obtain the domain, application and lightweight version and a lot more.

- Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems, submission to the Journal of Applied Ontology highlighting the ontology's axiomatization, download
- Foundations for Service Ontologies - Aligning OWL-S to DOLCE, In Proceedings of WWW 2004, download
- An Ontology of Services and Service Descriptions, Technical Report, University of Karlsruhe, download
- KAON SERVER

- **Contributors (in alphabetical order)**

  - Aldo Gangemi, LOA Rome
  - Stephan Grimm, FZI Karlsruhe
  - Peter Mika, VUA Amsterdam
  - Daniel Oberle, Univ. of Karlsruhe (maintainer)
  - Steffen Lamparter, Univ. of Karlsruhe
  - Marta Sabou, VUA Amsterdam
  - Steffen Staab, Univ. of. Koblenz
  - Denny Vrandecic, Univ. of Karlsruhe