Chapter 4
Developing Ontologies within Decentralised Settings

Alexander Garcia, Kieran O'Neill, Leyla Jael Garcia, Phillip Lord, Robert Stevens, Oscar Corcho, and Frank Gibson

Abstract This chapter addresses two research questions: "How should a well-engineered methodology facilitate the development of ontologies within communities of practice?" and "What methodology should be used?" If ontologies are to be developed by communities then the ontology development life cycle should be better understood within this context. This chapter presents the Melting Point (MP), a proposed new methodology for developing ontologies within decentralised settings. It describes how MP was developed by taking best practices from other methodologies, provides details on recommended steps and recommended processes, and compares MP with alternatives. The methodology presented here is the product of direct first-hand experience and observation of biological communities of practice in which some of the authors have been involved. The Melting Point is a methodology engineered for decentralised communities of practice for which the designers of technology and the users may be the same group. As such, MP provides a potential foundation for the establishment of standard practices for ontology engineering.

4.1 Introduction

The maturity of a particular scientific discipline can be defined by its progress through three main stages. First, *innovation* followed by the subsequent dissemination of the resulting knowledge or artefact. Second, the formation of *communities* or collaborations, that utilise or build upon the innovations. Third, the proposal and agreement upon *standards* for protocols to achieve the unified and consistent progression of innovation and knowledge [1]. The discipline of ontology engineering can be thought of as progressing through the second stage of scientific maturity, moving from ontologies developed by a single authoritative expert to harvesting the collective intelligence of an application domain [2–4]. This trend is also reflected in

A. Garcia (✉)
University of Bremen, Bremen, Germany
e-mail: alexgarciac@gmail.com

H. Chen et al. (eds.), *Semantic e-Science*, Annals of Information Systems 11,
DOI 10.1007/978-1-4419-5908-9_4,

the availability of software supporting the engagement of several domain experts, communities, representing knowledge and developing ontologies [2, 5]. Therefore, ontology engineering is on the cusp of the third stage of scientific maturity, requiring the development of common working practices or standard methodologies.

Knowledge engineering (KE) is a field that involves integrating knowledge within computer systems [6] or the building, maintaining and development of knowledge-based systems [7]. Therefore, some of the methods proposed within the field of KE are applicable when building ontologies [8]. However, the experiences from KE have not always been applied when developing ontologies. In general KE methodologies focus primarily on the use of the ontology by a software system as opposed to the development of the ontology [9].

Within the domain of ontology engineering several methodologies have been proposed and applied [10–17]. The majority of the proposed methodologies have been engineered for centralised settings. However, none of these have gained widespread acceptance, predominant use or have been proven to be applicable for multiple application domains or development environments [18]. To date the community has not been widely involved or considered within ontology engineering methodologies. This situation has encouraged debate amongst those within the ontology community as to which methodology or combination of methodologies are most applicable [18, 9].

The language choice for encoding an ontology is still an open debate across the ontology building communities. This situation can be illustrated by the use of both the OBO format and the OWL within the bio-ontology community [19]. Conforming to or accepting a single formalism for ontology encoding would bring consistency and standardisation to the engineering methodology, such as tool support and reasoning engines. However, it is outside the scope of this work to recommend a particular formalism for ontology encoding. Therefore, the ontology methodologies are considered and analysed in a language-independent manner.

Whatever methodology emerges, it is essential that the methodology should be able to support the construction of ontologies by communities and utilise the collective intelligence of the application domain. Of the published methodologies that have been proposed or applied, no methodology completely satisfies all the criteria for collaborative development. To this end, we have reviewed the existing methodologies, identified commonalities and assessed their suitability for use within community ontology development. We have summarised these commonalities into a convergence of existing methodologies, with the addition of new aspects which we have termed the Melting Point (MP) methodology. The MP methodology builds upon the authors' experiences with community-developed ontologies, re-using methods and techniques that already exist and suggesting new mechanisms to cope with collaborative ontology development in decentralised settings.

4.1.1 Decentralised Communities

As knowledge is in a constant flux, ontologies should be flexible so they are reusable and easily extensible. This is not effortlessly achievable as representing

knowledge requires the active participation of domain experts. The majority of existing methodologies have been engineered for centralised settings, in which the ontology is developed and deployed on a one-off basis. Afterwards, the maintenance, as well as the evolution of the ontology, is left to the knowledge engineer and a reduced group of domain experts. This situation is also true throughout the development process: a reduced group of domain experts work together with the knowledge engineer to build the ontology. To date the community has not been widely involved or considered within ontology engineering methodologies.

The costs and efforts associated with the development of ontologies are considerable, it is therefore important to facilitate this process by allowing the community to participate in the development. By having this active participation some important aspects are covered: first, the quality of the model is constantly verified and second the evolution of the ontology is feasible. This paradigm follows the "wisdom of crowds" — assuming that more contributors implies higher quality or volume of information — as is employed within wiki-based collaborations. Collaboration is thus at the Melting Point in a methodology for developing ontologies.

4.1.2 Community-Driven Ontology Engineering

To illustrate the motivation and applicability of the MP methodology, examples are given from the life sciences, specifically the biomedical ontology domain. Within the knowledge-intensive biological domain, collaboration and community involvement is common place and encouraged in ontology development maintenance, evaluation and evolution.

Despite the lack of formal methodologies, bio-ontologies continue to be developed and the nature of this development has two very interesting properties. First, it is highly distributed; domain experts in any given sub-domain of the biological sciences are rarely in one place. Rather, they are distributed across the globe yet frequently interact to either collaborate or peer review each others' work. Hence, when biologists build ontologies, they tend to form virtual organisations in which experts with different but complementary skills collaborate in building an ontology for a specific purpose. The structure of this collaboration does not necessarily have a central control; different domain experts join and leave the network at any time and decide on the scope of their contribution to the joint effort. Leveraging this kind of virtual collaboration can be a powerful tool when constructing an ontology. Second, biological ontologies continue to evolve, even after the initial development drive. The continued evolution reflects the advancement of scientific knowledge discovery. New classes, properties and instances may be added at any time and new uses or extended scope for the ontology may be identified [17]. By engendering and facilitating this level of community participation, an ontology engineer can speed up the initial development and help to ensure that the ontology remains up to date as knowledge within the domain advances.

For example, the Ontology of Biomedical Investigations (OBI)[1] aims to provide an ontological representation of life science investigations covering common components of experimentation, such as equipment, materials and protocols. The developer community of OBI[2] is currently affiliated with 18 diverse biomedical communities, ranging from functional genomics to crop science to neuroscience. In addition to having a diverse community of expertise, the OBI developers work in a decentralised environment encompassing multiple countries and time zones.

The diversity of the life science domain results in a multitude of application domains for ontology development. To account for and identify available bio-ontologies the Open Biomedical Ontologies (OBO) Foundry [3] was formed. The OBO Foundry acts as a registry to collect public domain ontologies that, by design and revision, are developed by and available to the biomedical community, fostering information sharing and data interpretation. As of 23 October 2008 there are 76 registered ontologies at the OBO Foundry, representing knowledge domains ranging from Amphibian gross anatomy, infectious diseases to scientific experimentation. Although registered in the same library, the bio-ontologies, often present overlap in terminology or application domain. In addition to providing a registry, the OBO Foundry was formed to reduce ontology overlap and ensure bio-ontology orthogonality. Initial steps at achieving this aim have produced a set of design principles[3] to which domain ontologies should adhere, such as openness, a shared syntax and class definitions. However, the OBO Foundry does not suggest a community-orientated engineering methodology, methods or techniques by which these principles can be met.

Case studies have been described for the development of ontologies in diverse domains, yet surprisingly very few of these have been reported to have been applied to a domain allied to bioscience, the chemical ontology [13], and the ontology for immune epitopes [20] being noteworthy exceptions. The research focus for the bio-ontology community to date has typically centred on the development of domain-specific ontologies for particular applications, as opposed to the actual "how to" of building the ontology or the "materials and methods" [17, 21]. This has resulted in a proliferation of bio-ontologies, developed in different ways, often presenting overlap in terminology or application domain.

The biomedical domain is not the only domain where ontologies are being developed and applied. The Semantic Web (SW) encompasses a vision where knowledge and relationships between documents are disseminated via ontologies by annotating the current, largely human-accessible Web, to facilitate a Web amenable to machine processing [22]. Indeed, the creators of that vision consider the life sciences to potentially be the "incubator" community for the SW, as the physics community was for the Web [23]. Within the SW vision, as within the biomedical domain, the involvement of communities of practice is crucial, not only for the development, but also for the maintenance and evolution of ontologies.

[1] http://purl.obofoundry.org/obo/obi/

[2] http://obi-ontology.org/page/consortium

[3] http://www.obofoundry.org/crit.shtml

4.1.3 Upper Level Ontologies

As the biomedical domain is highly interconnected, domain ontologies may overlap with each other. For instance, OBI requires the availability of definitions for those chemicals used in any investigation. These definitions do not need to be developed within the OBI ontology as there is already a biomedical ontology for the domain of chemicals, called ChEBI [24]. Similarly, software making use of an ontology may require more than a single domain ontology. Typically, in these types of scenarios, it is necessary to integrate multiple ontologies into a single coherent narrative. In order to integrate or re-use specific domain ontologies following this "building-block" approach there has to be a high-level structure or common "scaffold" where different parts of different domain ontologies may be "plugged" into. To ensure ease of interoperation or re-use of a domain ontology, well designed and documented ontologies are essential and upper ontologies are fundamental in this integrative effort.

Upper level ontologies provide a domain-independent conceptual model that aims to be highly re-usable across specific domain applications. Most of the upper ontologies provide a general classification criterion that makes it easy to re-use, extend and maintain those existing ontologies required by a particular application. Therefore, it is essential, to aid interoperability and re-use, that ontology development methodologies should provide general guidelines for the use of upper level ontologies. These guidelines should cover the documentation of (i) the design decisions and the justification for choosing one upper ontology over another and (ii) examples that illustrate how they used in the conceptualisation of a particular domain. Examples of upper level ontologies include the Basic Formal Ontology (BFO) [25], DOLCE [26] and GFO [27]. This adoption has, however, not been documented within a methodological framework that facilitates both the adoption of the upper level ontology and its proper use. However, the OBO foundry has recommended that ontologies registered on the OBO Foundry should use BFO.

4.1.4 Dynamic Ontologies

Ontologies, like software, evolve over time; specifications often change as the development proceeds, making a straightforward path to the ontology unrealistic. Different software process models have been proposed; for instance, linear sequential models and prototyping models. Linear sequential models are also known as waterfall models [28, 29] and are designed for straight-line development. The linear sequential model suggests a systematic, sequential approach in which the complete system will be delivered once the linear sequence is completed [29]. The role of domain experts is passive as end-users of technology. They are placed in a reacting role in order to give feedback to designers about the product. The software or knowledge engineer leads the process and controls the interaction amongst domain experts. A high-speed adaptation of the linear sequential model is the

Rapid Application Development (RAD) model [30, 31]. This emphasises short development cycles for which it is possible to add new software components, as they are needed. RAD also strongly suggests reusing existing program components or creating reusable ones [29].

The prototyping model is more flexible as prototypes are constantly being built. Prototypes are built as a means for defining requirements [29], this allows for a more active role from domain experts. A quick design is often obtained in short periods of time. The model grows as prototypes are being released [32]; engineers and domain experts work on these quick designs. They focus on representational aspects of the ontology, while the main development of the ontology (building the models, defining what is important, documenting, etc.) is left to the knowledge engineer.

The evolutionary nature of the software is not considered in either of the aforementioned models, from the software engineering perspective evolutionary models are iterative, and allow engineers to develop increasingly more complex versions of the software [29, 31, 33]. Ontologies are, in this sense, not different from other software components for which process models have evolved from a "linear thinking" into evolutionary process models that recognise that uncertainty dominates most projects, that timelines are often impossibly short and that iteration provides the ability to deliver a partial but extendible solution, even when a complete product is not possible within the time allotted. Evolutionary models emphasise the need for incremental work products, risk analysis, planning followed by plan revision, and customer (domain expert) feedback [29].

4.1.5 The Melting Point: A Methodology for Distributed Community-Driven Ontology Engineering

A general purpose methodology should aim to provide ontology engineers with a sufficient perspective of the stages of the development process and the components of the ontology life cycle, and account for community development. In addition, detailed examples of use should be included for those stages, outcomes, deliverables, methods and techniques; all of which form part of the ontology life cycle [9, 34].

To address ontology development methodology in a distributed community environment the "The Melting Point" methodology is described. Consideration has been applied to the previously proposed methodologies and their integration adaptation and the re-use of components were possible within the MP. Several IEEE software engineering practices have also been included to formulate a methodology applicable to the requirements of community ontology development. The Melting Point also follows Sure's [9] work as it considers throughout the whole process the importance of the software applications that will ultimately use the ontology. The following sections not only present the MP methodology but also the relationship between methodological issues and the life cycle of community-based ontologies.

An analysis of current ontology engineering methodologies is presented in Section 4.2, emphasising the significance of commonalities across methodologies as

well as the engagement of communities of practice. Section 4.3 presents a detailed definition of the methodology and related components; methods, techniques, activities, and tasks of the MP methodology. Sections 4.4 and 4.5 contain discussion and conclusions, respectively.

4.2 Review of Current Methodologies

Melting Point espouses the combination of good practices from existing methodologies. A comparison of these methodologies is therefore appropriate, in order to give context to MP. Several ontology methodology approaches are analysed below, according to criteria described in detail in Section 4.2.1. These criteria are derived from the work done by Fernandez [35], Mirzaee [18] and Corcho et al. [36].

The engineering methodologies analysed are the Enterprise Methodology proposed by Uschold and King [10]; the TOVE Methodology proposed by Gruninger and Fox [11]; the Bernaras methodology proposed by Bernaras et al. [12]; the METHONTOLOGY methodology proposed by Fernandez et al. [37] the SENSUS methodology proposed by Swartout et al. [14]; the DILIGENT methodology proposed by Pinto et al. [15, 16]; the GM methodology proposed by Garcia et al. [17] the iCAPTURer Methodology proposed by Good et al. [38] and the NeOn methodology[4]. Table 4.1 provides a summary of the methodologies and the results of the analysis against the criteria. Complete details of the analysis have been provided in Appendix for reference.

4.2.1 Criteria for Review

C1. *Inheritance from knowledge engineering*. Ontology building is ultimately the assertion and representation of knowledge. Therefore, this criterion considers the influence traditional Knowledge Engineering (KE) has had on the methodologies studied.

C2. *Detail of the methodology*. This criterion is used to assess the clarity with which the methodology specifies the orchestration of methods and techniques.

C3. *Strategy for building the ontology*. This should provide information about the purpose of the ontology, as well as the availability of domain experts. There are three main strategic lines to consider: (i) the application of the ontology; (ii) the use and type of domain experts available and (iii) the type of ontology to be developed. These three aspects are defined in more detail from C3.1 to C3.3.

C3.1. *Application of the ontology*. This criterion describes how tightly coupled the ontology is going to be in relation to the application within

[4] http://www.neon-project.org

Table 4.1 Evaluation of ontology engineering methodologies according to common criteria (specified in Section 4.1). Full details are provided in Appendix A

	Inheritance from knowledge engineering	Detail of the metho-dology	Strategy for building the ontology			Strategy for identifying concepts	Recommended life cycle	Recommended methods, techniques and technology	Applicability	Community involvement	Knowledge elicitation
			Application of the ontology	Domain experts	Ontology type						
Enterprise	Partial	Very little	Application independent	N/A	N/A	Middle out	N/A	N/A	Enterprise ontology	N/A	N/A
TOVE	Small	Little	Application semi-independent	N/A	Domain and task ontologies	Middle out	N/A	N/A	Business and foundational ontologies	N/A	N/A
Bernaras	A lot	Very little	Application independent	N/A	Domain and task ontologies	Top down	Follows software	N/A	Electrical engineering domain	N/A	N/A
METH-ONTOLOGY	A lot	A lot	Application independent	N/A	In principle it is applicable to any kind of ontologies	Middle out	Evolutionary prototype	Some activities missing. Technology recommended	Multiple developments reported	N/A	Partially
SENSUS	Inexistent	Medium	Application semi-independent	N/A	In principle it is applicable to any kind of ontologies	N/A	N/A	N/A	Air campaign planning ontology	N/A	N/A
DILIGENT	Small	Small	Application dependent	N/A	N/A	Problem dependent	N/A	N/A	N/A	Yes	Partially
GM	Small	A lot	Application dependent	The focus is more on specialized domain experts	Domain and task ontologies	Top down	Evolutionary prototype	Some techniques and methods recommended Technology recommended	Developments reported for the bio domain	Yes	Supported
iCAPTURer	Little	A lot	Application-independent	Broad: All levels of domain experts considered	Domain and task ontologies	Bottom up	Evolutionary/ iterative	N/A	N/A	Yes	Integral
NeOn	Strongly influenced	A lot	Application-independent	Domain experts and ontology engineers assumed to be actively working together.	Domain and task ontologies	N/A	Recognised but not prescribed	Many, with specific detail	Multiple developments reported	Collaboration of the community is assumed	Significant

which, in principle, it should be used. This is often evaluated via competency questions. Competency questions can be typically classified in two forms: informal competency questions (natural language) and formal competency questions. The criteria for describing the level of application engagement are described as follows:

C3.1.1. *Application dependent*. The ontology is built on the basis of an application knowledge base, by means of a process of abstraction [35].
C3.1.2. *Application semi dependent*. Possible scenarios of ontology use are identified in the specification stage [35].
C3.1.3. *Application independent*. The process is totally independent of the uses to which the ontology will be put in knowledge-based systems or any software layer making use of the ontology.

C3.2. *Domain experts*. This criterion outlines the level of perceived expertise of the individuals consulted within the ontology development process. A domain expert can be graded in the following manner:

C3.2.1. *Specialised domain experts*. Those with an in-depth knowledge of their field. Within the biological context these are usually researches with vast laboratory experience, very focused and narrowed within the domain of knowledge. Having the specialised domain expert helps define very specific concepts within the ontology; this can lead to a strategy for identifying concepts known as the bottom-up approach (see C4.1).
C3.2.2. *Broader-knowledge domain experts*. Those who have a general knowledge or a higher level view of the domain(s). Having this kind of domain experts usually facilitates capturing concepts more related to high-level abstraction, and general processes, rather than specific vocabulary describing those processes. The ontology may be built from high-level abstractions downwards to specifics. This approach known as the top–down strategy for identifying concepts (see C4.2).

C3.3. *Ontology type*. The ontology-type criterion classifies the ontology being developed into the following types or categories [39, 40]:

C3.3.1. *Top-level ontologies*. These describe very general concepts like space, time, event, which are independent of a particular problem domain. Such unified top-level ontologies aim at serving large communities [9]. These ontologies are also known as foundational ontologies or commonly called upper ontologies; see, for instance, the DOLCE ontology [26].

C3.3.2. *Domain ontologies*. These are focused within a particular domain and describe specific vocabulary.
C3.3.3. *Task ontologies*. These describe vocabulary related to tasks, processes or activities.
C3.3.4. *Application ontologies*. As Sure [9] describes them, application ontologies are specialisations of domain and task ontologies as they form a base for implementing applications with a concrete domain and scope.

C4. *Strategy for identifying concepts*. There are three strategies regarding the construction of the ontology and the kinds of terms it is possible to capture [41]:

C4.1. *Bottom-up* work from the most concrete or specific concepts to the most abstract concepts. [41–43].
C4.2. *Top-down* work from the most abstract to the more domain/application specific concepts [35].
C4.3. *Middle-out* work from the most relevant to the most abstract and most domain/application-specific concepts [35, 40, 43].

C5. *Recommended life cycle*. This criterion evaluates whether and to what degree the methodology implicitly or explicitly proposes a life cycle [35].
C6. *Recommended methods and techniques*. This criterion evaluates whether or not there are explicit methods and techniques as part of the methodology. This is closely related to C2. An important issue to be considered is the availability of software supporting either the entire methodology or a particular method of the methodology. This criterion also deals with the methods or software tools available within the methodology for representing the ontology, for example, in OWL[5], or RDF[6].
C7. *Applicability*. As knowledge engineering is still in its infancy it is important to evaluate the methodology in the context of those ontologies for which it has been applied.
C8. *Community involvement*. As has been pointed out before in this chapter, it is important to know the level of involvement of the community. Phrasing this as a question, is the community a consumer of the ontology or is the community taking an active role in its development?
C9. *Knowledge elicitation*. Knowledge elicitation is a major bottleneck when representing knowledge [44]. It is therefore important to know if the methodology assumes knowledge elicitation to be an integral part of the methodology; does it describe the elicitation techniques?

[5] http://www.w3.org/TR/owl-ref/
[6] http://www.w3.org/RDF/

4.2.2 Finding the Melting Point

The considerable number of methodologies coupled with the limited descriptive detail of the ontology development approach makes it extremely difficult to present a consensus or a melting point where the methodologies converge. From the methodologies studied, very few clearly state the methods and techniques suggested in the methodology. However, the roles of those participating in the development of the ontology are clearly outlined. The following sections discuss the identified commonality and differences in approach, elucidated from the evaluation of the methodologies.

Common features. There are certain commonalities in the ontology development approach across the methodologies. For instance, all the studied methodologies consider an inception, formalisation as well as an evaluation phase. Figure 4.1 illustrates those shared stages across all methodologies. The DILIGENT and GM methodologies, however, present some fundamental differences when compared to the other methodologies — both were engineered for developing ontologies within geographically distributed settings. The differences identified between the DILIGENT and the GM methodology on the one hand and the other methodologies are presented and described below:

Life cycle. For both DILIGENT and GM, the ontology is constantly evolving, in a never-ending cycle. The life cycle of the ontology is understood as an open cycle in which the ontology evolves in a dynamic manner.

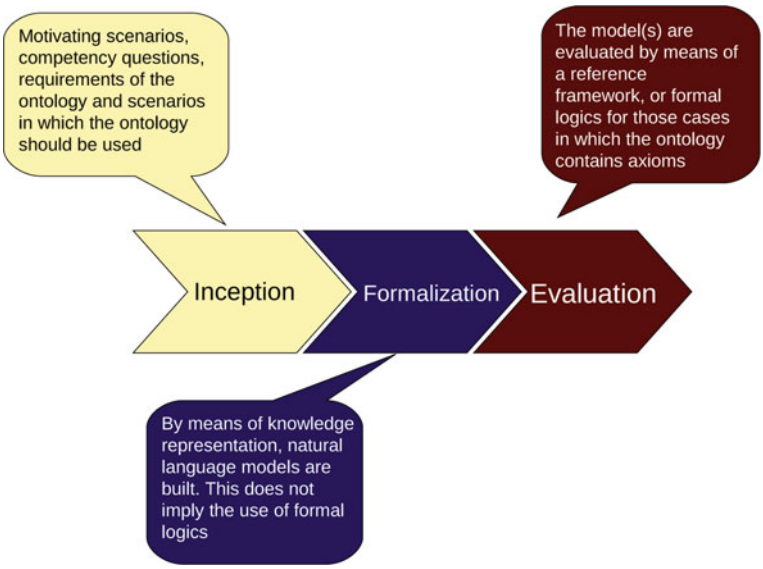

Fig. 4.1 Common features of the methodologies reviewed

Collaboration. For both DILIGENT and GM, a group of people agrees on the formal specification of the concepts, relations, attributes, and axioms that the ontology should provide. This approach empowers domain experts in a way that sets DILIGENT apart from the other methodologies.

Knowledge elicitation. Due in part to the involvement of the community and in part to the importance of the agreements, for DILIGENT and the GM methodology knowledge elicitation is assigned a high level of importance; it supports the process by which consensus is reached.

The GM, DILIGENT and NeOn methodologies consider the ontology to be constantly evolving. In fact, the life cycle spirals, with the ontology progressing over each iteration of the cycle. In addition, the GM methodology also emphasises the notion of collaboration in the development process, particularly during knowledge elicitation. The GM knowledge elicitation relies heavily on interaction; the higher the level of interaction amongst domain experts, the more refined the specific models are likely to be. Both DILIGENT and GM methodologies assume a leading role for the domain experts as well as a tight relationship between the ontology and the software application in which it will ultimately be used. Within the NeOn framework the focus is more on the network of ontologies rather than specifically on the act of collaboration amongst domain experts. However, NeOn offers a complete review of methods that in principle support collaboration when building ontologies; NeOn supports the collaboration over two main axes: argumentation and collaborative editing.

The GM, DILIGENT and NeOn methodologies consider the ontology to be constantly evolving. In fact, the life cycle spirals, with the ontology progressing over each iteration of the cycle. In addition, the GM methodology also emphasises the notion of collaboration in the development process, particularly during knowledge elicitation. The GM knowledge elicitation relies heavily on interaction; the higher the level of interaction amongst domain experts, the more refined the specific models are likely to be. Both DILIGENT and GM methodologies assume a leading role for the domain experts as well as a tight relationship between the ontology and the software application in which it will ultimately be used.

Differences amongst Methodologies. As illustrated by the summary table, no methodology completely satisfies all the criteria. Some of the methodologies, such as that of Bernaras, provide information about the importance of the relationship between the final application using the ontology and the process by which the ontology is engineered. This consideration is not always taken from the beginning of the development; clearly the kind of ontology that is being developed heavily influences this relationship. For instance, foundational ontologies rarely consider the software using the ontology as an important issue; these ontologies focus more on fundamental issues affecting the classification system such as time, space, and events. They tend to study the intrinsic nature of entities independently from the particular domain in which the ontology is going to be used [37].

The final application in which the ontology will be used also influences which domain experts should be considered for the development of the ontologies. For instance, specialised domain experts are necessary when developing application ontologies, domain ontologies or task ontologies, but they tend not to have such a predominant role when building foundational ontologies. For these kinds of ontologies philosophers and broader knowledge experts are usually more appropriate.

None of the methodologies investigated provided detail; the descriptions for the processes were scarce, and where present theoretical. There was no analysis of actual ontology building sessions. The methods employed during the development of the ontologies were not fully described. For instance the reasons for choosing a particular method over a similar one were not presented. Similarly there was no indication as to what software should be used to develop the ontologies. METHONTOLOGY was a particular case for which there is a software environment associated to the methodology; the recommended software WebODE [45] was developed by the same group to be used within the framework proposed by their methodology.

Although the methodologies investigated have different views on the life cycle of the ontology, only DILIGENT, NeOn and GM consider the life cycle to be dynamic. This is reflected in the processes these methodologies propose. The development happens in a continuum; some parts within the methodologies are iterative processes, but the steps are linear, taking place one after the other. In the case of DILIGENT the different view on the life cycle is clear. NeOn poses a view of the process that is closer to the one proposed by the MP methodology; it provides a clear view of the overall process and provides some detail as to the actual ontology building practice.

The lack of support for the continued involvement of domain experts who may be located around the world was not considered when engineering most of the studied methodologies. As both, the SW and the biodomain, pose a scenario for which information is highly decentralised, domain experts are geographically distributed and the interaction takes place mostly on a virtual basis, such consideration is important. For both cases the realisation of the SW vision, as well as the achievement of collaboration, is more about a change in people and communities of practices than it is about technology [4, 46].

Evolution and community involvement. Ontologies in the biomedical domain not only are domain and/or task specific but also application oriented. Within both, the SW and the biodomain, the construction of applications and ontologies will not always take place as part of the same software development projects. It is therefore important for these ontologies to be easily extensible; their life cycle is one in which the ontologies are in constant evolution, highly dynamic and highly re-usable. Ontologies in biology have always supported a wide range of applications; the microarray ontology (MO) [47], for instance, is used by several, unrelated microarray laboratories information systems around the world. In both scenarios, SW and biology, not only is the structure of the ontology constantly evolving but also the

role of the knowledge engineer is not that of a leader but more that of a facilitator of collaboration and communication amongst domain experts.

Parallels can be drawn between the biological domain and the SW. The SW-related scenarios are often described as being distributed, loosely controlled and evolving [15]. The main differences between the classic proposals for building ontologies and those requirements applied to the SW have been summarised by Pinto et al. [15], as well as Garcia et al. [17], and are described in four key points:

1. Distributed information processing with ontologies: Within the SW scenario, ontologies are developed by geographically distributed domain experts willing to collaborate, whereas KE deals with centrally developed ontologies.
2. Domain expert-centric design: Within the SW scenario, domain experts guide the effort while the knowledge engineer assists them. There is a clear and dynamic separation between the domain of knowledge and the operational domain. In contrast, traditional KE approaches relegate the role of the expert as an informant to the knowledge engineer.
3. Ontologies are in constant evolution in SW, whereas in KE scenarios, ontologies are simply developed and deployed.
4. Additionally, within the SW scenario, fine-grained guidance should be provided by the knowledge engineer to the domain experts.

Collaboration is present in the DILIGENT, iCAPTURer, NeOn and GM methodologies. However, neither DILIGENT nor GM propose methods for engaging the collaborators, nor do they provide clear methodological guidelines. Alternatively, NeOn proposes a set of methods and techniques for most of the steps described. Nevertheless, the process of knowledge elicitation, whether within the context of collaboration or as a focus group activity, is not fully addressed in most of the methodologies investigated. METHONTOLOGY and NeOn consider knowledge elicitation as part of the methodology, but there are no recommendations regarding knowledge elicitation methods.

One important feature that is not covered by any of the methodologies investigated is the use of upper level ontologies; as these are meant to support classification based on universal criterion it is important to understand the structure proposed by these ontologies in order to ease the integration of domain ontologies.

Collaboration, knowledge elicitation, a better understanding of the ontology life cycle and detailed description for the different steps involved are important criteria that should be documented to ensure that methodologies may be more efficiently replicated and applied. There is also an increasing need to emphasis the reuse of methodologies rather than developing ad hoc, de novo methodologies. The reuse of methodologies will go some way to ensure efficient development, interoperability and the elucidation of best practice in ontology development, irrespective of the domain of application. These are precisely the issues that a methodology for community-based ontologies needs to address.

4.3 The Melting Point Methodology

The following outlines the Melting Point methodology and can serve as a "manual" for ontology engineering. As mentioned in Section 4.1, many of the techniques are best practices chosen from other methodologies, and as such extensive reference is made back to these methodologies. The MP methodology aims to provide ontology developers with a detailed view of the processes that should take place when building ontologies; it supports the orchestration of steps in the development process based on the inputs consumed and outputs produced by the methods and techniques used. MP is not prescriptive about specific techniques or methods; ontology developers should consider the use of those that best suit their particular situation. This document, as well as several deliverables from the NeOn project are a good source of information regarding the methods and techniques available.

For the purpose of MP, the activities involved are framed within processes and activities, as illustrated in Fig. 4.3; this conception is promoted by METHONTOLOGY [35] for centralised settings. As these activities were not conceived within decentralised settings, their scope has been redefined, so that they better fit the life cycle of ontologies developed by communities. The methodology here presented emphasises: decentralised settings and community involvement. It also stresses the importance of the life cycle these ontologies follow, and provides activities, methods and techniques coherently embedded within this life cycle.

The methodology and the life cycle are illustrated in Fig. 4.2. The overall process starts with documentation and management processes; the development process immediately follows. Managerial activities happen throughout the whole life cycle; as the interaction amongst domain experts ensures not only the quality of the ontology, but also that those predefined control activities take place. The development process has five main activities: specification, conceptualisation, formalisation implementation and evaluation. Different prototypes or versions of the ontology are thus constantly being created. Initially these prototypes may be unstable, as the classes and properties may drastically change. In spite of this, the process evolves rapidly, achieving a stability that facilitates the use of the ontology; changes become more focused on the inclusion of classes and instances, rather than on the redefinition of the class hierarchy.

4.3.1 Definition of Terminology

To aid the clarity of the methodology descriptions the interpretation of key terminology must be made clear. The meaning of the terms methodologies, techniques and methods follow the Institute of Electrical and Electronics Engineers (IEEE) descriptions [18, 19], as recommended by Perez-Gomez et al. [32], [48], Fernandez et al. [37] Pinto et al. [15, 49] and Garcia et al. [17]. Both Fernandez et al. and Perez-Gomez et al. emphasise the importance of complying with the Institute of Electrical and Electronics Engineers (IEEE) standards, more

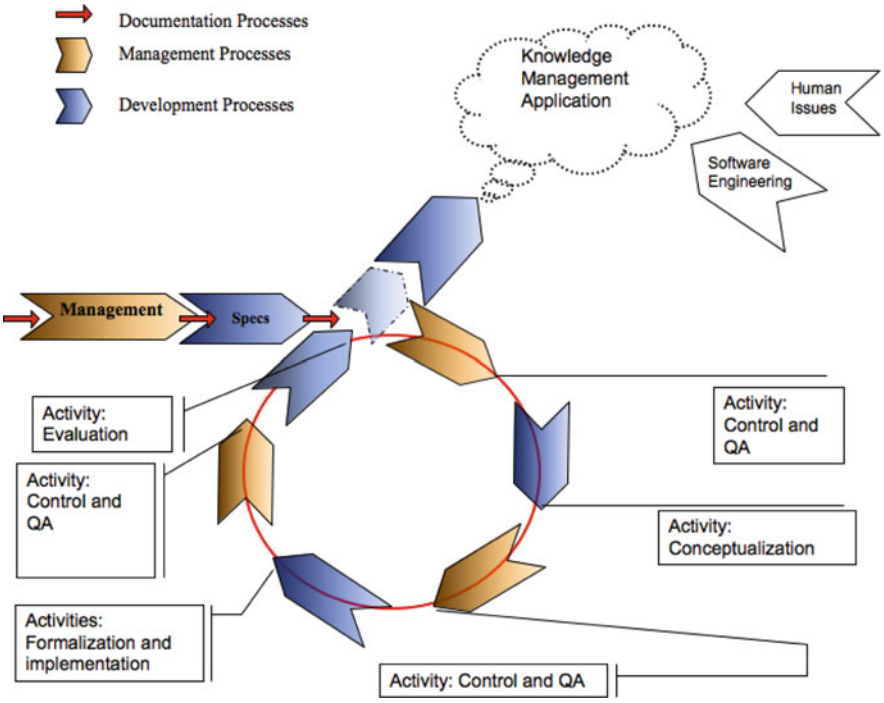

Fig. 4.2 Life cycle, processes, activities and view of the methodology

specifically with the IEEE standard for software quality assurance plans [50]. Not only does standards compliance ensure careful and systematic planning for the development, but it also ensures the applicability of the methodology to a broad range of problems. As such, we have also adopted terminology from the above-mentioned IEEE standard. A *methodology* should be interpreted as a "comprehensive integrated series of techniques or methods creating a general system theory of how a class of thought-intensive work ought to be performed" [18]. Methodologies are composed of both techniques and methods. A *method* is an "orderly" process or procedure used in the engineering of a product or performing a service [20]. A *technique* is a "technical and managerial procedure used to achieve a given objective" [18]. Thus methodologies bring together techniques and methods in an orchestrated way such that the work can be done. Figure 4.3 illustrates these relationships graphically.

Greenwood [51] and Gomez-Perez et al. [52] present these terminological relationships in a simple way: "a method is a general procedure while a technique is the specific application of a method and the way in which the method is executed" [52]. According to the IEEE [53] a process is a "function that must be performed in the software life cycle. A process is composed by activities". The same set of standards defines an activity as "a constituent task of a process" [53]. A task is the atomic unit

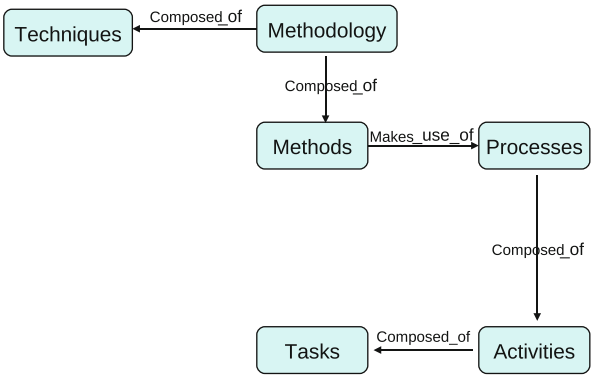

Fig. 4.3 Relationships amongst ontology engineering terms

of work that may be monitored, evaluated and/or measured; more formally, a task is "a well defined work assignment for one or more project member. Related tasks are usually grouped to form activities" [53].

4.3.2 Management Processes

The management activities are initiated as soon as there is a motivation (specification) and a decision for developing the ontology, therefore an artefact and a process to manage. The management process continues through the remainder of the ontology development process. Some of the activities involved in the management processes are

Scheduling. Scheduling identifies tasks, time and resources needed.
Control. Control ensures that the planned tasks are completed.
Inbound interaction. Inbound interaction specifies how the interaction amongst domain experts will take place, for instance, by phone calls, mailing lists, wiki and static Web pages.
Outbound interaction. As different communities should in principle be allowed to participate, there has to be an inclusion policy that specifies how a new community could collaborate and engage with the ongoing development.
Quality assurance. This activity defines minimal standards for the outputs from each and every process, activity or task carried out during the development of the ontology.

Scheduling project management techniques can be employed such as Gantt charts to and define milestones and deadlines. Several software suites exist to assist in project management, both commercial and open source. Specific technologies for documentation and communication are discussed in Garcia et al.17]. In addition,

more generic content management and communication systems can be employed for documenting and communicating the management process, such as those identified and reviewed by Mooney and Baenziger [54]. For both scheduling and controlling, the software tool(s) should in principle

– help to plan the activities and tasks that need to be completed,
– give a basis for scheduling when these tasks will be carried out,
– facilitate planning the allocation of resources needed to complete the project,
– help to work out the critical path for a project where one must complete it by a particular date,
– facilitate the interaction amongst participants and
– provide participants with simple means for exchanging information.

4.3.3 Documentation Processes

The documentation is a continuum process throughout the entire development of the ontology. This documentation should make it possible for new communities of practice to get involved in the development of the ontology. These include early processes such as the specification of the purpose of the ontology right through to later processes such as formalisation and evaluation of the ontology.

Documenting classes and properties. Although documentation can happen naturally, facilitated by discussions via an email basis, it is often difficult to follow the argumentative thread. Even so, the information contained in mailing lists is useful and should whenever possible be related to classes and properties. Use cases, in the form of examples for which the use of a term is well illustrated, should also be part of the documentation of classes and properties. Ontology editors allow domain experts to comment on the ontology; this kind of documentation is useful, as it reflects the understanding of the domain expert. For classes and properties there are three main sources of documentation:

Mailing lists. Discussions about why a class should be part of the ontology, why it should be part of a particular class hierarchy, how it is being used by the community, how a property relates two classes, and in general all discussions relevant to the ontology happen on an email basis.

On-the-ontology comments. In the cases when domain experts are familiarised with the ontology editor, they usually comment on classes and properties.

Use cases. This should be the main source of structured documentation provided by domain experts. However, gathering use cases is often difficult and time consuming. The use cases should illustrate how a term is being used in a particular context, how the term is related to other terms, and those different uses or meanings a term may have. Guidance is available for the construction of use cases when developing software; however, this direction is not available when building ontologies. From those experiences in which

the author participated some general guide can be drawn, for instance, use cases should be brief, they should be based upon real-life examples, knowledge engineers have to be familiar with the terminology as well as with the domain of knowledge because use cases are usually provided in the form of narratives describing processes, graphical illustrations should be part of the use case, and also whenever possible concept maps, or other related KA artefacts, should be used.

4.3.4 Development-Oriented Processes

These are the processes by which the ontology is actually built and represent the core of the methodology. The life cycle, documentation and management provide a framework in which development-oriented processes are embedded.

Specification. The specification of the ontology involves defining the motivation; in other words why the development of an ontology is required for the application domain. The specification phase can also be called a feasibility study and includes addressing straightforward questions such as "What is the ontology going to be used for?", "How is the ontology ultimately going to be used by the software implementation?", "What do we want the ontology to be aware of?" and "What is the scope of the knowledge we want to have in the ontology?". The answers to these questions are typically represented as competency questions, which define the requirements of the ontology. The requirements are dependent on the motivation and are described as informal questions or tasks that an ontology must be able to answer or perform. In other words, competency questions are those questions for which we want the ontology to be able to provide support for reasoning and inferring processes [17]. It is often helpful to include competency questions, as they can help to enforce the boundaries of the scope of the ontology.

Conceptualisation. The conceptualisation of the ontology is the process of identifying the key concepts that exist in the domain, their properties and the relationships that hold between them; this includes identifying natural language terms to refer to such concepts, relations and attributes as well as structuring domain knowledge into explicit conceptual models [55]. Gruber's design principles. [4] are relevant to the conceptualisation process as described below:

Gruber's first principle. "The conceptualisation should be specified at the knowledge level without depending on a particular symbol-level encoding."
Gruber's second principle. "Since ontological commitment is based on the consistent use of the vocabulary, ontological commitment can be minimised by specifying the weakest theory and defining only those terms that are essential to the communication of knowledge consistent with the theory."
Gruber's third principle. "An ontology should communicate effectively the intended meaning of defined terms. Definitions should be objective. Definitions can be stated on formal axioms, and a complete definition

(defined by necessary and sufficient conditions) is preferred over a partial definition. All definitions should be documented with natural language."

The process of conceptualisation typically involves the activities of domain analysis (DA) and knowledge elicitation (KE) and knowledge acquisition (KA). DA is the process by which a domain of knowledge is analysed in order to find common and variable components that best describe that domain. KE the process of collecting from a human source of knowledge, information that is relevant to that knowledge [44]. KA includes the elicitation, collection, analysis, modelling and validation of knowledge for knowledge engineering and knowledge management projects. The notion for both KA and KE comes from the development of knowledge bases; for the purposes of developing ontologies, KA and KE can be considered as transposable terms. KA and DA are interchangeable and complementary activities by which the information used in a particular domain is identified, captured and organised for the purpose of making it available in an ontology [56].

Those activities related to DA and KA focus more on capturing and representing knowledge in a more immediate manner and not necessarily on having logical expressions as part of the models; whereas when formalising and evaluating an ontology, activities and tasks are more oriented to include logical constrains and expressions. DA and KA may be seen as the art of questioning, since ultimately all relevant knowledge is either directly or indirectly in the heads of domain experts. This activity involves the definition of the terminology, i.e. the linguistic phase. This starts by the identification of those reusable ontologies and terminates with the baseline ontology, i.e. a draft version containing few but seminal elements of an ontology.

Identifying available sources of knowledge is also important; by doing so it can help to refine or confirm the ontology specification. In the bio-ontology domain this process can be facilitated by the OBO Foundry, which is a registry of available and accessible domain ontologies. Searching the registry can be made possible via the BioPortal from the National Center for Biomedical Ontology (NCBO) [57] or the Ontology Lookup Service[7]. The OLS provides a user-friendly single entry point for querying publicly available ontologies in the Open Biomedical Ontology (OBO) format. By means of the OLS it is possible to verify if an ontology term has already been defined and in which ontology is available [58].

The following criteria are important during knowledge acquisition [17]:

Accuracy in the definition of terms. The linguistic part of the ontology development is also meant to support the sharing of information/knowledge. The availability of context as part of the definition is useful when sharing knowledge.

Coherence. The narrative should be coherent; descriptions should make sense within the context in which they are intended to have a meaning.

[7] http://www.ebi.ac.uk/ontology-lookup/

Moreover, narratives should provide examples from which instances can be gathered.

Extensibility. This approach may be seen as an aggregation problem; CMs are constantly gaining information, which is always part of a bigger narration. Extending the conceptual model is not only about adding more detail to the existing CMs or just about generating new CMs; it is also about grouping concepts into higher-level abstractions and validating these with domain experts. Scaling the models involves the participation of both the domain experts and the knowledge engineer. It is mostly done by direct interview and confrontation with the models from different perspectives. The participation of new "fresh" domain experts, as well as the intervention of experts from allied domains, allows analysing the models from different angles. This participatory process allows re-factorising the models by increasing the level of abstraction.

The OBO Foundry has tried to define their criteria for defining terms. These OBO Foundry naming conventions[8] outline how to represent class labels and definitions to maintain consistency within one ontology and to provide a common naming conventions for integration across resources to avoid conflicts both at a human readable level and at a logical level.

For the purpose of DA and KA it is critical to elicit and represent knowledge from domain experts. They do not, however, have to be aware of knowledge representation languages; this makes it important that the elicited knowledge is represented in a language-independent manner. Researchers participating in knowledge elicitation sessions are not always aware of the importance of the session; however, they are aware of their own operational knowledge. This is consistent with the first of Gruber's design principles.

Regardless of the syntactic format in which the information is encoded domain experts have to communicate and exchange information. For this matter it is usually the case that wide general theories, principles, broad-scope problem specifications are more useful when engaging domain experts in discussions, as these tend to contain only essential basic terms, known across the community and causing the minimal number of discrepancies (see the second design principle). As the community engages in the development process and the ontology grows, it becomes more important to have definitions that are usable by computer systems and humans (see the third design principle). The relevant milestones, techniques and tasks for DA- and KA-related activities are

Tasks. Focal groups, limited information and constrained-processing tasks, protocol analysis, direct one-to-one interviews, terminology extraction, and inspection of existing ontologies.

[8] http://www.obofoundry.org/wiki/index.php/Naming

Techniques. Concept mapping, sorting techniques, automatic or semi-automatic terminology extraction, informal modelling and identifying pre-existing resources.

Milestones. Baseline ontology, knowledge sources, basic terminology, reusable ontologies.

Formalisation. Formalisation of the ontology is the activity during which the classes are constrained and instances are annotated against their corresponding classes. For example, "a male is constrained to be an animal with a y-chromosome". During the formalisation domain experts and knowledge engineers work with an ontology editor. When building iterative models and formalising the ontology the model grows in complexity; instances, classes and properties are added and logical expressions are built in order to have definitions with necessary and sufficient conditions. For both formalisation and iterative building of models, Gruber's fourth designing principle and Noy and McGuinness' guidelines [59] are applicable:

Gruber's fourth principle. "An ontology should be coherent: that is, it should sanction inferences that are consistent with the definitions. [. . .] If a sentence that can be inferred from the axioms contradicts a definition or example given informally, then the ontology is inconsistent."

Noy and McGuinness' first guideline. "The ontology should not contain all the possible information about the domain: you do not need to specialise (or generalise) more than you need for your application."

Noy and McGuinness' second guideline. "Subconcepts of a concept usually (i) have additional relations that the superconcept does not have or (ii) restrictions different from these of superconcepts or (iii) participate indifferent relationships than superconcepts. In other words, we introduce a new concept in the hierarchy usually only when there is something that we can say about this concept that we cannot say about the superconcept. As an exception, concepts in terminological hierarchies do not have to introduce new relations."

Noy and McGuinness' third guideline. "If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new concept for the distinction."

Implementation. The implementation of the ontology concerns the choice and justification of the encoding formalism, for example, the OBO format or the Web Ontology Language (OWL). The choice and the justification of a language take into account the required expressivity demanded by the specification process and by extension the tools required to facilitate the encoding. For example, if the chosen language was OWL, then it would be appropriate to use an ontology editor such as Protege[9]. Ultimately implementation is concerned with encoding the decisions

[9] http://protege.stanford.edu/

made as part of the formalisation process. However, the implementation process and the formalisation process can often happen simultaneously as an iterative process.

Iterative building of ontology models (IBOM). Iterative building of informal ontology models helps to expand the glossary of terms, relations, their definition or meaning, and additional information such as examples to clarify the meaning where appropriate. Different models are built and validated with the domain experts. There is a fine boundary between the baseline ontology and the refined ontology; both are works in progress, but the community involved has agreed upon the refined ontology.

Methods, techniques and milestones for the IBOM. Some milestones, techniques and tasks for IBOM related activities are

Tasks. Focal groups.
Techniques. Concept mapping, informal modelling with an ontology editor.
Milestones. Refined ontology.

4.3.5 Evaluation

There is no unified framework to evaluate ontologies and this remains an active field of research [32]. When developing ontologies on a community basis four main evaluation activities have been identified:

Specification evaluation. The specification defines the motivation and the scope of the ontology in the form of competency questions. Specification evaluation concerns the ability of the ontology to answer the competency questions and therefore demonstrate fulfilment of the intended scope.

Application-dependent evaluation. It is considered that ontologies should be evaluated according to their fitness for purpose, i.e. an ontology developed for annotation purposes should be evaluated by the quality of the annotation and the usability of the annotation software [17]. The community carries out this type of evaluation in an interactive manner; as the ontology is being used for several purposes a constant feedback is generated. The feedback thus gathered also helps in the evolution of the ontology; as the community comments on an ontology term being used to annotate a resource, ontology engineers are able to include, delete or edit terms in the ontology. This makes it possible for the community to effectively guarantee the usability and the quality of the ontology. By the same token, the recall and precision of the data, and the usability of the conceptual query builder, should form the basis of the evaluation of an ontology designed to enable data retrieval.

Terminology evaluation. This activity was proposed by Perez-Gomez et al. [60]. The goal of the evaluation is to determine what the ontology defines and how accurate these definitions are. Perez-Gomez et al. provides the following criteria for the evaluation:

Consistency. It is assumed that a given definition is consistent if, and only if, no contradictory knowledge may be inferred from other definitions and axioms in the ontology.

Completeness. It is assumed that ontologies are in principle incomplete [32, 60], however, it should be possible to evaluate the completeness within the context in which the ontology will be used. An ontology is complete if and only if *All that is supposed to be in the ontology is explicitly stated, or can be inferred*.

Conciseness. An ontology is concise if it does not store unnecessary knowledge, and the redundancy in the set of definitions has been properly removed.

Taxonomy evaluation. This evaluation is usually carried out by means of reasoned systems such as RACER [61] and Pellet [62]. The knowledge engineer checks for inconsistencies in the taxonomy, these may due to errors in the logical expressions that are part of the axioms.

4.4 Discussion

4.4.1 Melting Point Evaluated

The Melting Point (MP) methodology emphasises an integral knowledge management cycle. It is influenced by METHONTOLOGY and the work done by Sure in the field of knowledge management. The MP makes use of several methods and techniques, defining the steps which should be undertaken. The MP methodology stresses the importance of the orchestration of methods and techniques based on coherence between outcomes and deliverables for each step, thus proposing a flexible structure that can be adapted without losing rigor in the process. When evaluated against the criteria presented in Section 4.1, the MP methodology can be seen to have the following properties:

C1. *Inheritance from knowledge engineering*. Highly influenced by knowledge engineering.
C2. *Detail of the methodology*. Although it defines steps the MP methodology stresses the importance of an orchestration based on those outcomes and deliverables from each step. The MP aims for a flexible rigor, rather than a strict series of steps.
C3. *Strategy for building the ontology*.

C3.1. *Application of the ontology*. application independent.
C3.2. *Domain experts*. The methodology is intended to make use of knowledge gathered from all levels of domain experts. It is assumed an active participation of domain experts.
C3.3. *Ontology type*. The methodology is best suited for domain ontologies.

C4. *Strategy for identifying concepts*. Concepts are identified by a variety of methods and techniques; the MP does not enforce the use of a particular method or technique; it proposes processes for which there can be several methods and techniques available. It assumes an active participation of domain experts in that for the MP methodology domain experts are also modelers.
C5. *Recommended life cycle*. Processes, activities and tasks are proposed and orchestrated within an incremental evolutionary spiral model.
C6. Recommended methods and techniques.] The MP methodology proposes some methods and techniques; these are, however, changeable as the methodology does not emphasise the use of particular methods and techniques but rather stresses the impotence of an orchestrated Knowledge management process.
C7. *Applicability*. Parts of the proposed methodology have been applied and reported [17, 63]. The MP methodology is based upon these experiences and on the observation of several ontology development processes such as the CARMEN project[10].
C8. *Community involvment*. Active steps are taken to ensure that the community takes a leading role in the development process.
C9. *Knowledge elicitation*. Knowledge elicitation is an integral part of the overall process.

4.4.2 IEEE Standards Compliance

As discussed by [34], METHONTOLOGY is the only methodology that rigorously complies with IEEE standards; this facilitates the applicability and extendibility of the methodology. Other methodologies, such as those studied by [42] do not intentionally meet the terms posed by the IEEE. However, some of the proposed activities by those ontologies may be framed within IEEE standards. The Melting Point methodology proposed here reuses and adapts many components from METHONTOLOGY and other methodologies within the context of decentralised settings and participatory design. It also follows Sure's [9] work as it considers throughout the whole process the importance of the software applications that will ultimately use the ontology. The work done by Sure is complementary to the one presented in this chapter, as both works study different edges of the same process: developing knowledge-based software.

4.4.3 Quality Assurance

METHONTOLOGY allows for a controlled development and evolution of the ontology placing special emphasis on quality assurance (QA) thought the processes.

[10] http://carmen.org.uk/

Although QA is considered, the authors do not propose any methods for this specific task. Management, development and support activities are carried out in a centralised manner; a limited group of domain experts interact with the knowledge engineer, conceptualise and prototype the ontology, successive prototypes are then built, the ontology gains more formality (e.g. logical constraints are introduced) until it is decided that the ontology may be deployed. Once the ontology has been deployed a maintenance process takes place. Neither the development nor the evolution of the ontology involves a decentralised community; the process does not assume a constant incremental growth of the ontology as it has been observed, and reported by [17] QA is also considered to be a centralised activity, contrasting with the way decentralised ontologies promote the participation of the community in part to ensure the quality of the delivered ontology.

4.4.4 Activities Become Interrelated

As those required ontologies grow in complexity so does the process by which they are obtained. Methods, techniques, activities and tasks become more group-oriented, making it necessary to re-evaluate the whole process as well as the way by which it is described. The IEEE proposes a set of concepts that should in principle facilitate the description of a methodology; however, these guidelines should be better scoped for decentralised environments.

Activities within decentralised ontology developments are highly interrelated. However, the maturity of the product allows engineers and domain experts to determine boundaries and by doing so establishing milestones for each and every activity and task. Although managerial activities are interrelated and impact at a high level those development processes it is advisable not to have rigid management structures. For instance, control and inbound–outbound activities usually coexist with some development activities when a new term needs to be added. This interaction requires the orchestration of all the activities to ensure the evolution of the ontology. This interaction and orchestration of activities with defined boundaries and milestones are evident in the bio-ontology domain from the development of the Proteomics Standards Initiatives (PSI) sample processing and separations controlled vocabulary, sepCV. The PSI aims to facilitate global proteomics models for publication, data storage,data comparisons and dataintegration and to standardise and advance proteomics research [64]. To this end, they have developed minimum reporting guidelines [65], data transfer formats and ontologies to control the terminology used for reporting. The sepCV ontology had an initial specification and therefore milestones to represent the technology of gel electrophoresis [66]. However, its scope was then expanded to cover gel image informatics, so the life cycle continued collecting and representing community-defined concepts for both gel electrophoresis and gel informatics. In addition to these two technologies the sepCV is also expected to expand its specification to cover other separation technologies, such as column chromatography and capillary electrophoresis, with the

consequences that these interactions require the orchestration of all the activities to ensure the evolution of the ontology to fit dynamic boundaries and expanding specification over its life cycle.

4.4.5 Recommended Life Cycle: Incremental Evolutionary Spiral

When communities are developing ontologies the life cycle varies. The ontology is not deployed on a one-off basis; there is thus no definitive final version of the ontology. The involvement of the community allows for rapid evolution, as well as for very high-quality standards; errors are identified and discussed then corrections are made available within short time frames.

The model upon which this proposed methodology is based brings together ideas from, linear sequential modelling [29, 67], prototyping, spiral [68], incremental [69, 70] and the evolutionary models [29, 71]. Due to the dynamic nature of the interaction when developing ontologies on a community basis the model grows rapidly and continuously. As this happens prototypes are being delivered, documentation is constantly being generated, and evaluation takes place at all times as the growth of the model is due to the argumentation amongst domain experts. The development process is incremental as new activities may happen without disrupting the evolution of the collaboration. The model is therefore an incremental evolutionary spiral in which tasks and activities can coexist simultaneously at some level of detail. As the process moves forward activities and/or tasks are applied recursively depending on the needs. The evolution of the model is dynamic and the interaction amongst domain experts and with the model happens all the time. Figure 4.4 illustrates the model as well as how processes, activities and tasks are consistent with the model.

4.5 Conclusions

The Melting Point methodology proposed here reuses some components that various authors have identified as part of their methodologies for ontology development. This chapter has investigated how to use these components within decentralised settings, using the biomedical domain as an example. A domain where community development is critical to understanding a large, complex and an ever expanding knowledge base. Not only can the Melting Point methodology be demonstrated in the life science domain, the methodology can also be applicable to the development of the knowledge infrastructure of the Semantic Web, a decentralised environment by design.

The Melting Point methodology stresses the importance of a detailed description of the methods, techniques, activities, and tasks that could be used for developing community-based ontologies. Furthermore, a discussion of how the development process evolves adapts and expands with increasing or redefining of the ontology specification is presented within the life cycle model of these ontologies.

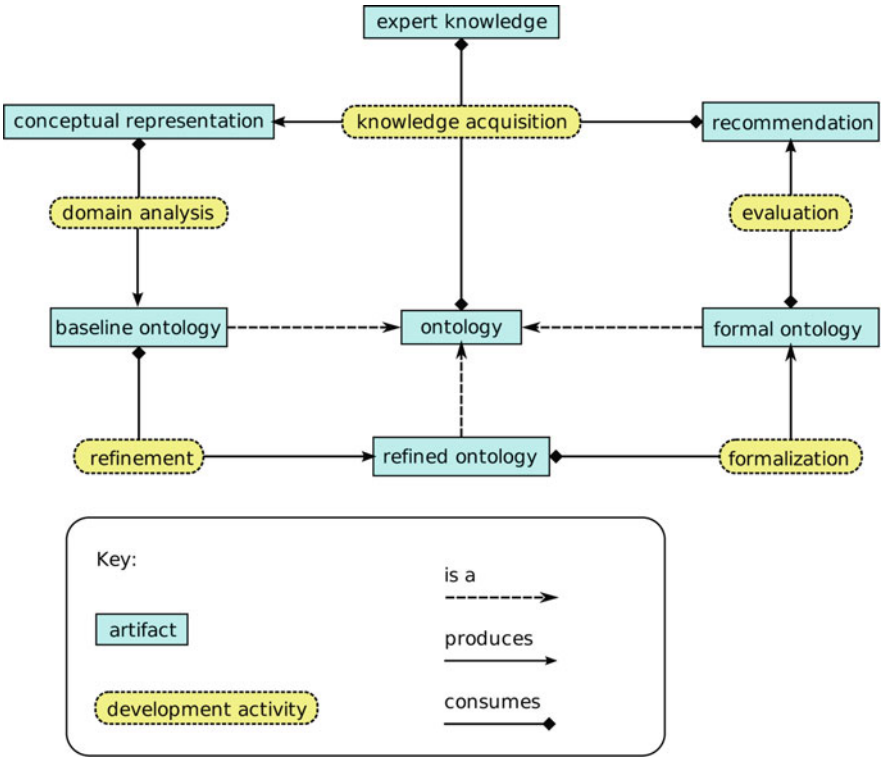

Fig. 4.4 A view of the whole process, showing development activities and the artefacts they produce and consume

The adoption of the Melting Point methodology should provide a level of rigour and consistent development of ontologies, with a particular focus on community contribution. The methodology may facilitate a process by which the OBO Foundry principles for bio-ontology development[11] can be achieved. Ontologies developed within the same methodology framework may aid in increasing ontology interoperability and integration, as the processes and design decisions can be disseminated upon publication and therefore followed and evaluated.

As we increasingly build large ontologies against complex domain knowledge in a community and collaborative manner there is an identified need for a methodology to provide a framework for this process. A shared methodology tailored for the decentralised development environment, facilitated by the Internet should increasingly enable and encourage the development of ontologies fit for purpose. The Melting Point methodology provides this framework which should enable the ontology community to cope with the escalating demands for scalability and

[11] http://www.obofoundry.org/crit.shtml

repeatability in the representation of community-defined knowledge bases, such as those in biomedicine and the Semantic Web.

A. Appendix: Review of Methodologies

A.1 The Enterprise Methodology

Uschold and King proposed a set of four activities that are listed here and illustrated in Fig. 4.5

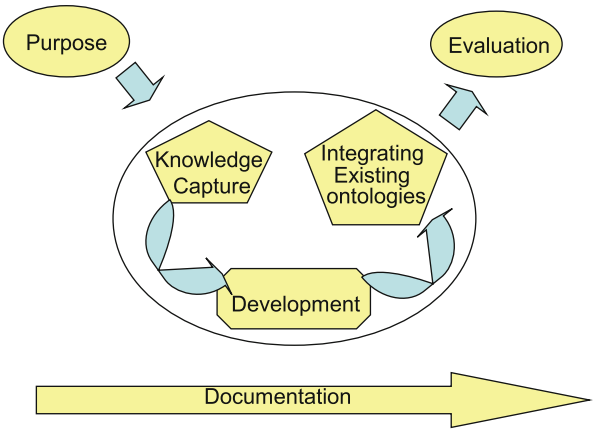

Fig. 4.5 Uschold and King methodology

1. Identify the purpose and scope of the ontology
2. Build the ontology, for which they specify three activities:
   - Knowledge capture
   - Development / coding
   - Integrating with other ontologies
3. Evaluate
4. Document the ontology

C1. The methodology does not explicitly inherit methods from knowledge engineering. Although Uschold and King identify steps that are in principle related to some methodologies from knowledge engineering. Neither a feasibility study nor a prototype method is proposed.

C2. Stages are identified, but no detail is provided. In particular the Ontology Coding Integration and Evaluation sections are presented in a superfluous manner [18].

C3. Limited information is provided. The proposed method is application independent and very general, in principle it is applicable to other domains. The authors do not present information about the kind of domain experts they advise working with.
C4. Uschold and Kind do not provide a clear criterion for the selection of either approach. For Uschold and King the disadvantage of using the top-down approach is that by starting with a few general concepts there may be some ambiguity in the final product. Alternatively, with the bottom-up approach too much detail may be provided and not all this detail could be used in the final version of the ontology [41]. This in principle favours the middle-out approach proposed by Lakoff [43]. The middle-out is not only conceived as a middle path between bottom-up and top-down, but also relies on the understanding that categories are not simply organised in hierarchies from the most general to the most specific, but are rather organised cognitively in such a way that categories are located in the middle of the general-to-specific hierarchy. Going up from this level is the generalisation and going down is the specialisation [43, 18].
C5. No life cycle is recommended.
C6. No techniques or methods are recommended. The authors mention the importance of representing the captured knowledge but do not make explicit recommendations as to which knowledge formalism to use. This methodology does not support any particular software as a development tool. The integration with other ontologies is not described, nor is any method recommended to overcome this issue, nor is whether this integration involves extending the generated ontology or merging it with an existing one explained.
C7. The methodology was used to generate the Enterprise ontology [10].
C8. Communities are not involved in this methodology.
C9. For those activities specified within the building stage the authors do not propose any specific method for representing the ontology (e.g. frames, description logic). The authors place special emphasis on knowledge elicitation. However, they are not specific in developing this further.

A.2 The TOVE Methodology

The Toronto Virtual Enterprise (TOVE) methodology involves building a logical model of the knowledge that is to be specified by means of an ontology. The steps involved as well as their corresponding outcomes are illustrated in Fig. 4.6.

C1. The methodology is heavily influenced by the development of knowledge-based systems using first-order logic [36].
C2. No specifics are provided on the activities involved.
C3. The TOVE methodology emphasises competency questions as well as motivating scenarios as important components in their methodology. This methodology is application semidependent as specific terminology is used not only to

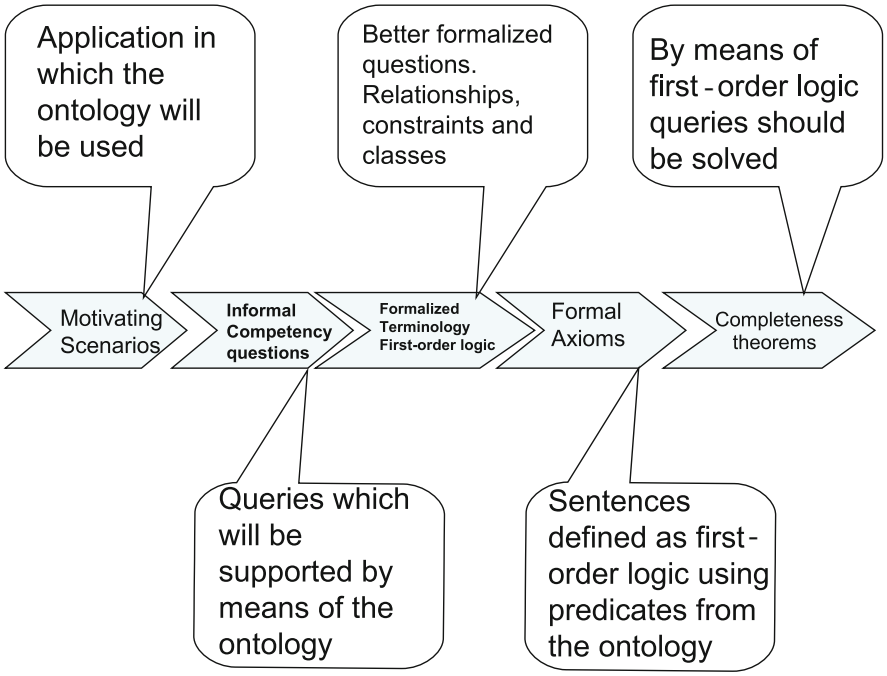

Fig. 4.6 The TOVE methodology

formalise questions but also to build the completeness theorems used to evaluate the ontology. Once the competency questions have been formally stated, the conditions under which the solutions to the questions must be defined should be formalised. The authors do not present information about the kind of domain experts they advise working with.

C4. This methodology adopts a middle-out strategy.
C5. No indication about a life cycle is given.
C6. The importance of competency questions are emphasised. However, they do not provide techniques or methods to approach this problem.
C7. The Toronto Virtual Enterprise ontology was built using this methodology [72].
C8. Communities are not involved in this methodology.
C9. No particular indication for eliciting knowledge is given.

A.3 The Bernaras Methodology

Bernaras work was developed as part of the KACTUS [12] project which aimed to investigate the feasibility of knowledge reuse in technical systems.

C1. This methodology is thus heavily influenced by knowledge engineering.
C2. Limited detail about the methodology is provided.

C3. This methodology is application dependant. As the development of this methodology took place within a larger engineering effort ontologies were being developed hand-in-hand with the corresponding software. This implies that domain experts were being used for both tasks, for requirements interviews and studies as well as for ontology development. This, however, does not mean that domain experts were taking an active role. The authors present very little information about the kind of domain experts they advise working with.
C4. This methodology adopts a bottom-up approach [36].
C5. As the ontology is highly coupled with the software that uses it, the life cycle of the ontology is the same as the software life cycle.
C6. For the specific development of the ontology no particular methods or techniques are provided. However, as this methodology was meant to support the development of an ontology at the same time as the software it is reasonable to assume that some software engineering methods and techniques were also applied to the development of the ontology.
C7. It has been applied within the electrical engineering domain.
C8. Communities are not involved in this methodology
C9. No particular indication for knowledge elicitation is provided.

A.4 The METHONTOLOGY Methodology

The authors of METHONTOLOGY aim to define a standardisation of the ontology life cycle (development) with respect to the requirements of the Software Development Process (IEEE 1074-1995 standard) [18]. The METHONTOLOGY methodology is illustrated in Fig. 4.7.

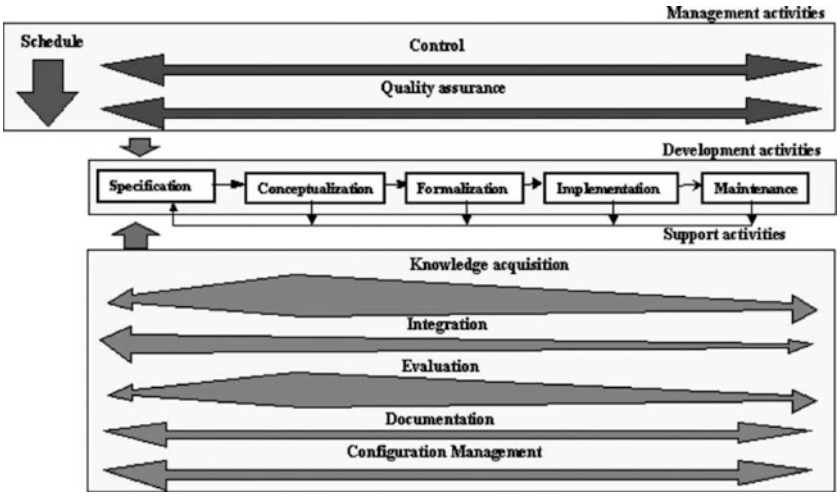

Fig. 4.7 METHONTOLOGY. Reproduced with permission from [36]

C1. METHONTOLOGY has its roots in knowledge engineering.
C2. Detail is provided for the ontology development process; Fig. 4.7 illustrates the methodology. It includes the identification of the ontology development process, a life cycle based on evolving prototypes, and particular techniques to carry out each activity [36]. This methodology heavily relies on the IEEE software development process as described in [50]. Gomez-Perez et al. [52] consider that all the activities carried out in an ontology development process may be classified into one of the following three categories:

    1. Management activities: Including planning, control and quality assurance. Planning activities are those aiming to identify tasks, time and resources.
    2. Development activities: Including the specification of the states, conceptualisation, formalisation, implementation and maintenance. From those activities related to the specification knowledge engineers should understand the context in which the ontology will be used. Conceptualisation activities are mostly those activities in which different models are built. During the formalisation phase the conceptual model is transformed into a semi-computable model. Finally, the ontology is updated and corrected during the maintenance phase [42].
    3. Support activities: These include knowledge elicitation, evaluation, integration, documentation, and configuration management.

C3. Application independent. No indication is provided as to the kind of domain experts they advise working with. In principle METHONTOLOGY could be applied to the development of any kind of ontology.
C4. This methodology adopts a middle-out
C5. METHONTOLOGY adopts an evolving-prototype life cycle.
C6. No methods or techniques recommended. METHONTOLOGY heavily relies on WebODE [45] as the software tool for coding the ontology. However, this methodology is in principle independent from the software tool.
C7. This methodology has been used in the development of the Chemical OntoAgent [73] as well as in the development of the Onto2Agent ontology [73].
C8. No community involvement is considered.
C9. Knowledge elicitation is part of the methodology. However, no indication is provided as to which method to use.

A.5 The SENSUS Methodology

The SENSUS-based methodology [14] is a methodology supported on those experiences gathered from building the SENSUS ontology. SENSUS is an extension and reorganisation of WordNet [74], this 70,000-node terminology taxonomy may[12]

[12] http://www.isi.edu/natural-language/projects/ONTOLOGIES.html

be used as a framework into which additional knowledge can be placed [75]. SENSUS emphasises merging pre-existing ontologies and mining other sources such as dictionaries.

C1. SENSUS is not influenced by knowledge engineering as this methodology mostly relies on methods and techniques from text mining.
C2. Although there is extensive documentation for those text-mining techniques and developing structures for conceptual machine translation [75–77] no detail is provided as for how to build the ontology.
C3. As SENSUS makes extensive use of both text mining and conceptual machine translation the methodology as such is application semi-independent. The methods and techniques proposed by SENSUS may, in principle, be applied to several domains.
C4. SENSUS follows a bottom-up approach. Initially instances are gathered, as the process moves forward abstractions are then identified.
C5. No life cycle is identified; from those reported experiences the ontology is deployed on a one-off basis.
C6. Methods and techniques are identified for gathering instances. However, no further detail is provided.
C7. SENSUS was the methodology followed for the development of knowledge-based applications for the air campaign planning ontology [78].
C8. No community involvement is considered.
C9. Knowledge elicitation is not considered explicitly.

A.6 DILIGENT

Diligent (DIstributed, Loosely controlled and evolvInG Engineering of oNTologies) was conceived as a methodology for developing ontologies on a community basis. Although the DILIGENT approach assumes the active engagement of the community of practice throughout the entire process, it does not give extensive details. Some particularities may be found reported for those cases in which DILIGENT has been used, for instance [15].

C1. DILIGENT is influenced by knowledge engineering as this methodology has been developed assuming the ontologies will be used by knowledge-based systems. However, DILIGENT introduces novel concepts such as the importance of the evolution of the ontology and the participation of communities within the development and life cycle of the ontology.
C2. DILIGENT provides some details specifically for those developments in which it has been used.
C3. DILIGENT is application dependant. There is no indication about the kind of domain experts they advise working with.
C4. The selection between top-down, bottom-up or middle-out is problem dependent. No indication is given as to which strategy would be best to follow.

C5. DILIGENT assumes an iterative life cycle in which the ontology is in constant evolution.
C6. In principle DILIGENT does not recommend methods or techniques. By the same token DILIGENT is not linked to any software supporting, either the development or the collaboration.
C7. Some cases for which DILIGENT has been used have been reported, for instance, the study of legal cases [79].
C8. The involvement of communities is considered in this methodology.
C9. Although knowledge elicitation is considered in this methodology no special emphasis is placed on it.

A.7 The GM Methodology

The GM methodology emphasises on knowledge acquisition when developing ontologies within decentralised settings. Similar to DILIGENT, the GM methodology was engineered for scenarios in which geographically distributed domain experts were working together on the same ontology. The GM methodology makes use of conceptual maps to support the acquisition of knowledge. In contrast to the DILIGENT methodology, the GM methodology provides a detailed description of the process applied within their development scenario.

C1. GM applies knowledge engineering principles.
C2. A detailed description of the methods and techniques used are provided.
C3. GM is application dependant. GM assumes the participation of both specialised and broader knowledge domain experts.
C4. A top-down approach is applied within GM.
C5. GM assumes an iterative life cycle in which the ontology is in constant evolution.
C6. Methods and techniques for some of the stages of the development process are recommended.
C7. GM has been used within the biomedical domain [79].
C8. GM assumes an active participation of the community.
C9. GM has an emphasises on knowledge elicitation.

A.8 The iCapturer Methodology

The GM methodology emphasises on knowledge acquisition within decentralised settings. Unlike GM and DILIGENT, iCapturer [38] makes use of text-mining approaches, such as text-to-onto, to identify important terms and to suggest candidate ontological relationships between them.

C1. The iCAPTURer approach has received little influence from knowledge engineering.
C2. The iCAPTURer methodology is very specific in terms of the orchestration of methods used. The first step is term and relationship extraction from text containing domain knowledge. The second is web-based, massively collaborative correction, refinement and extension of the automatically extracted concepts and relationships. The second step may be divided into phases of knowledge elicitation, evaluation, and aggregation.

C3.1. *Application of the ontology*. Application independent.
C3.2. *Domain experts*. The methodology is intended to make use of knowledge gathered from all levels of domain experts. It is assumed that the pool of experts contains all of the knowledge that is intended to be represented in the ontology.
C3.3. *Ontology type*. The methodology is best suited for domain ontologies.

C4. *Strategy for identifying concepts*. The strategy for identifying concepts is to extract representative terms automatically from text. Though this will typically result in what appears to be a more bottom-up approach, different bodies of text will produce different results.
C5. *Recommended life cycle*. The recommended life cycle is to

1. identify a domain of knowledge to be represented in an ontology,
2. identify a corpus thought to contain that knowledge,
3. apply text-mining approaches, such as text-to-onto, to identify important terms and to suggest candidate ontological relationships between them,
4. define user-interfaces for correcting and extending this knowledge,
5. assemble a broad array of experts in the domain and engage them in using the interface to improve the ontology,
6. evaluate the quality of each contributor based on expected correct interactions with the knowledge elicitation system,
7. weight their contributions based on this level of quality,
8. aggregate the contributions of all of the experts so that a candidate ontology can be generated and
9. iterate and refine as needed.

C6. *Recommended methods and techniques*. The methodology specifies the process but does not suggest any specific method. Several text-mining algorithms or knowledge gardening interfaces might be applied depending on the domain and the community.
C7. *Applicability*. iCAPTURer has not yet been applied in real scenarios.
C8. *Community involvement*. The community is assumed to develop the ontology.
C9. *Knowledge elicitation*. Knowledge elicitation is an integral part of the methodology. iCAPTURer describes some techniques for KE, but there is also wide room for expansion and adaptation of other methods.

A.9 NeOn Methodology

NeOn[13] is a framework for developing networked ontologies. It is one of the most comprehensive works in terms of ontology engineering. The framework incorporates a methodology.

C1. Highly influenced by knowledge engineering.
C2. It defines those steps that should be undertaken when developing ontologies.
C3.

- C3.1. *Application of the ontology*. Application independent.
- C3.2. *Domain experts*. It assumes an active participation of domain experts and ontology engineers.
- C3.3. *Ontology type*. The methodology is best suited for domain ontologies.

C4. *Strategy for identifying concepts*. No particular detail is provided for identifying concepts.
C5. *Recommended life cycle*. Project aims specifically to support life cycle activities, but does not prescribe a particular type of life cycle.
C6. *Recommended methods and techniques*. It provides specifics for methods and techniques.
C7. *Applicability*. The methodology is proposed based on cases that have been studied; however, it is not clear which ontology has been developed applying the proposed framework.
C8. *Community involvement*. It assumes collaboration and the involvement of a community of practice.
C9. *Knowledge elicitation*. Knowledge elicitation is recognised to play a significant role during the development process.

References

1. Editorial: Compete, collaborate, compel. Nat Genet 39(8) (Aug 2007) 931
2. Julian, S., J. Rector, A.: The state of multi-user ontology engineering. In: Proceedings of the 2nd International Workshop on Modular Ontologies, Canada (2007)
3. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Lewis, S.: The obo foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11) (2007) 1251–1255
4. Gruber, T.: Collective knowledge systems: Where social web meets the semantic web. In: Proceedings of the 5th International Semantic Web Conference, Athens, GA, USA (2006)
5. Tudorache, T., Noy, N.: Collaborative protege. In: Social and Collaborative Construction of Structured Knowledge, Proceedings of 16th International WWW Conference, Alberta, Canada (2007)
6. Feigenbaum, E., McCorduck, P.: The Fifth Generation. Addison-Wesley, Reading, MA (1983)

[13] http://www.neon-project.org

7. Kendal, S., Creen, M.: An Introduction to Knowledge Engineering. Springer, New York, NY (2007)
8. Sowa, J.: Knowledge Representation: Logical, Philosophical, and Computational Foundation. Brooks Cole Publishing, Pacific Grove, CA (2000)
9. Sure, Y.: Methodology, Tools and Case Studies for Ontology Based Knowledge Management. PhD Thesis, Universitat Fridericiana zu Karlsruhe (2003)
10. Uschold, M., King, M.: Towards methodology for building ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing, Held in Conjunction with IJCAI-95. Cambridge, UK (1995)
11. Gruninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Proceedings of the IFIP WG5.7 Workshop on Benchmarking – Theory and Practice, Trondheim, Norway (1994)
12. Bernaras, A., Laresgoiti, I., Corera, J.: Building and reusing ontologies for electrical network applications, 12th European Conference on Artificial Intelligence ECAI. Wiley, Budapest, Hungary (1996) 298–302
13. Fernadez-Lopez, M., Perez, A.G., Pazos, S.J., Pazos, S.A.: Building a chemical ontology using methontology and the ontology design environment. IEEE Intelligent Systems and Their Applications 14 (1999) 37–46
14. Swartout, B., Ramesh, P., Knight, K., Russ, T.: Toward distributed use of largescale ontologies. In: Symposium on Ontological Engineering of AAAI, Stanford, California (1997)
15. Pinto, H.S., Staab, S., Tempich, C.: Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In: European Conference on Artificial Intelligence, Valencia, Spain (2004) 393–397
16. Vrandecic, D., Pinto, H.S., Sure, Y., Tempich, C.: The diligent knowledge processes. Journal of Knowledge Management 9(5) (2005) 85–96
17. Garcia, C.A., Rocca-Serra, P., Stevens, R., Taylor, C., Nashar, K., Ragan, M.A., Sansone, S.: The use of concept maps during knowledge elicitation in ontology development processes – the nutrigenomics use case. BMC Bioinformatics 7 (2006) 267
18. Mirzaee, V.: An Ontological Approach to Representing Historical Knowledge. MSc Thesis. PhD Thesis, Department of Electrical and Computer Engineering, University of British Columbia (2004)
19. Moreira, D., Musen, M.A.: Obo to owl: A protege owl tab to read/save obo ontologies. Bioinformatics 23(14) (2007) 1868–1870
20. Sathiamurthy, M., Peters, B., Bui, H.H., Sidney, J., Mokili, J., Wilson S.S., Fleri, W., McGuinness, D., Bourne, P., Sette, A.: An ontology for immune epitopes: Application to the design of a broad scope database of immune reactivities. BMC Immunology 1(2) (2005)
21. Bada, M., Stevens, R., Goble, C., Gil, Y., Ashbourner, M., Blake, J., Cherry, J., Harris, M., Lewis, S.: A short study on the success of the geneontology. Journal of Web Semantics 1 (2004) 235–240
22. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific American 284(5) (2001) 28–37
23. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems (2006) 96–101
24. Degtyarenko, K., Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcantara, R., Darsow, M., Guedj, M., Ashburner, M.: ChEBI: A database and ontology for chemical entities of biological interest. Nucleic Acids Research (2007)
25. Smith, B., Kumar, A., Bittner, T.: Basic formal ontology for bioinformatics. Retrieved Jul. 12, 2010 from http://www.uni-leipzig.de/~akumar/JAIS.pdf Journal of Information Systems (2005) 1–16
26. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening Ontologies with Dolce. Lecture Notes in Computer Science (2002) 166–181

27. Herre, H., Heller, B., Burek, P., Hoehndorf, R., Loebe, F., Michalek, H.: General Formal Ontology (GFO) – A Foundational Ontology Integrating Objects and Processes. Onto-Med Report 8
28. Eden, H.A., Hirshfeld, Y.: Principles in formal specification of object oriented design and architecture. In: Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research, Toronto, Canada, IBM Press (2001)
29. Pressman, S.R.: Software Engineering, A Practitioners Approach. 5th edn. McGraw-Hill Series in Computer Science. Thomas Casson, New York, NY (2001)
30. Martin, J.: Rapid Application Development. Prentice-Hall, Englewood Cliffs, NJ (1991)
31. Gilb, T.: Evolutionary project management: Multiple performance, quality and cost metrics for early and continuous stakeholder value delivery. In: International Conference on Enterprise Information Systems, Porto, Portugal (2004)
32. Perez, A.G.: Some Ideas and Examples to Evaluate Ontologies. Technical Report, Stanford University (1994a)
33. Gilb, T.: Principles of Software Engineering Management. Addison-Wesley Longman, Boston, MA (1988)
34. Garcia, A.: Developing Ontologies Within the Biomedical Domain. PhD, University of Queensland (2007)
35. Fernandez, M.: Overview of methodologies for building ontologies. In: Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5), Stockholm, Sweden (1999)
36. Corcho, O., Fernadez-Lopez, M., Gomez-Perez, A.: Methodologies, tools, and languages for building ontologies. Where is their meeting point? Data and Knowledge Engineering 46(1) (2003) 41–64
37. Fernandez, M., Gomez-Perez, A., Juristo, N.: Methontology: From ontological art to ontological engineering. In: Workshop on Ontological Engineering. Spring Symposium Series. AAAI97, Stanford (1997)
38. Good, B., Tranfield, E.M., Tan, P.C., Shehata, M., Singhera, G., Gosselink, J., Okon, E.B., Wilkinson, M.: Fast, cheap, and out of control: A zero curation model for ontology development. In: Pacific Symposium on Biocomputing. Maui, Hawaii, USA. (2006)
39. Van Heijst, G., Van der Spek, R., Kruizinga, E.: Organizing corporate memories. In: Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96). (1996)
40. Mizoguchi, R., Vanwelkenhuysen, J., Ikeda, M.: Task ontology for reuse of problem solving knowledge. In: Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS'95). (1995) 46–57
41. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review 11 (1996) 93–136
42. Fernadez-Lopez, M., Gomez-Perez, A.: Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review 17(2) (2002) 129–156
43. Lakoff, G.: Women, Fire, and Dangerous Things: What Categories Reveal About the Mind. Chicago University Press, Chicago (1987)
44. Cooke, N.: Varieties of knowledge elicitation techniques. International Journal of Human-Computer Studies 41 (1994) 801–849
45. Arpirez, J., Corcho, O., Fernadez-Lopez, M., Gomez-Perez, A.: Webode in a nutshell. AI Magazine 24(3) (2003) 37–47
46. Hinchcliffe, D.: Dion hinchcliffe's web 2.0 blog web 2.0 (2008)
47. Stoeckert, C.J., Parkinson, H.: The mged ontology: A framework for describing functional genomics experiments. Comparative and Functional Genomics 4 (2003) 127–132
48. Perez, A.G., Juristo, N., Pazos, J.: Evaluation and assessment of knowledge sharing technology. In Mars, N. (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing(KBK95), IOS Press, Amsterdam, The Netherlands, (1995) 289–296
49. Pinto, H.S., Martins, P. J.: Ontologies: How can they be built? Knowledge and Information Systems 6 (2004) 441–463

50. IEEE: IEEE standard for software quality assurance plans (1998)
51. Greenwood, E.: Metodologia de la investigacion social. Paidos, Buenos Aires (1973)
52. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: Ontological Engineering. Springer, London (2004)
53. IEEE: IEEE standard for developing software life cycle processes (1996)
54. Mooney, S.D., Baenziger, P.H.: Extensible open source content management systems and frameworks: A solution for many needs of a bioinformatics group. Brief Bioinform 9(1) (Jan 2008) 69–74
55. Stevens, R., Goble, C., Bechhofer, S.: Ontology-based knowledge representation for bioinformatics. Briefings in Bioinformatics (2000) 398–414
56. Gaines, B.R., Shaw, M.L.Q.: Knowledge acquisition tools based on personal construct psychology. The Knowledge Engineering Review 8(1) (1993) 49–85
57. Rubin, D., Lewis, S., Mungall, C., Misra, S., Westerfield, M., Ashburner, M., Sim, I., Chute, C., Solbrig, H., Storey, M., Smith, B., Day-Richter, J., Noy, N., Musen, M.: National center for biomedical ontology: Advancing biomedicine through structured organization of scientific knowledge. OMICS 10(2) (2006) 85–98
58. Cote, R., Jones, P., Apweiler, R., Hermjakob, H.: The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. BMC Bioinformatics 7(97) (2006)
59. Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report, Stanford University (2001)
60. Perez, A.G., Fernadez-Lopez, M., Corcho, O.: Ontological Engineering. Computer Sciences. Springer. London (2004)
61. Haarslev, V., Mller, R.: Racer: A core inference engine for the semantic web. In: Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), Sanibel Island, Florida, USA (2003) 27–36
62. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl resoner. Journal of Web Semantics 5(2) (2007)
63. Garcia, A., Zhang, Z., Rajapakse, M., Baker, C., Tang, S.: Capturing and modeling neuro-radiological knowledge on a community basis: The head injury scenario. In: Health and Life Sciences workshop at the WWW2008. (2008)
64. Orchard, S., Hermjakob, H., Apweiler, R.: The proteomics standards initiative. Proteomics 3(7) (2003) 1374–1376
65. Taylor, C., Paton, N., Lilley, K., Binz, P., Julian, R.J., Jones, A., Zhu, W., Apweiler, R., Aebersold, R., Deutsch, E., Dunn, M., Heck, A., Leitner, A., Macht, M., Mann, M., Martens, L., Neubert, T., Patterson, S., Ping, P., Seymour, S., Souda, P., Tsugita, A., Vandekerckhove, J., Vondriska, T., Whitelegge, J., Wilkins, M., Xenarios, I., Yates, J.R., Hermjakob, H.: The minimum information about a proteomics experiment (miape). Nature Biotechnology 25(8) (2007) 887–93
66. Jones, A., Gibson, F.: An update on data standards for gel electrophoresis. Proteomics 7(Suppl 1) (2007) 35–40
67. Dagnino, A.: Coordination of hardware manufacturing and software development lifecycles for integrated systems development. In: IEEE International Conference on Systems, Man, and Cybernetics 3 (2001) 850–1855
68. Boehm, B.: A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes 11(4) (1986) 14–24
69. McDermid, J., Rook, P.: Software development process models. In: Software Engineer's Reference Book. CRC Press, Boca Raton, FL (1993) 15–28
70. Larman, C., Basili, R., V.: Iterative and incremental development: A brief history. Computer, IEEE Computer Society 36 (2003) 47–56
71. May, L, E., Zimmer, A, B.: The evolutionary development model for software. HP Journal (1996) Retrieved Jul. 12, 2010 http://www.hpl.hp.com/hpjournal/96aug/aug96a4.pdf
72. Fox, M.S.: The tove project: A common-sense model of the enterprise systems. In: Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. (1992)

73. Arpirez, J., Corcho, O., Fernadez-Lopez, M., Gomez-Perez, A.: Webode in a nutshell. AI Magazine 24(3) (2003) 37–47
74. Fellbaum, C.: WordNet, An Electronic Lexical Database. The MIT Press, Cambridge, MA (2000)
75. Knight, K., Luk, S.: Building a large-scale knowledge base for machine translation. In: Proceedings of the National Conference on Artificial Intelligence. Wiley, New York (1994) 773–773
76. Knight, K., Chander, I.: Automated postediting of documents. In: Proceedings of the 12th National Conference on Artificial Intelligence (vol. 1) Table of Contents, American Association for Artificial Intelligence Menlo Park, CA, USA (1994) 779–784
77. Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics 24(4) (1998) 599–612
78. Valente, A., Russ, T., MacGregor, R., Swartout, W.: Building and (Re) Using an Ontology of Air Campaign Planning. IEEE Intelligent Systems (1999) 27–36
79. Tempich, C., Pinto, H., Sure, Y., Vrandecic, D., Casellas, N., Casanovas, P.: Evaluating diligent ontology engineering in a legal case study. In: XXII World Congress of Philosophy of Law and Social Philosophy, IVR2005 Granada, May 24th, 29th (2005)
80. Garcia, A.: Developing Ontologies in the Biological Domain. PhD Thesis, University of Queensland (2007)