Designing for knowledge maturing: from knowledge-driven software to supporting the facilitation of knowledge development

Andreas P. Schmidt
Karlsruhe University of Applied Sciences
Moltkestr. 30
76133 Karlsruhe, Germany
+49 721 925-2914
andreas_peter.schmidt@hs-karlsruhe.de

Christine Kunzmann
Pontydysgu Ltd.
Ankerstr. 47
75203 Königsbach-Stein
+49 7232 4093309
kontakt@christine-kunzmann.de

ABSTRACT
Software engineering has been transformed in recent years by understanding the interaction with customers and the target context as an ongoing learning process. Responsiveness to change and user-centered design have been the consequences. In a similar way, knowledge and ontology engineering are undergoing fundamental changes to acknowledge the fact that they are part of a collective knowledge maturing process. We explore three examples: (i) social media based competence management in career guidance, (ii) ontology-centered reflection in multi-professional environments in palliative care, and (iii) aligning individual mindlines in pratice networks of General Practitioners. Based on these, we extract four levels of designing for knowledge maturing and associated technical implementations. This shows that future technology support should especially target facilitation of self-organized, but tool-mediated knowledge development processes, where, e.g., workplace learning analytics can play a prominent role.

Categories and Subject Descriptors
K4.3 [**Organizational Impacts**], I.2.4 [**Knowledge Representation Formalisms and Methods: Language Constructs and Features**]

General Terms
Design

Keywords
Knowledge maturing, knowledge management, knowledge engineering, design processes

1. INTRODUCTION
Current developments in software engineering are characterized by two major developments:

- *Make software engineering more responsive to change.* This has been achieved through replacing more rigid engineering processes such as the waterfall model by agile methodologies, ranging from pair programming to scrum or similar.
- *Making complexity of domains more manageable* by incorporating domain knowledge. This has been achieved by making software solutions more knowledge-driven. Prime example are the family of semantic technologies, which has proven valuable in a variety of scenarios, such as information and service integration, or context-aware recommendations.

While often considered separate, these two strands of development are largely intertwined as both of them refer to the same learning processes. The request for responsiveness to change originates from the very nature of design processes of software tools: it is a mutual learning process in which designing tools (and using them) deepens the understanding of the domain. This interdependency already creates the need of constant change. This is aggravated by the fact that domain knowledge itself accumulates and changes that make it necessary that software systems become a partner in this process. This is achieved by embedding more and more knowledge into tools themselves.

While both developments on their own have been investigated for far more than a decade, both in theory and practice, only limited research has gone into investigating the interdependency of both of them. In this paper, we want to take a second look at knowledge-based application and engineering strategies. We will use the knowledge maturing perspective to analyze the problem and solutions.

2. BACKGROUND
2.1 KNOWLEDGE ENGINEERING
Traditional knowledge engineering methods (for an overview see [25]) very much follow a waterfall model of software engineering, transferred to the domain of modelling domain knowledge. They focus on (few) domain experts. In the wake of emerging social media approaches, this has been criticized [7] and more agile methods have been proposed, such as [23], focusing on the principles from the agile manifesto (thus emphasizing the social nature of the engineering process) while other strands have focused on emergent semantics as outlined by [1] (thus emphasizing the automated processing).

There have been several attempts at “continuous knowledge engineering” which try to remove the wall between design time and runtime, such as [22], [3], [12], or [9]. These are mostly based on the wiki paradigm, which externalizes knowledge and makes it accessible for editing to a larger group of domain experts. Particularly [22] is pointing towards the necessity of considering the knowledge development process as embedded into a software engineering world that is moving towards continuous delivery.

Little investigation has gone into considering knowledge engineering as a collective learning process in which knowledge co-evolves as a result of the modelling process, i.e., modelling

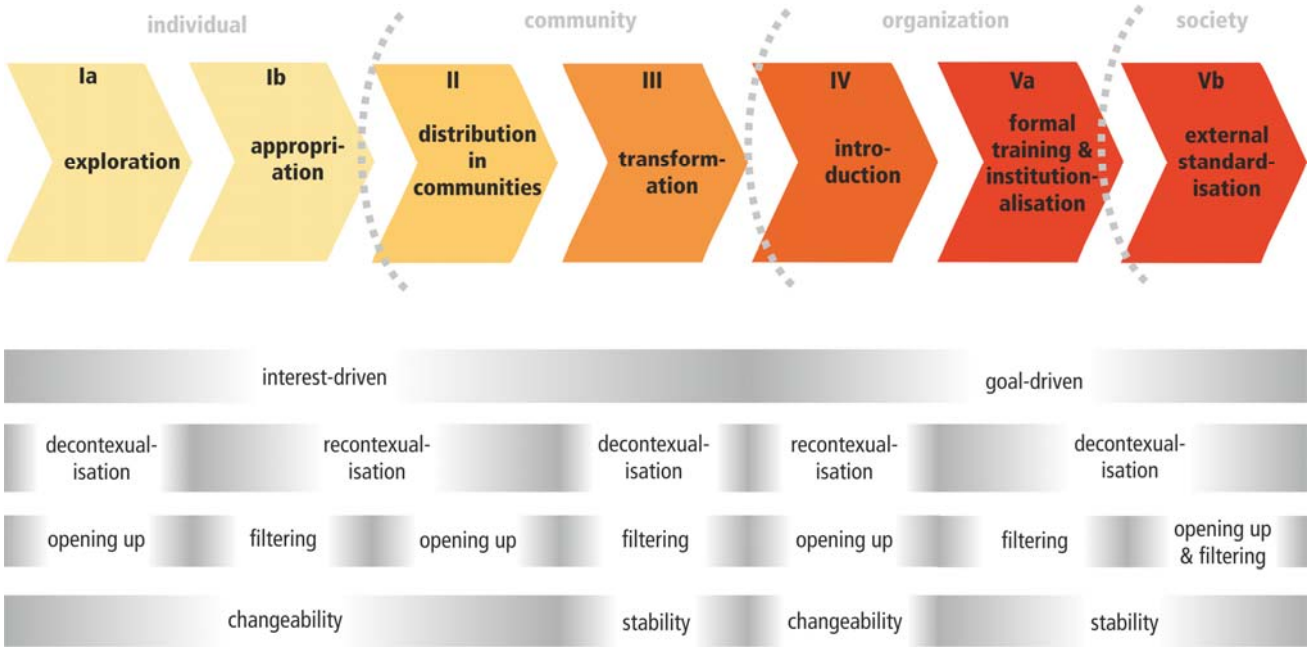

Figure 1: Knowledge maturing phases and characteristics [14]

does not just represent in a machine processable form knowledge that already exists, but the creating models influences the knowledge development process as such. The ontology maturing approach has incorporated this view [6] based on the knowledge maturing model that is going to be presented in the next section.

2.2 KNOWLEDGE MATURING

Knowledge maturing [15] looks at the processes of collective knowledge development processes and identifies across a variety cases that it can be classified into distinct phases. While not prescribing the process as a sequence of steps, it identifies phase with characteristic patterns that impact the way knowledge is dealt with, how learning takes place, and how to best represent the knowledge in artefacts. These phases range from emergence to external standardization and crosses the scopes of the the individual, the community, the organization, and the society where the primary conversation that drive the process take place.

The initial phases (**I. Emergence)** are characterized by the **exploration** (Ia) of new spaces, either as activities of analyzing existing material or by creative processes (new ideas). In both cases, knowledge is deeply subjective, and the individual decides through appropriation (Ib) where or not to pursue further development of the usually abundant items in phase Ia.

In the next phase (**II. distribution in communities),** where knowledge gets discussed and negotiated between different individuals of a social group. This includes the development of a shared vocabulary and associated understanding, and usually many individual contributions get amalgamated. To reach beyond the social group, **transformation (III)** is required where the focus is on creating artefacts by restructuring and agreeing on. Transformation means that knowledge is restructured and decontextualized to ease the transfer to collectives other than the originating community.

For further outreach, the **introduction phase (IV)** provides an initial step in which either knowledge is prepared in a way that it is easier to understand for others as part of workshops or trainings (instructional strand) or put to practice in a pilot (such as process knowledge). Both is experimental and is a learning phase where experiences are incorporated that prepare for a wider roll-out in the **institutionalization phase (Va)** where the knowledge gets a stable place, either as part of formal training plans, or as company-wide implementations (processes, products or similar). The goal is here to gain efficiency.

Finally, moving beyond the limited scope of companies, phase **Vb (External standardization)** moves towards standardisation or certification where comparability and compliance play a primary role.

Along these phases, key patterns can be observed, such as the alternation between the focus on changeability (Ia-II, IV) vs. stability (III, V), or the openness to impulses from outside (Ia, II, IV, Vb) vs. the filtering in phases Ib, III, Va.

2.3 APPLICATION TO DESIGN PROCESSES

This model can be applied to design processes. Let's suppose that the goal of application is to support specific domain activities, then the knowledge maturing phase model allows for describing and understanding the design process. The target users as well as the designers co-develop knowledge on how to support the usage scenario, which evolves from creating a shared understanding, to using models for transformation and prototyping.

But more important, the knowledge maturing model also emphasizes the difference between knowledge and the artefacts. All too often, artefacts are used that are not appropriate for the knowledge they are supposed to represent. Formalized ontologies are useless (or even counter-productive) if they try to formalize knowledge that is still in phase II, while formality can help to gain efficiency in phase V.

Even more, this perspective can help to understand that the design processes interact with learning processes in the domain itself where new knowledge is constantly being developed – with the use of the system. Knowledge that is relevant is always moving along the process, and just focusing on phase IV or V (which many engineering methodologies do) would deprive us from a significant part of what is relevant for design.

We want to investigate that in the following section more closely.

3. TYPOLOGY FOR KNOWLEDGE-BASED APPLICATIONS

To better understand the issues associated with applications that are knowledge-driven, a typology has proven useful that we have applied to a series of examples (some of them can be found in section 4).

Key idea is to take a – more or less strict - historic perspective on how state-of-the-art engineering methods for knowledge-based applications have evolved. This perspective focuses on three elements that seem to be key

a) *Design time vs. runtime.* A fundamental distinction in software engineering is between design time (design and development) and runtime (use).

b) *Roles for developing knowledge.* A second focus area is concerned with the question: who contributes to developing knowledge? Who evolves knowledge representations in the respective application?

c) *Processes for developing knowledge.* A third focus area is concerned with processes that characterize knowledge development that is linked to the respective applications.

3.1 HARDCODED KNOWLEDGE

There is hardly any software solution that does not include a considerable amount of (domain) knowledge. The naïve way of incorporating domain knowledge follows the traditional design process models where during the requirements phase, relevant pieces of knowledge are collected by business analysts, modelled in an appropriate way and passed on to developers who then based their design and implementation on this domain knowledge. The knowledge is implicit to many elements of their code, ranging from database schemas, via domain classes, business logic and control flows up to user interface design. All of it is incorporated at design time by the developers and designers themselves, and the processes for that are not separate from software engineering processes, such as the waterfall model.

There are two major weaknesses to this approach:

Responsiveness to change. While this might work in a world characterized by yearly or longer release cycles, this hits limits when software needs to deal with domain knowledge that evolves at a faster pace. Most domains, actually, have increased their pace so that even traditional domains where this would apply (such as finances) can no longer rely on this approach as changes are costly – although infrequent.

Knowledge ready at design time. Furthermore, from a knowledge maturing perspective, this approach is fundamentally flawed because it assumes that knowledge can be more or less "collected" from domain experts at a mature stage. But obviously, the knowledge which is best to support domain activities needs to be co-developed by those who want to do something with the designed artefact and those who know how to design – it is just not available at a sufficient level of maturity that allows for automation at the beginning of the design process. These interactions, however, can only happen at large intervals.

3.2 DESCRIPTIVE KNOWLEDGE REPRESENTATION

Agile methods in software engineering have addressed this problem by introducing iterations into design processes where after each iteration, software can be (potentially) used already. This alone does not help for knowledge-intensive domains. Because learning processes might lead to changing large portions of already existing code where knowledge might already be embedded.

To address that problem, explicit knowledge representation has been gaining popularity. Here, knowledge is represented explicitly through specific formalisms. This has a long history in computer science, but is now moving beyond its traditional application areas (such as expert systems and AI in general). RDF and ontology (such as OWL-DL) or rule formalisms (such as F-LOGIC) help to manage the complexity of domain knowledge by partially automating common processing patterns, introduce consistency checking etc.

In this area, there are two basic approaches to modelling that compete: the engineering approach (where humans create the models) and the mining approach (where algorithms create the models). Big data has clearly made an argument for the mining approach, but the knowledge maturing perspective also explains that this will not advance the development as such: only human sense making can achieve this. This argues in favor of using descriptive models (also as output of machine methods) that humans can understand and re-use for their learning processes.

Descriptive knowledge representation also has the advantage that transferring between domains becomes easier. There are often structural patterns where the same solutions can be applied not only in one, but in multiple domain, "just" the domain knowledge needs to be exchanged. Furthermore, from a learning perspective, the representation can more easily serve as a subject for reflection. This promotes the development of knowledge. This clearly links to the role of artefacts in the knowledge maturing model in phases III-V, but also shows that these approaches have a rather difficult time with earlier phases.

A major weakness of most descriptive knowledge representation approaches is that they are designed for (modelling) experts – typically for admin roles that change the representations.

3.3 PARTICIPATORY EVOLUTION OF KNOWLEDGE REPRESENTATIONS

While separating domain knowledge from other parts of software systems makes it more maintainable and thus easier to change by the software engineering, it still introduces a lag between users discovering the need for change and its actual change. This might introduce motivational barriers to give feedback and might hamper human negotiation processes – in terms of the knowledge maturing model: phase I and II interactions.

To address this issue, social media inspired approaches (which focus on phase II) have enabled users to change knowledge representations. Most popular has been the replacement of controlled-vocabulary-based approaches by free tagging (moving from taxonomies to folksonomies) [13], or wiki-based modelling

Type	Primary point in time for knowledge modelling	Primary roles for knowledge modelling	Processes for knowledge modelling	Implications for engineering
Hardcoded knowledge	design time	designer/developer	(software engineering)	-
Descriptive knowledge representation	design time / runtime	admin	hardcoded (for admin)	separation of knowledge and other components
Participatory evolution of knowledge representations	runtime	user	hardcoded (for users)	knowledge representation formalisms understandable for end users; support for user contributions
Self-Organized knowledge modeling processes	runtime	user	socially negotiated	support for activities instead of processes; negotiation spaces
Facilitated knowledge processes	runtime	user + facilitator	socially negotiated with facilitation support	support for facilitating roles and activities

Figure 2: Typology of knowledge-based applications

of domain knowledge [12] or even processes [9]. Knowledge modeling becomes a runtime activity.

This has considerable impact on knowledge representation itself, as we need to move from expert-based modelling to a wider range of people involved in the modelling process. Knowledge representation formalisms need to be understandable by non-experts. This is the why weak formalisms (in terms of expressiveness) have become more popular, such as concept maps or SKOS. Where the weakness of what "ordinary users" can deal with has not been acknowledged, it easily leads to conceptual problems, e.g., when building OWL-DL ontologies in Semantic MediaWiki systems – strict is-a semantics is hard for non-modelling experts to distinguish from part-whole relationships.

3.4 SELF-ORGANIZED KNOWLEDGE MODELLING PROCESSES

When moving towards more participatory approaches, there are still regulating processes for user contributions (e.g., editorial processes). The tool makes assumptions where new contributions can be added, and how they become part of a more authoritative set. Or roles and responsibilities are defined who is allowed and expected to check etc.

One important lesson from early test drives with social media inspired approaches is that it's not the features, but the way users appropriate those features (a useful concept in this context are affordances [14]). Tools are used differently than what they were designed for. Practices have a complex interrelationship with the tools they make use of – if tools get introduced, practices change, and changed practices lead to changed use of tools. This makes it difficult to prescribe certain processes that are meant to guarantee quality control.

Gardening [18] and the earlier Seeding – Evolutionary Growth – Reseeding model [10] provide an appropriate framework: (i) users themselves define the social processes, (ii) roles can be flexibly accepted without making any formal changes. The social group – not the technical system – defines the processes and rules. In many social media systems, this has worked remarkably well. The knowledge how to develop domain knowledge in a social group itself is subject to a maturing process – where we cannot formalize the result until it has been developed.

What does that mean for tool design? While in the wake of business process engineering, software has concentrated on supporting processes in the domain of the users, this view argues more towards a more modest approach: tools may support processes, but these processes are under constant negotiation so that the design should actually concentrate on the activities that constitute these processes. Instead of defining the steps how to include a new term in the ontology, the tool should concentrate on activities for adding, organizing, removing, merging etc. – and leave it to the users to negotiate the best way. This is in line with recent approaches to knowledge management, such as [15], focusing on activities rather than processes.

3.5 FACILITATED KNOWLEDGE PROCESSES

While the flexibility introduced by a new "modesty of tools", which does not prescribe structures nor processes and which is exemplified by the lightweight Web 2.0 tools, is more appropriate for the reality encountered, but it clearly asks too much from "ordinary users". Creating models is demanding, negotiating processes is even more demanding. They need support – from their peers. And tools can do a lot to facilitate this. Towards that end, design processes not only need to address the level of actual use, but also the level of facilitating the use of others. This is the current frontier: how to facilitate the learning processes in social systems. In [4], different approaches to facilitation have been conceptualized that can serve as a foundation. Among the manifold forms of facilitation, the concept distinguishes between human facilitation, facilitation through tool functionality, and facilitating environments – all of which can be supported by a respective tool design. Human facilitators can be provided with information where, when and how to intervene. Tools themselves can provide facilitating features, such as recommendations, triggers or similar. And tools can provide the basic infrastructure for discussion, negotiation, reflection etc.

Specific tool functionalities that have emerged include support for gardening activities (proactive recommendations such as [8]), but also various forms of analytics, including visualizations. Particularly at the workplace, these tools help the facilitators to identify areas for intervention into social processes, and reduce the effort to actually intervene.

3.6 SUMMARY OF TYPOLOGY

In the previous sections, we have come up with a typology for knowledge-based applications that is summarized in the table in Fig. 2, together with the implications on design. The different types can also be viewed as an evolutionary route with major

steps: (i) externalization of knowledge in applications for moving from *Hardcoded knowledge* to *Descriptive knowledge representation*, (ii) user-generated models for moving to *Participatory evolution of knowledge representations*, (iii) breaking down prescriptive process support into enabling activity support for moving to *Self-organized knowledge modeling processes*, and (iv) the inclusion of facilitator role into the system design for *Facilitated knowledge processes*.

After this rough sketch of types of knowledge-based application, we want to illustrate the observations along three examples in the next section.

4. ANALYZING THREE EXAMPLES

In the following, we investigate three examples of knowledge-based systems in different target areas to illustrate the typology introduced in the previous section. These have been selected from our prior research in the context of tools for knowledge maturing in different domains. The first example refers to a social-media approach to competence management based on a combination of people tagging and user-driven engineering of ontologies. The second example is about spiritual care support with an empirically derived ontology for spiritual care that is continuously refined. The third example is about a system based on the metaphor of "living documents" and represents the need for support in the complex pattern of stability and changeability.

4.1 PEOPLE TAGGING: A SOCIAL MEDIA APPROACH TO COMPETENCE MANAGEMENT

4.1.1 GENERAL OVERVIEW

Controlled vocabularies are at the heart of state-of-the-art competence management approaches [17], as competence catalogs that are used for competence profiles and requirements profile which can be matched to detect competence gaps, set up training plans, and staff teams [20]. These competence catalogues are descriptive knowledge representations and suffer from the problem that there developments lags behind the development of relevant skills that should be in the focus of competence management approaches.

One approach that has been inspired by Web 2.0 approaches has been the people tagging system SOBOLEO [5] which is based on the social semantic bookmarking paradigm (which is now used in several enterprise social media suites). Users can tag each other with topics, but these topics can also be organized in a taxonomies where typical problems of free tagging can be solved, such as synonym, multilinguality, or typos. It therefore includes a real-time collaborative editor that allows users to edit the SKOS taxonomy, and to move new tags to the appropriate position, and discuss with other users about it.

4.1.2 ANALYSIS

If we have a look along the stages from the previous section:

- SOBOLEO is based on a explicit, descriptive model of the domain knowledge – the competence catalogue.
- The catalogue is not fixed, but users can at any point in time contribute to the catalogue.
- User roles and restrictions are kept to a minimum. Anyone can take over gardening responsibilities, and evaluations have shown that some users are happy to do so without being formally assigned. SOBOLEO does not prescribe any process, but just provides the features for moving, creating synonyms etc. – it does not assume a particular order.
- As a first step towards facilitation, it provides proactive gardening recommendations (suggesting possibly synonymous concepts, or broader-narrower relationships) and analytics to see topics that are more frequently used than others. These should be in the focus for any gardening activities as they seem to be most relevant to create stability.

4.2 SPIRONTO: ONTOLOGY-BASED ENHANCED SPIRTUAL CARE

4.2.1 GENERAL OVERVIEW

Multi-professional domains are particularly knowledge-intensive as (i) multiple domains meet, and (ii) development of knowledge at domain boundaries is particularly dynamic. One example we have investigated is the case of SpirOnto [16]. It addresses a particular area of medical care: palliative care where physicians, nursery care, and spiritual care need to work together. Particularly spiritual care is often neglected as it is perceived not to follow a system approach (as the other disciplines do). To overcome this, Stiehl [24] has developed an initial spiritual care ontology based on existing patient records and documentation practices. This serves three purposes:

- Promote understanding in a multi-disciplinary setting and provide a boundary object in regular joint reflection sessions.
- Improve patient documentation to better reflect spiritual care to be able to more systematically provide spiritual care.
- Provide a conceptual framework for increasing the body of evidence for spiritual care and its effectiveness.

Towards that end, SpirOnto has been built, a tool that enhances patient records by annotating patient record entries with concepts from a shared spiritual care ontology.

4.2.2 ANALYSIS

If we analyze the example more closely, we discover the following:

- Domain knowledge has been made explicit through a manually created ontology (based on empirical finding). Instead of hard-coding categories that are relevant to design, these are externalized, and the representation is descriptive and human-understandable. Towards that end, concept maps have been used that are represented as RDF graphs. Only a core has been formalized as an OWL ontology. This core is needed to dynamically create the user interface based on the current status of the ontology. This include fundamental concepts, such as "observation", "spiritual concept", and "intervention".
- The ontology is not fixed at design time, but users can develop it further. While a stable core is fixed (and the software design relies on it), new entries can be added on the fly on two occasions: (i) when adding new entries, new categories can be added, and (ii) when reflecting on patterns between cases, new relationships

and intermediate concept can be added. Both can reflect a progressed understanding.

The possibility for self-organized social processes and facilitation support are currently not included, but under development. Particularly facilitation support seems to be promising by providing visual analysis that allows for overlaying several similar cases to suggest and discover patterns that might create new insights that could be even scientifically evaluated as a separate activity.

4.3 LIVING DOCUMENTS: DEVELOPING OPINIONS INTO COLLECTIVE KNOWLEDGE

4.3.1 GENERAL OVERVIEW

The third example moves away from vocabulary-centric knowledge development to more practice-oriented knowledge and illustrates the complexity of changeability vs. stability for tool support. The Living Documents system [2], which has been developed as part of the Learning Layers project, has emerged from observations in practices of General Practitioners. These GPs receive national guidelines, which have been developed through meta-analysis of scientific studies, but they equally need to incorporate their experiences and peer opinions, which form their mindlines [11]. Social processes to negotiate this knowledge is hampered by barriers that are related to changeability (opinions), stability (internal rules in practices, or national guidelines).

To overcome this, the LivingDocuments system has created an environment in which shared knowledge representations (such as local implementation plans) can be developed as living documents where (i) parts can be declared stable, (ii) comments and document parts can be associated with maturity indicators, and (iii) practice members can be notified about prescriptive changes.

4.3.2 ANALYSIS

From a knowledge maturing perspective, this tool has taken even one step back by not specifying precisely the formalism in which the knowledge is represented, but rather concentrated on key facilitation aspects: how to allow for changeability and stability within the same system without knowing or prescribing how the negotiation process takes place. Key aspects that have been found: (i) indicating maturity for all contributions, and (ii) creating awareness about changes at different levels of engagement.

5. CONCLUSIONS

In this paper, we have provided a birds-eye view on the role of knowledge and learning in engineering of knowledge-based applications, which constitute the majority of today's applications. Towards that end, knowledge maturing has provided a useful framework for analysis.

In this analysis, we could observe four principles for modern knowledge-based systems and associated engineering processes:

- *Do not hardcode knowledge into designs - make software knowledge-driven*. Knowledge becomes a "configuration" for applications, and the algorithms and interfaces explicitly depend on it instead of implicitly implementing it.
- *Tear down the wall between design time and runtime - knowledge models can be changed by users.* This acknowledges the fact that the appropriation of tools in users' practice is a learning process which co-evolves with knowledge about a domain. Experiencing competence in this respect is a key motivating driver [21].
- *Let users define their social processes for developing knowledge models - support activities, not processes.* Processes themselves are knowledge that evolve constantly. Applications prescribing certain ways of evolving knowledge representations limit social learning processes and might force users into something that is based on wrong assumptions. Also, experiencing autonomy is a second major factor for intrinsic motivation [21].
- *Support facilitators in this process through analytics: support guidance activities.* In the future, design of tools should concentrate on how to support users in supporting others in various activities of their learning process. This acknowledges that with the additional autonomy of user-generated models and negotiated processes an additional complexity is created. Tools are no longer fixed process support tools, but toolboxes which need to be appropriated in the same way as social processes around the tools might need to changed.

For an overall design process for knowledge-based systems design-based research process models as a starting point, such as [19] which put a shared conceptual model at the center which represents the knowledge in focus. Furthermore, design-based research has bridged theory-building, design, and evaluation activities without prescribing more than a high-level process framework. Finally, design-based research puts one important lesson learnt into the center of attention: that designing and engineering solutions is a socio-technical activity.

6. ACKNOWLEDGMENTS

This work has been conducted within the research projects EmployID (http://employid.eu) and Learning Layers (http://learning-layers.eu), which have received funding from the European Commission under the 7th Framework Programme, contracts no. 619619 and no. 318209.

7. REFERENCES

1. Aberer, K., Cudré-Mauroux, P., Ouksel, A.M., et al. Emergent semantics principles and issues. *Database Systems for Advanced Applications*, (2004), 25–38.

2. Bachl, M., Zaki, D., and Schmidt, Andreas Kunzmann, C. Living Documents as a Collaboration and Knowledge Maturing Platform. *International Conference on Knowledge Management (I-KNOW 2014)*, ACM (2014).

3. Baumeister, J., Reutelshoefer, J., and Puppe, F. Continuous Knowledge Engineering with {Semantic Wikis}. *CMS'09: Proceedings of 7th Conference on Computer Methods and Systems (Knowledge Engineering and Intelligent Systems)*, Oprogramowanie Naukowo-Techniczne (2009), 163–168.

4. Bimrose, J., Brown, A., Holocher-Ertl, T., et al. Introducing learning innovation in public employment

services. What role can facilitation play? *International Conference on E-Learning at the Workplace*, (2014).

5. Braun, S., Kunzmann, C., and Schmidt, A. Semantic People Tagging & Ontology Maturing: An Enterprise Social Media Approach to Competence Management. *International Journal on Knowledge and Learning (IJKL) 8*, 1/2 (2012), 86–111.

6. Braun, S., Schmidt, A., Walter, A., Nagypal, G., and Zacharias, V. Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering. (2007).

7. Braun, S., Schmidt, A., and Zacharias, V. Ontology Maturing with Lightweight Collaborative Ontology Editing Tools. GITO-Verlag (2007).

8. Braun, S. Community-driven & Work-integrated Creation, Use and Evolution of Ontological Knowledge Structures. 2012. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000025701.

9. Dengler, F. and Happel, H.-J. Collaborative Modeling with Semantic MediaWiki. *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, ACM (2010), 23:1–23:2.

10. Fischer, G., McCall, R., Ostwald, J., Reeves, B., and Shipman, F. Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments. 2001. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.2330.

11. Gabbay, J. Evidence based guidelines or collectively constructed “mindlines?” Ethnographic study of knowledge management in primary care. *British Medical Journal 329*, (2004).

12. Ghidini, C., Kump, B., Lindstaedt, S., et al. MoKi: The Enterprise Modelling Wiki. *The Semantic Web: Research and Applications. 6th European Semantic Web Conference, ESWC 2009 Heraklion, Crete, Greece, May 31–June 4, 2009 Proceedings*, Springer (2009), 831–835.

13. Golder, S. and Huberman, B.A. The Structure of Collaborative Tagging Systems. *Journal of Information Sciences 32*, (2006), 198–208.

14. Hutchby, I. Technology, Texts, and Affordances. *Sociology 35*, 2 (2001), 441–456.

15. Kaschig, A., Maier, R., Sandow, A., et al. Organizational Learning from the Perspective of Knowledge Maturing Activities. *IEEE Transactions on Learning Technologies 6*, 2 (2013), 158–176.

16. Kunzmann, C., Roser, T., Schmidt, A., and Stiehl, T. SpirOnto: Semantically Enhanced Patient Records for Reflective Learning on Spiritual Care in Palliative Care. *3rd Workshop on Awareness and Reflection in Technology-Enhanced Learning, co-located with ECTEL 2013*, (2013).

17. Kunzmann, C. and Schmidt, A. Ontology-based Competence Management for Healthcare Training Planning - A Case Study. (2006).

18. Peters, I. and Weller, K. Tag Gardening for Folksonomy Enrichment and Maintenance. *Webology 5*, 3 (2008).

19. Ravenscroft, A., Schmidt, A., Cook, J., and Bradley, C. Designing social media for informal learning and knowledge maturing in the digital workplace. *Journal of Computer Assisted Learning 28*, 3 (2012), 235–249.

20. Reinhardt, K. and Biesalski, E. Beyond skill management. Potentials and limitations of skill catalogues. *2006 IRMA INTERNATIONAL CONFERENCE. Theme: Emerging Trends and Challenges in Information Technology Management*, (2006).

21. Ryan, R.M. and Deci, E.L. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist 55*, (2000), 68–78.

22. Schilstra, K. and Spronck, P. Towards Continuous Knowledge Engineering. *Applications and Innovations in Intelligent Systems VIII*, (2001), 49–62.

23. Sören Auer. Agile Knowledge Engineering – Methodology, Concepts and Algorithms. 2007.

24. Stiehl, T., Führer, M., Roser, T., Kunzmann, C., and Schmidt, A. Describing spiritual care within pediatric palliative care. An ontology-based method for qualitative research. *12th Congress of the European Association for Palliative Care 2011, Portugal*, (2011).

25. Studer, R., Benjamins, V.R., and Fensel, D. Knowledge Engineering: Principles and Methods. *IEEE Transactions on Data and Knowledge Engineering 25*, (1998), 161–197.