


Graz University of Technology

Knowledge Management Institute

Know-Center

Master Thesis

Usage-based Calculation of Knowledge Maturing Indicators for Digital Resources in a Personal and Organizational Learning and Maturing Environment

Dieter Theiler, BSc

Graz, Austria, March, 2012

Advised by Univ.-Prof. Dr. Stefanie Lindstaedt

Abstract

Research in the field of ‘Knowledge Maturing’ focuses on improving knowledge construction processes within organizations. As guidance of knowledge workers plays an important role when it comes to developing knowledge, tools are sought to foster ‘Knowledge Maturing’-related activities. In this master thesis, ‘Knowledge Maturing Indicators’ get calculated upon traces of users’ interactions with software supporting ‘Knowledge Maturing’ and learning. ‘Knowledge Maturing Services’ get developed and integrated in respective ‘Knowledge Maturing and Learning Environment’ so that knowledge workers receive additional support in selecting content with regard to maturity.

This work contains the adaption and extension of a software environment allowing knowledge workers to collaboratively develop knowledge so that ‘Knowledge Maturing Indicators’ can be calculated automatically within respective environment. Building on continuative concepts from within the realm of ‘Knowledge Maturing’, logging of users’ interactions with the software application gets integrated in the client side of the software and a library for handling improved communication between the client and server side of the software necessary for transferring logged ‘User Events’ gets implemented. Further, ‘Knowledge Maturing Services’ get implemented in order to first, receive and store respective ‘User Events’ within the server side of the ‘Knowledge Maturing and Learning Environment’ and second, frequency-based calculate a chosen set of ‘Knowledge Maturing Indicators’ using traced ‘User Events’. To allow for the calculation of ‘Knowledge Maturing Indicators’, first, ‘User Events’ get assigned to groups representing certain semantics of to calculate ‘Knowledge Maturing Indicators’ and second, ‘Intermediate Indicators’ get introduced so that calculating ‘Knowledge Maturing Indicators’ upon usage-based information of digital resources within the maturing environment gets feasible. These ‘Intermediate Indicators’ in turn get used to extend current search processes within the software allowing for better support of chosen ‘Knowledge Maturing Activities’ carried out by users of respective software system. Concluding evaluations get done within this work in order to show the feasibility of followed approach for calculating ‘Knowledge Maturing Indicators’ in a ‘Personal and Organizational Learning and Maturing Environment’ and to elicit that established ‘Intermediate Indicators’ help first, valuing digital resources by users with regard to maturity and second, making searching more targeted at more mature digital resources.

Forschung auf dem Gebiet der Wissensreifung hat die Verbesserung des Wissensentwicklungsprozesses innerhalb von Organisationen zum Ziel. In diesem Prozess spielt die Unterstützung von Wissensarbeitern eine entscheidende Rolle, um Tätigkeiten bezogen auf die Reifung von Wissen zu fördern. In dieser Masterarbeit werden sogenannte Wissensreifeindikatoren mit Hilfe von benutzerbezogenen Nutzungsdaten aus entsprechender Wissensreifungs- und Lernumgebung berechnet. Es werden Wissensreifungsservices entwickelt und in jene Software – die Wissensreifung und Lernen unterstützt – integriert, so dass Wissensarbeiter in der Auswahl von reifen Inhalten zusätzliche Unterstützung erfahren.

Diese Arbeit enthält die Anpassung und Integration einer Softwareumgebung – die es Wissensarbeitern erlaubt gemeinsam Wissen aufzubauen –, um entsprechende Wissensreifeindikatoren automatisch zu berechnen. Aufbauend auf weiterführenden Konzepten aus dem Bereich der Wissensreifung, wird das Loggen der Benutzerinteraktionen mit der Softwareanwendung auf der Seite des Clients integriert. Zudem wird eine Bibliothek implementiert, die die Kommunikation zwischen der Client- und der Serverseite der Anwendung verbessert, um sie so für die Übertragung der gesammelten Benutzerinterkationen einzusetzen. Weiter werden Wissensreifungsservices entwickelt, um einerseits jene Benutzerinteraktionen in der Wissensreifungs- und Lernumgebung serverseitig zu empfangen und zu speichern und andererseits mittels jener Informationen automatisch ein ausgewähltes Set von Wissensreifeindikatoren zu berechnen. Zur Berechnung der Indikatoren werden einerseits die Instanzen jener Benutzerinterkationen zu Gruppen, welche die spezifische Semantik gewählter Wissensreifeindikatoren widerspiegeln, zugeteilt und andererseits sogenannte Zwischenindikatoren eingeführt, mit deren Hilfe die frequenzbasierte Berechnung der Wissensreifeindikatoren auf ressourcenbezogenen Nutzungsdaten erfolgt. Überdies werden jene Zwischenindikatoren für die Erweiterung der Suchprozesse innerhalb der Softwareanwendung verwendet, um so die Umsetzung ausgewählter Wissensreifungsaktivitäten durch die Benutzer der Software zu verbessern. Abschließende Evaluierungen aus dieser Arbeit bestätigen einerseits den gewählten Ansatz für die Berechnung der Wissensreifeindikatoren innerhalb der individualen und organisationalen Lern- und Wissensreifungsumgebung und zeigen andererseits, dass eingeführte Zwischenindikatoren helfen, erstens digitale Ressourcen durch Benutzer hinsichtlich deren Reife zu bewerten und zweitens Suchprozesse auf reifere digitale Ressourcen auszurichten.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

24.05.2012

…… ……..

date (signature)

Acknowledgement

At this point I want to thank all persons and institutions actively supporting me during my years of study, especially Julia, my parents and my friends.

Content

List of Figures

List of Tables

Acronyms

AMD	Attention Metadata
ARSS	Aggregated Resource Search Service
CK	Connextions Kent
DS1	Demonstrator 1
II	Intermediate Indicator(s)
KMA	Knowledge Maturing Activity(ies)
KMGA	Knowledge Maturing Guidance Activity(ies)
KMI	Knowledge Maturing Indicator(s)
KMS	Knowledge Maturing Service(s)
MC	Mature Components
POLME	Personal and Organizational Learning and Maturing Environment
RM	Resource Model
RMKMS	Resource Modeling Knowledge Maturing Service
RP	Resource Profile(s)
SA	Structuralia
TI	Transition Indicator(s)
TS	Triplestore
UE	User Event
ULKMS	Usage Logging Knowledge Maturing Service
UM	User Model
VS	Virtuoso Server
WS	Webservice

1 Introduction

The share of knowledge work [Blackler, 1995] has risen continuously during recent decades [Wolff, 2005]. It became clear that knowledge work has to be enhanced in order to give workers the possibility to update and expand their knowledge more rapidly [Schmidt, 2005a]. Instruments supporting knowledge workers have to be designed independently from where knowledge work gets applied [Kaschig et al., 2010] as traditional development methods are not longer capable of dealing with those demands [Schmidt, 2005a]. Moreover, programs to support training into the job, on the job, near the job, of the job and out of the job are necessary to develop knowledge work [Scholz, 2000], to avoid a high degree of individual learning paths [Schmidt, 2007] and to hinder the adoption of inert knowledge [Bereiter and Scardamalia, 1985]. Whether a certain form of learning is appropriate in a given situation depends on the type of knowledge, the individual and the situation itself [Barnes et al., 2009a].

Both e-learning and knowledge management try to improve the construction, preservation, integration, transfer and (re-) use of knowledge [Maier and Schmidt, 2007]. Schmidt elaborates on the difference between e-learning and knowledge management [Schmidt, 2005a]. E-learning deals with current information. It focuses on how to support the individual's learning process through pedagogical guidance (e.g. by a tutor organizing the learning process) and how individuals become able to construct new knowledge with some help. Knowledge management mainly deals with communication technology and focuses on the organizational perspective of how knowledge is constructed and maintained. In contrast to e-learning, knowledge management focuses more on how knowledge can be shared and transferred and models sequences of knowledge activities. Moreover, knowledge management does not strive to investigate how learning takes place but elaborates on what and why.

Employees must develop their competencies mutually dependently to sustain the flexibility of organizations as critical success factor for competitiveness [Rehaeuser and Hrcmar, 1996]. Individual learning is considered as the process of augmenting and changing knowledge bound to individuals' minds and their structures [Barnes et al., 2009a]. One special form of learning is seen in the alignment to the understanding of others of the same real-world phenomenon [Barnes et al., 2009a]. In an organizational context this requires extending the individual by an organizational perspective [Maier and Schmidt, 2007]. In fact, socio-cultural

theories of knowledge acquisition stress the importance of collaborative learning and learning communities [Hung, 2002]. Moreover, the group is seen as the most important building block when it comes to processing information [Wenger, 1987]. From that, collective knowledge gets aggregated by individual pieces of knowledge so that it becomes existent and shared within a larger group [Barnes et al., 2009a].

As e-learning and knowledge management raise boundaries concerning the organizational learning process, research in the field of 'Knowledge Maturing' tries to address and to tear down those barriers [Schmidt, 2005a] so that bounds in performance support get crossed [Schmidt, 2005b].

Building on former research in the field of 'Work-integrated Learning' [Lindstaedt and Farmer, 2004] [Lindstaedt and Ulbrich, 2006] 'Knowledge Maturing' strives for a new form of organizational guidance originating in community-driven approaches yielding continuous social learning in knowledge networks [Barnes et al., 2009a]. In turn, this leverages the intrinsic motivation of employees to engage in collaborative learning activities [Ravenscroft, Schmidt and Cook, 2010]. 'Knowledge Maturing' aims at advancing the collective level of knowledge following that learning has to be realized as active construction process at the collective level represented by organizations, groups or even smaller teams [Barnes et al., 2009a]. Knowledge not only has to be managed but has to be matured in order to master upcoming business changes [Riss et al., 2009].

Research in the field of 'Knowledge Maturing' is done with the focus on analyzing an organization's knowledge level by looking at the maturity of business processes as well as at knowledge development and communication processes. From that, the need evolved to reflect an organization's knowledge level by examining and assessing the organization's knowledge and learning practices.

In order to support 'Knowledge Maturing', it is necessary to describe knowledge worker's activities with regard to 'Knowledge Maturing' as well as to establish indicators reflecting results of such activities from a more global perspective [Barnes et al., 2009a]. 'Knowledge Maturing Indicators' got established helping to evaluate an organization's knowledge maturity status with regard to daily work practices. As the knowledge produced within an organization can manifest in digital artifacts [Nelkner, Magenheim and Reinhardt, 2009],

‘Knowledge Maturing Indicators’ can provide a means to judge (the content within) digital artifacts with regard to maturity at certain points in time.

1.1 Aim of Work

The aim of this work is to develop a usage-based approach automatically calculating ‘Knowledge Maturing Indicators’ for digital artifacts in a certain ‘Knowledge Maturing and Learning Environment’. In other words, the goal of this work is to automatically aggregate (knowledge worker’s) traces of ‘Knowledge Maturing Activities’ upon digital artifacts and in turn using them for the calculation of a subset of defined ‘Knowledge Maturing Indicators’. Calculated ‘Knowledge Maturing Indicators’ shall be presented to users of respective ‘Knowledge Maturing and Learning Environment’ supporting selecting digital artifacts with the help of ‘Knowledge Maturing Indicators’. In turn, the possibility to use ‘Knowledge Maturing Indicators’ directly within the users’ working context shall yield dedicated support for several ‘Knowledge Maturing Activities’. The work presented will provide, first, the implementation of a framework for storing defined traces of user interaction with respective software environment, second, software-based services using collected usage data for the calculation of ‘Knowledge Maturing Indicators’ for digital artifacts, third, an integration of calculated ‘Knowledge Maturing Indicators’ in available search processes within the ‘Knowledge Maturing and Learning Environment’ and fourth an elaboration on results within real world settings.

1.2 Structure of Work

This work is structured as follows: After motivating this work and presenting its goals, chapter 2 elaborates on the theoretical basis for this work in the field of ‘Knowledge Maturing’. Chapter 3 describes basic concepts used in this work beyond the concept of ‘Knowledge Maturing’. In chapter 4 this work’s developments for the calculation of ‘Knowledge Maturing Indicators’ get described before chapter 5 demonstrates the integration of calculated ‘Knowledge Maturing Indicators’ in search processes. Chapter 6 describes evaluations done upon results of this work. Chapter 7 and 8 conclude with results and give an outlook on possible further developments to be based on this work.

2 Knowledge Maturing

In organizational contexts ‘Knowledge Maturing’ describes the organizationally guided learning process interweaving informal learning processes of individuals [Schmidt et al., 2009] at group or community levels [Brown and Adler, 2008]. The generated knowledge allows knowledge workers to deal with complex and knowledge-intensive processes and adapt to unpredictable situations [Feldkamp, Hinkelmann and Thoenssen, 2007].

The term ‘mature’ as such expresses the meaning of complete, elaborated, fully aged, fully developed, having attained definite form, having reached a limit, perfected, ripe or saturated [Barnes et al., 2009a]. From this, maturing is the process of development or of becoming mature [Barnes et al., 2009a]. This process typically involves combining, aggregating and recombining different pieces of the maturing subject [Schmidt, 2005a].

In knowledge domains the maturing subject is the knowledge about a topic in a socially distributed activity system following that ‘Knowledge Maturing’ denotes changing and or increasing formality, distribution, commitment, legitimation, understandability and teachability of the maturing subject [Kohlegger, Maier and Thalmann, 2009].

From a capacities’ point of view, ‘Knowledge Maturing’ is related to the actualization and application of respective capacities (knowledge) [Riss et al., 2009]. From an informal perspective, ‘Knowledge Maturing’ is the fuzzy or goal-oriented development or evolution of (collective) knowledge [Barnes et al., 2009a]. ‘Knowledge Maturing’ from the learning perspective reflects the changing nature of knowledge that is passed on, learnt and taught [Schmidt, 2005a] [Schoefegger et al., 2009b] where learning is understood as social and collaborative activity [Schmidt et al., 2009].

Knowledge in its different forms gets continuously repackaged, enriched, shared, reconstructed, translated, integrated and etc. across different interlinked individual learning processes [Schmidt et al., 2009]. Through complex patterns of individual steps [Schmidt et al., 2009] knowledge becomes less contextualized, more explicitly linked and easier to communicate [Maier and Schmidt, 2007] [Attwell et al., 2008]. Altogether, this is resulting in the so called ‘Knowledge Flow’, a metaphor for interconnected individual learning processes [Maier and Schmidt, 2007].

The evolutionary process of ‘Knowledge Maturing’ transforms socially constructed knowledge objects in an organization [Kohlegger, Maier and Thalmann, 2009] from informal contextualized into formalized learning objects [Schoefegger, Seitlinger and Ley, 2009a]. Knowledge gets matured by generation, application and sharing of both informal and formal knowledge [Schmidt et al., 2009].

As the aim of ‘Knowledge Maturing’ and its general approach is now comprehensible, it is necessary to bring into light applicable concepts evolved from research in the field of ‘Knowledge Maturing’.

2.1 How does Knowledge Maturing take place?

2.1.1 Knowledge Maturing Process Model

The ‘Knowledge Maturing Process Model’ [Schmidt, 2005a] is dedicated to explore the continuity of knowledge within an organization. It sets out the basis for the elaboration of context interplays where maturing steps (see chapter ‘Knowledge Maturing Step’) happen.

Used in a descriptive way, the model shown in figure 1 explains changes observed in reality [Maier and Schmidt, 2007]. Used in a normative way, the maturity model supports owners, managers and persons realizing a guiding role in making changes in the maturity of maturing elements [Kohlegger, Maier and Thalmann, 2009]. It helps to identify (and then reduce) barriers for ‘Knowledge Maturing’ [Kohlegger, Maier and Thalmann, 2009] as well as to structure the analysis of existing organizational and technical infrastructures, information and communication technology solutions [Maier and Schmidt, 2007]. It supports goal-directed learning on a collective level [Kohlegger, Maier and Thalmann, 2009]. Moreover, it allows for analyzing the possibilities for learning support in a differentiated way [Schmidt, 2005a] and points out that learning takes place differently based on the maturity of knowledge to be constructed [Maier and Schmidt, 2007]. It describes how individual learning processes are interlinked within organizational contexts and different forms of knowledge [Schmidt et al., 2009].

Typically, the knowledge flow in an organization is characterized by inefficiencies and time and resource consuming friction [Schmidt, 2005a]. The model as shown in figure 1 tries to elicit discontinuities between certain phases of ‘Knowledge Maturing’ [Maier and Schmidt,

2007] and addresses competing concepts like knowledge transfer & sharing vs. training, informal vs. formal learning and explicit knowledge vs. learning resources [Schmidt, 2005a].

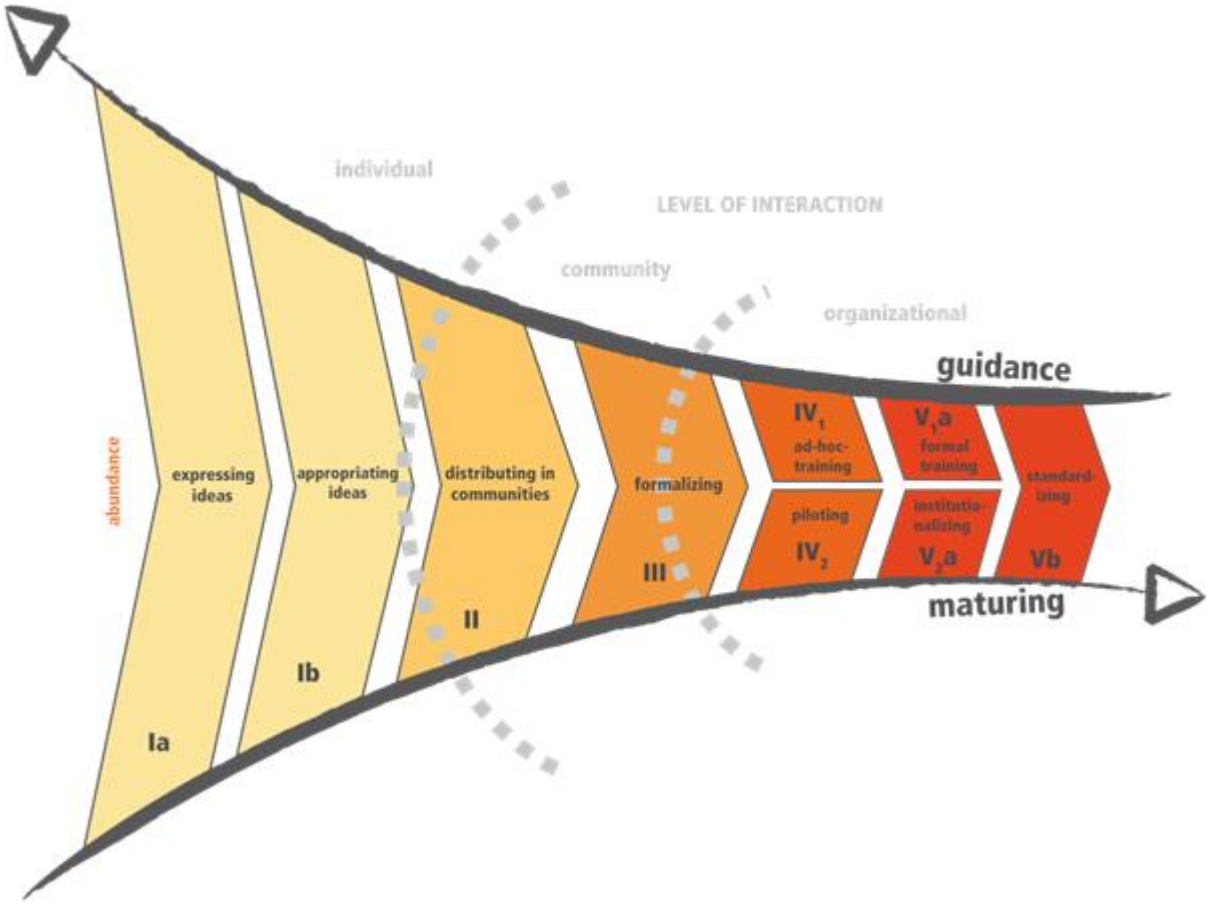

Figure 1: Knowledge Maturing Process Model [Barnes et al., 2011, p. 17]

2.1.1.1 Knowledge Maturing Process Model Phases

The phases of the 'Knowledge Maturing Process Model' describe increasing quantitative or qualitative capability changes of a maturing element so that the sequence of phases implemented in this model reflects certain aspects of reality and defines qualitative attributes to classify respective elements [Kohlegger, Maier and Thalmann, 2009]. The input for 'Knowledge Maturing' in a certain phase is the output from other phases and vice versa [Kohlegger, Maier and Thalmann, 2009].

The macro perspective on knowledge evolution (provided by the model) [Schmidt, 2005a] uses from each other independent stages as the upper stage cannot be traced back to the lower stage [Kohlegger, Maier and Thalmann, 2009]. The maturing subject (i.e. a person, object or a social system) matures implicitly between the stages, though there are or are no explicit triggers between the stages; nevertheless there can be explicit decisions to take a knowledge domain from one phase to the next one if all trigger conditions of the lower phases plus the ones of the actual phase are fulfilled [Kohlegger, Maier and Thalmann, 2009].

Often, knowledge is not able to develop until the latest phase but rather stays at earlier phases [Schmidt, 2005a]. Moreover, socialization or internalization of knowledge is very different in the various phases of the 'Knowledge Maturing Process' [Schmidt, 2005a]. To determine whether a maturing element finds itself in a certain phase, it gets drawn on four criteria, i.e. hardness, interconnectness / contextualization, legitimation and teachablility of and commitment to knowledge [Maier and Schmidt, 2007].

Emergence of Ideas

In this phase knowledge is created in highly informal discussions following a vague and to the originator restricted vocabulary [Maier and Schmidt, 2007] [Schmidt, 2007]. Certain creativity techniques like brainstorming or notices, sketches, to-do lists and etc. get applied in this phase in order to generate new ideas, exploit one's own creativity potential and foster learning [Schmidt, 2008].

The sub-phase Ia 'Expressing Ideas' in a more practical understanding is dedicated to searching for information, keeping aware and informed of certain changes, judging the relevance of certain content or generating initial answers as well as collecting initial information. Subsequently, sub-phase Ib 'Appropriating Ideas' of this knowledge evolution stage contains work on individual annotations and establishing of collections of material as well as finishing up and delivering developed content to certain audience.

Distribution in Communities

In this phase information and experience exchange are seen as main targets besides elaborating on new knowledge [Schmidt, 2008]. It gets strived for common terminology shared among community members [Maier and Schmidt, 2007] [Schmidt, 2007]. During 'Distribution in Communities' various learning techniques are applied, e.g. learning by imitation or corporative learning as well as discussions and collaborative working [Schmidt, 2008]. Other relevant activities in this phase include staying informed and being aware of important changes, identifying relevant people from within the community, aggregating and sharing of digital artifacts with colleagues.

Formalization

The focus at this stage lies on structuring inherently unstructured knowledge from the preceding two phases up to content with a certain purpose, e.g. project reports or design

documents [Maier and Schmidt, 2007] [Schmidt, 2007]. Problem solving decisions get supported by already existing knowledge and experienced practical knowledge gets passed on to be provided for others [Schmidt, 2008]. Besides the revision of content and search for information [Schmidt, 2008], knowledge artifacts created or established in this phase not necessarily reflect monolithic wholes but can consist of content chunks connected through a meaningful order and explicit underlying relationships [Schmidt, 2007].

Ad-hoc Training

Knowledge produced in preceding phases is not well suited as learning material as no didactical considerations were taken into account so far [Maier and Schmidt, 2007] [Schmidt, 2007]. Within this phase pedagogical aspects should be added to topics in order to enable broader dissemination [Maier and Schmidt, 2007] [Schmidt, 2007]. The target of this phase lies in the communication of deepened and actualized knowledge for advanced knowledge workers [Schmidt, 2008]. Learning then develops with the help of more sophisticated knowledge artifacts, learning objects, tutorials or short training courses [Schmidt, 2008].

Formal Training

Phase 'Formal Training' suggests establishing global from individual learning objects in order to cover a broader subject which makes them teachable to novices [Maier and Schmidt, 2007] [Schmidt, 2007]. Learning happens within lectures and with the help of tutorials, textbooks, and e-learning and certification processes [Schmidt, 2008].

2.1.2 Knowledge Maturing Step

From the fact that all learning processes occur in context, 'Knowledge Maturing' involves repeated steps of de-contextualization and re-contextualization of knowledge so that context aspects which remain unchanged provide the glue between those different steps [Schmidt, 2005a].

The maturing step in figure 2 denotes one step from a certain maturity level to another one [Schmidt, 2005a]. In organizational contexts certain maturing steps require a transformation from commitment by individuals to the legitimating by a formal organizational unit [Maier and Schmidt, 2007]. As fostering knowledge evolution yields the necessity to reduce barriers in knowledge flows and to develop cohesion among different phases of 'Knowledge Maturing' [Schmidt, 2005a], above described 'Knowledge Maturing Process Model' got set up describing the different phases of wherein or between knowledge matures.

The daily work of knowledge workers within the different phases of the ‘Knowledge Maturing Process’ leads to the evolution of knowledge. From that, dedicated activities got identified which foster ‘Knowledge Maturing’ within the ‘Knowledge Maturing Process’ described by the ‘Knowledge Maturing Process Model’.

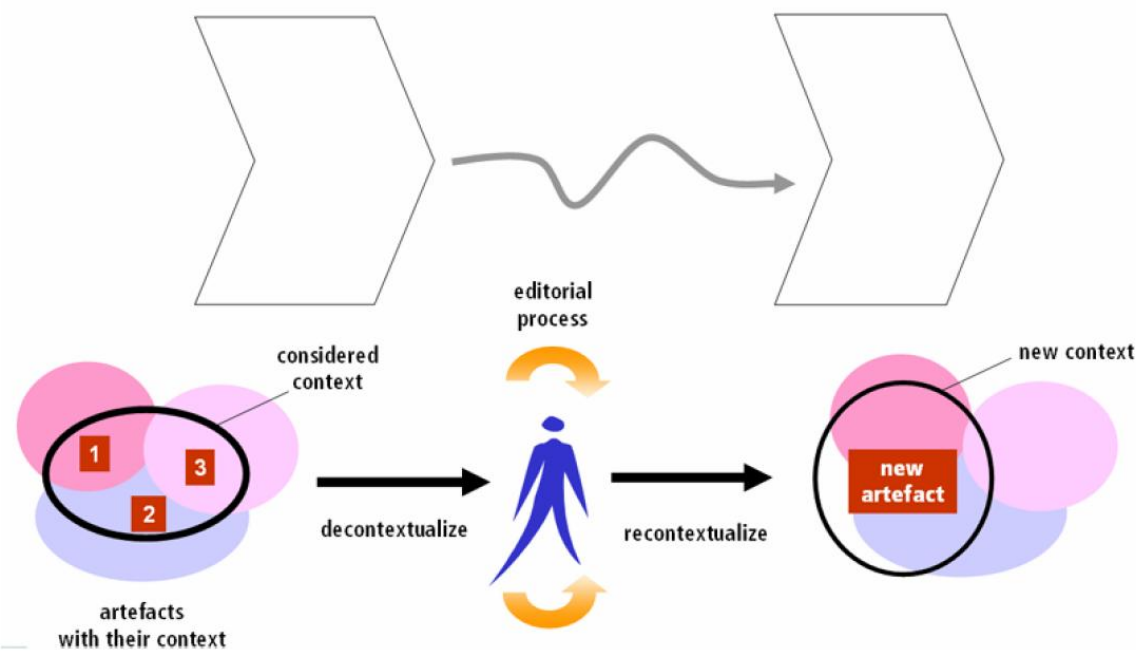

Figure 2: Knowledge Maturing Step [Schmidt, 2005a, p. 6]

2.1.3 Knowledge Maturing Activities

‘Knowledge Maturing Activities’ (KMA) typically contribute to ‘Knowledge Maturing’ or guide it [Barnes et al., 2011]. KMA in general can be defined as individual or group activities that contribute to the goal-oriented development of knowledge within an organization [Kaschig et al., 2010] [Mazarakis et al., 2011]. As such, they constitute the basis for systematic interventions in companies with the help of tools and IT services [Kaschig et al., 2010]. As KMA trigger learning processes which have to be embedded into, interwoven with and even indistinguishable from everyday work processes [Schmidt et al., 2009], KMA must take place in each phase of the ‘Knowledge Maturing Process’ [Kaschig et al., 2010].

Davenport, Jarvenpaa and Beers and Kelloway and Barling came up with knowledge work practices like acquiring, creating, gathering, organizing, packaging, maintaining, systemizing, communicating and applying knowledge [Davenport, Jarvenpaa and Beers, 1996] [Kelloway and Barling, 2000]. These got detailed by Schultze [Schultze, 2000] and subsequently combined with the results of a broad empirical ethnographically informed study [Kaschig et

al., 2010] to yield an extensive list of activities in knowledge work. These activities in turn were combined with research requirements from within 'Knowledge Maturing' [Barnes et al., 2010] and consolidated into a list of twelve KMA [Kaschig et al., 2010] as shown in table 1.

KMA provide basic insight into which activities in daily work practices are most relevant for letting knowledge mature. Therefore, it is important to see whether those activities get performed on a regular basis or get hindered by any circumstances.

Knowledge maturing activities	
Embed information at individual or organizational level	Find relevant digital resources
Keep up-to-date with organization-related knowledge	Familiarize oneself with new information
Re-organize information at individual or organizational level	Communicate with people
Reflect on and refine work practices or processes	Assess, verify and rate information
Find people with particular knowledge or expertise	Share and release digital resources
Restrict access and protect digital resources	Create and co-develop digital resources

Table 1: Knowledge Maturing Activities

2.1.3.1 Barriers for Knowledge Maturing Activities

In a study discussed by Mazarakis et al. [Mazarakis et al., 2011] in 139 interviews people were asked in particular about perceived barriers in 'Knowledge Maturing'. The result of the study yielded that the most prominent barriers for the application of KMA are lack of time, low awareness of the value of KMA and the lack of benefit and of usability. Additional comments related to the organizational culture subsuming the lack of individual autonomy, formalization and guidance, collaboration and personal interdependencies [Mazarakis et al., 2011].

2.1.3.2 Important, Less Supported and Not Successfully Performed Knowledge Maturing Activities

In the study discussed by Mazarakis et al. [Mazarakis et al., 2011], people were also asked to judge perceived importance, support and success of KMA in their respective organizations. The outcomes revealed that some KMA are very interesting concerning future research [Kaschig et al., 2010]. For example KMA 'familiarize oneself with new information', 'communicate with people' and 'find people with particular knowledge or expertise' got perceived most important in relation to other KMA. KMA 'embed information at individual or organizational level', 'keep up-to-date with organization-related knowledge' and 'reflect on

and refine work practices or processes' are of secondary importance. Moreover, the study showed that KMA 'find people with particular knowledge or expertise', 'reflect on and refine work practices or processes' and 'assess, verify and rate information' are currently least supported. It was also interesting to see that KMA 'find people with particular knowledge or expertise' and 'reflect on and refine work practices or processes' are performed least successfully in relation to their perceived importance.

It could be shown that KMA often are not performing as well as organization's success could be sustained. From that, additional activities could be identified aiming for improving the success of respective KMA.

2.1.4 Knowledge Maturing Guidance Activities

'Knowledge Maturing Guidance Activities' (KMGA) provide possible interventions in the 'Knowledge Maturing Process' in order to maintain the application and success of defined KMA [Barnes et al., 2011]. KMGA got derived from knowledge work activities stated by persons asked in an interview study described in Barnes et al. [Barnes et al., 2011]. Those findings got further analyzed and consolidated yielding the set of KMGA mostly containing KMA not taken into account by previously established (above shown) list of KMA [Barnes et al., 2011]. The list of resulting KMGA gets shown in table 2.

Knowledge maturing guidance activities	
Provide feedback	Structure & organize
Respond	Make aware
Recommend, suggest & advice	Encourage
Irritate & challenge	Evaluate & assess results
Coordinate	Create opportunities
Reward	Monitor activities & progress
Give legitimation	

Table 2: Knowledge Maturing Guidance Activities

KMGA allow for the design of guidance in the 'Knowledge Maturing Process' by natural or more or less artificial triggers for KMA.

In order to judge an organization's maturity status by the outcome of completed KMA and KMGA, 'Knowledge Maturing Indicators' got introduced indicating results of maturity developments.

2.1.5 Knowledge Maturing Indicators

'Knowledge Maturing Indicators' (KMI) have been introduced by Barnes et al. [Barnes et al., 2009a]. They provide a framework for the design of 'Knowledge Maturing' support in a specific target context as they provide a means to assess (changes in) the maturity of knowledge. In other words, as 'Knowledge Maturing' becomes not directly visible as such, KMI make maturing processes visible [Barnes et al., 2011]. The complete list of available KMI gets shown in the appendix.

KMI can be quality-related as well as activity-related [Barnes et al., 2011]. Whereas quality-related KMI refer to the quality of the maturing element like its formality, readability, experience, responsibility or membership, activity-related KMI are based on observing activities by individuals or groups representing the majority of observed KMI [Barnes et al., 2011].

In general, KMI can be stated as states or state changes with several variants [Barnes et al., 2011]. KMI in principle can refer to a single or number of event(s); can occur within a time frame, manifest in change in frequency or in comparison to previous time frames [Barnes et al., 2011]. Moreover, KMI can be aggregated or combined in order to yield a higher semantic expressivity.

When it comes to structuring KMI, different kinds of knowledge manifestations relevant for 'Knowledge Maturing' are of importance [Barnes et al., 2011].

2.1.5.1 Knowledge Manifestations

In the context of 'Knowledge Maturing' theory, knowledge is seen as cognitive structure which is bound to individuals' minds as well as an abstraction of individual knowledge in a collective [Barnes et al., 2009a]. From a different perspective, knowledge can be described by both subjective and objective matters, following that objective knowledge can be found in documents independent from the human interpreter [Nonaka and Peltokorpi, 2006]. Taking up this approach, subjective knowledge gets either represented by the concept of cognifacts referring to individual knowledge or sociofacts in turn referring to collective knowledge. It is

important to see that objective knowledge gets codified in artifacts [Nelkner, Magenheim and Reinhardt, 2009] [Barnes et al., 2009a].

Artifacts – as knowledge manifestations in electronic form – are able to reflect the maturity of knowledge even though they don't contain knowledge in its actual sense but provide a means to communicate knowledge [Barnes et al., 2009a]. Cognifacts, as knowledge in its narrower sense, include the know-how and knowing what which is bound to people's minds [Barnes et al., 2009b] [Barnes et al., 2009a]. Sociofacts represent knowledge being recognized on the basis of users' activities; hence not necessarily become manifested in any form [Riss et al., 2009] [Barnes et al., 2009b].

In order to assess and improve 'Knowledge Maturing' within an organization, different knowledge manifestations for which maturing takes place have to be considered [Braun and Schmidt, 2007] and hence reflected by the hierarchy of KMI.

2.1.5.2 Knowledge Maturing Indicator Hierarchy

To be able to structure KMI according to the form of knowledge being considered, a hierarchy of KMI got initially derived from an ethnographically-informed study [Barnes et al., 2009a]. Further, this hierarchy got tested for its feasibility in an interview study [Barnes et al., 2010] and received a major revision with regard to the categorization of KMI in order to yield a higher semantic expressivity [Barnes et al., 2011]. As a result, the following hierarchy of KMI can be shown:

- I. Artifacts
    - I.1 Artifact characteristics
    - I.2 Creation context and editing
        - I.2.1 Creator
        - I.2.2 Purpose
        - I.2.2 Creation process
    - I.3 Usage
    - I.4 Rating and legitimation
- II. Individual capabilities
    - II.1 Individual activities
    - II.2 Individual – organization
    - II.3 Individual – group

    - II.4 Rating, assessment
- III. Topic
    - III.1 Activities
- IV. Sociofacts
    - IV.1 Process/task (knowledge)
    - IV.2 Quality of social network
    - IV.3 Agreement
    - IV.4 Collective capability
- V. Impact and performance
    - V.1 Performance
    - V.2 Quality
    - V.3 Impact

KMI in above categories observe certain manifestations of knowledge and measure certain characteristics or qualities of respective manifestations [Barnes et al., 2011]. The possible category ‘cognifact’ which would represent knowledge of individuals got split into categories ‘individual capabilities’ and ‘topic’. On the one hand these categories are able to better reflect what persons are dealing with and are able to consider the persons themselves as maturing elements on other the hand. Category ‘impact and performance’ represents an output-oriented category that has been added additionally as suggested by a representative study [Barnes et al., 2011].

As KMI get used to judge the knowledge state of an organization upon different knowledge representations, KMI in turn have a close relationship to other concepts within the ‘Knowledge Maturing’ theory framework.

2.1.5.3 Knowledge Maturing Indicators and Knowledge Maturing Process Model

KMI operationalize the rather abstract characteristics of the phases of the ‘Knowledge Maturing Process Model’. More specifically, they indicate in which phase of the model a certain manifestation of knowledge is or which transitions have been mastered so far [Barnes et al., 2011]. To be able to apply KMI for certain elements maturing within the ‘Knowledge Maturing Process’, the focus of the ‘Knowledge Maturing Process Model’ applied in a concrete context has to be determined first.

2.1.5.4 Knowledge Maturing Indicators and Knowledge Maturing Activities

As there basically exist two forms of KMI, i.e. quality and activity-related KMI, the close relationship of activity-related KMI to KMA enables activity-related KMI to make KMA traceable [Barnes et al., 2011]. Some of the base forms of indicators, i.e. access, use, relate, search or discuss as shown in figure 3, refer to multiple KMA while other base forms are more specific and refer to one or two KMA only. Base forms of KMI which make more than one KMA traceable have to concretized with regard to the environment (e.g. software application) were they will be applied.

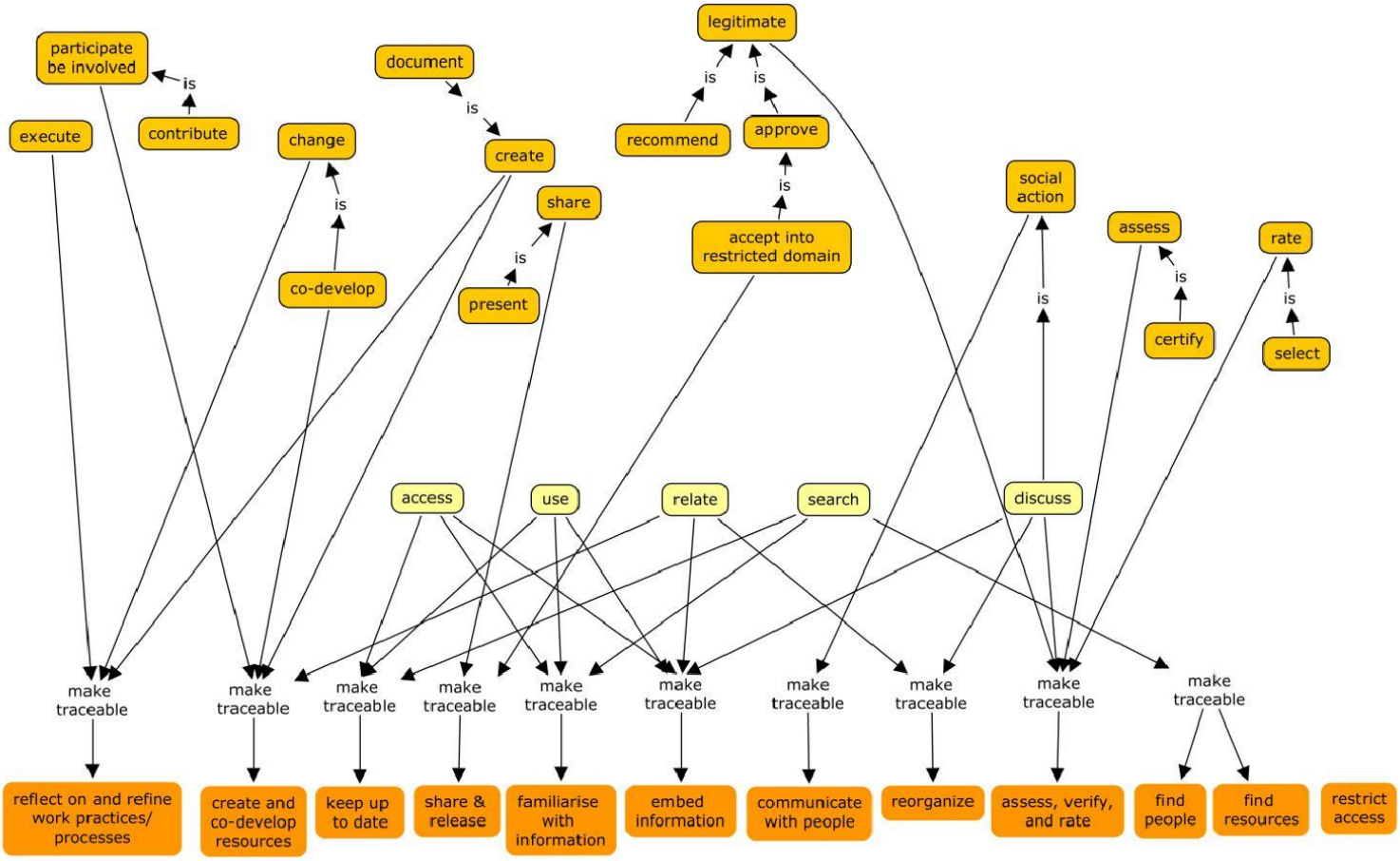

Figure 3: Base Forms of Activity-related Knowledge Maturing Indicators [Barnes et al., 2011], p. 96

2.1.5.5 Knowledge Maturing Indicators and Transition Indicators

The concept of 'Transition Indicators' (TI) constitutes the basis for measuring the current state of various knowledge types as well as the basis for tracing 'Knowledge Maturing Activities' upon artifacts at hand with the help tools [Nelkner et al., 2011]. From that, TI are very domain and tool specific and can be supported by concrete tool functionalities.

TI help to understand and evaluate KMI as not all KMI are measurable directly but by TI which are operable and detectable and can be related and mapped to KMI [Nelkner et al., 2011]. TI observe whether 'Knowledge Maturing' is indeed taking place assuming that the interaction with knowledge artifacts through tools results in 'Knowledge Maturing'. TI only cover automatically observable aspects of 'Knowledge Maturing', i.e. how workers interact with software and knowledge artifacts, ranging from activities such as clicking, viewing, opening and downloading to changing, rating and tagging [Nelkner et al., 2011].

2.2 Supporting Knowledge Maturing

To assist knowledge workers in KMA, certain tools get developed and integrated into workplaces so that knowledge workers, first, get supported in daily work with regard to 'Knowledge Maturing' and second, can receive KMGA provided by respective tool functionalities. 'Knowledge Maturing' support in daily work processes mainly is sought to be provided by certain kinds of software environments and services.

2.2.1 Learning and Maturing Environments

Software-based 'Personal and Organizational Learning and Maturing Environments' (POLME) are special types of 'Personal Learning Environments'. 'Personal Learning Environments' [Attwell, 2007] got proposed as work-integrated and personalized tools for communication, collaboration, structuring, reflection and awareness building [Schmidt et al., 2009]. POLME provide knowledge workers with a flexible tool kit letting them engage in the 'Knowledge Maturing Process' targeting organizational development [Schmidt, 2008]. To embed the 'Personal Learning Environment' paradigm into organizations, 'Organizational Learning Environments' [Barnes et al., 2009b] bring in the organizational perspective by a new form of organizational guidance [Schmidt et al., 2009] in order to foster innovation, learning and KMA as essential social processes [Kaschig et al., 2010]. As design studies [Ravenscroft, Schmidt and Cook, 2010] highlighted the problem of clearly differentiating between organizational on the one hand and personal learning and maturing environments on the other hand, tools supporting 'Knowledge Maturing' typically provide both functionalities supporting transitions from personal realms to community and functionalities supporting transitions from community to organizational contexts at the same time [Nelkner et al., 2011]. Hence, POLME try to overcome gaps in the 'Knowledge Maturing Process' support [Schmidt et al., 2009] [Schoefegger et al., 2009b].

From bringing together both organizational as well as personal approaches in learning and maturing support in the concept of POLME, the individual knowledge worker gets able to view her contributions in an organizational context [Schmidt et al., 2009]. First, this encourages the contribution of knowledge towards organizational goals and second, enhances the organization with the opportunity to analyze bottom-up activities in order to promote a consolidation of KMA activities towards organizational goals [Schmidt et al., 2009]. Enabling the creation of important communities as well as enriching existing knowledge artifacts so that they can be reused as learning objects [Schmidt et al., 2009] reseeds the innovation process within an organization [Attwell et al., 2008].

Motivational theories emphasize the importance of experiencing competence, autonomy and relatedness to the used system [Schmidt et al., 2009]. As a consequence, POLME get designed in the manner of social software wherein informal learning can happen by interest, learner driven and problem-based [Attwell et al., 2008]. It follows that contribution should be kept simple [Witschel et al., 2010] as well as users should not be forced to invest much effort into formalizing their contributions [Fischer et al., 1996]. The learning environment has to avoid being conceived operating in a top-down way [Schmidt et al., 2009] but rather recognize the individual as a consumer as well as a contributor and provide her with suitable forms of learning for each maturity level [Schmidt, 2007].

To be more concrete, POLME have to support learning by searching for and exploring artifacts for the task at hand, make content searchable, provide possibilities to communicate with people, make aware of changes by pushing information to the users, provide valuable metadata for the flow of changes concerning knowledge artifacts, support the creation, refinement, development, aggregation, structuring and sharing of artifacts, ease understanding of knowledge, provide a means to assess knowledge, provide standardized metrics to describe gathered knowledge, (pre-) process knowledge to reduce the workload for knowledge workers and support reflection and gardening [Bradley et al., 2010] [Schmidt et al., 2009] [Attwell et al., 2008]. To overcome the lack of support for emerging communities of interests in many virtual learning environments [Agostini et al., 2007], POLME leverage employees' creativity and hands-on experience and improve sharing of knowledge [Schmidt et al., 2009].

In order to bring POLME to life, a flexible infrastructure of reusable knowledge services has to be provided [Schmidt et al., 2009] including services operating on available information to

support ‘Knowledge Maturing’ tasks [Riss et al., 2009]. These services provide the basis for POLME (including Web 2.0 technologies) used for working, learning, reflection and collaboration [Attwell et al., 2008].

2.2.2 Knowledge Maturing Services

In the course of research done for supporting knowledge workers with concrete services, ‘Knowledge Maturing Services’ (KMS) have been suggested [Ley et al., 2009] as a means to support ‘Knowledge Maturing’ within a POLME based upon related work in the area of ‘Work-integrated Learning’ [Ley et al., 2008] [Lindstaedt et al., 2008]. KMS facilitate the ‘Knowledge Maturing Process’ from an organizational learning perspective [Kump et al., 2011] and guide users in collaboratively developing knowledge assets of organizational interest [Schoefegger et al., 2009b]. Additionally, these services make results of KMA visible and digestible for the individual [Schoefegger et al., 2009b]. KMS integrate support for the ‘Knowledge Maturing Process’ into knowledge work, thus bridge the separation along knowledge construction and sharing [Schmidt et al., 2009] [Schoefegger et al., 2009b]. KMS help along the way of the ‘Knowledge Maturing Process’ in order to bring knowledge over barriers between and within phases of the ‘Knowledge Maturing Process’.

KMS have been designed as loosely coupled arrangements of services building on a lightweight architecture [Blaževic et al., 2010] which are in line with the work environment context [Kaschig et al., 2010]. These services enable the knowledge worker to handle different knowledge assets, negotiate changes of knowledge (artifacts) and mediate new and powerful learning processes interlinked with the workers’ everyday digital behavior [Schmidt et al., 2009] [Schoefegger et al., 2009b] [Ravenscroft, Schmidt and Cook, 2010]. KMS in general are complex services composed of basic services [Schoefegger et al., 2009b] extended by maturing relevant functionalities for POLME [Schmidt et al., 2009]. Moreover, KMS make use of collective usage data [Jong, Specht and Koper, 2008] [Schmidt, 2005c] as well as user feedback to improve the system’s value and performance due to concepts described by Surowiecki [Surowiecki, 2005].

Knowledge workers, communities and organizations are able to use KMS iteratively during the overall process of content creation. KMS ease the complexity of underlying knowledge structures and their evolution within a POLME [Schoefegger et al., 2009b]. KMS interfaces for users help in improving, learning and discovering the knowledge system with its

knowledge entities and relationships; the services themselves mostly work in the background to analyze several kinds of knowledge and its use [Schoefegger et al., 2009b].

Various KMS have to be offered with regard to the current evolution state of a maturing element [Schmidt et al., 2009] following the description of complex software system evolutions [Fischer et al., 1996]. This yields the intervention of KMS in following different phases according to the SER model [Fischer et al., 1996] adapted for the 'Knowledge Maturing Process' [Schmidt et al., 2009]: 'Seeding' phase, wherein KMS are set up, initialized and applied to knowledge artifacts and structures; 'Evolutionary Growth' phase, wherein KMS help to create content and adapt the characteristics of knowledge; 'Reseeding' phase, wherein knowledge and collective KMA get analyzed and visualized, conceptualizations get negotiated and underlying structures get changed. KMS in the 'Reseeding' phase help to restructure the environment after the 'Seeding' phase provoked an evolutionary and undirected growth of the knowledge repository by community activity [Schmidt et al., 2009].

KMS combine rich representations of data employing statistical procedures as well as model-based approaches as they make use of emerging models, hence can be characterized as data rich and model driven (hybrid) services [Kump et al., 2011].

2.2.2.1 Knowledge Maturing Service Classification

KMS get structured according to their defined functionality from a more technical point of view [Kump et al., 2011] as shown in figure 4.

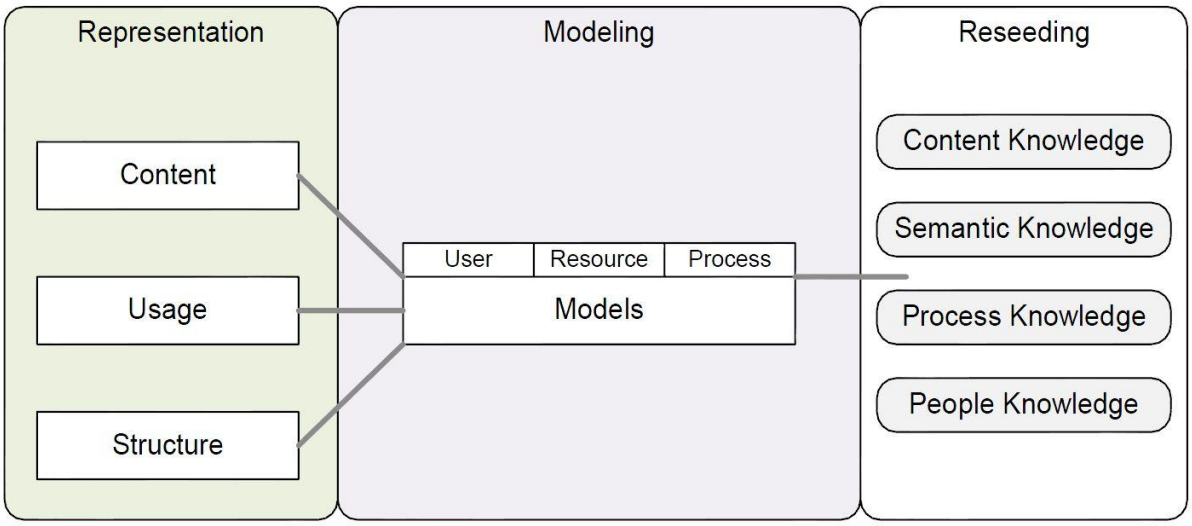

Figure 4: Knowledge Maturing Service Classification [Kump et al., 2011, p. 9]

2.2.2.1.1 Representation

KMS in this category maintain and analyze abstract representations of data within POLME mainly as they are working directly with certain knowledge representations. ‘Representation’ services get developed in order to store and analyze either content (natural language), structural (semantic or procedural) or usage-based knowledge representations [Schoefegger et al., 2009b] [Kump et al., 2011].

2.2.2.1.2 Modeling

‘Modeling’ services are responsible for automatically calculating certain properties of maturing elements in context by analyzing data which was traced by ‘Representation’ services. Calculated attributes reflect certain characteristics of persons, knowledge artifacts persons’ are dealing with and processes where persons’ get involved in [Kump et al., 2011].

‘Modeling’ services use certain inference mechanisms and computations exploiting stored information to gain knowledge about targeted maturing elements so that results can be distributed to ‘Reseeding’ services which in turn spark ‘Knowledge Maturing’ [Kump et al., 2011].

2.2.2.1.3 Reseeding

‘Reseeding’ services help to organize, generalize and formalize the knowledge base both for guiding evolutionary growth as well as supporting explicit guidance in the process of gardening of knowledge structures. They either work directly on available knowledge representations or use calculated properties of maturing elements from ‘Modeling’ services [Kump et al., 2011].

‘Reseeding’ services provide intelligent recommendation and information delivery mechanism and aim at both making knowledge workers aware of KMA activities and respective results as well as making contents usable. These services decisively increase the value of knowledge in specific contexts by additional measurements allowing further refinement.

From above theoretical and conceptual findings including the concepts of KMA, KMGA, KMI, TI, POLME, KMS and TI, this work’s aim should be explicated with regard to be shown demands evolving from respective concepts.

2.3 Explication of Aim of Work

The aim of this work is to support maturing of knowledge artifacts (resources). This shall be achieved by calculating KMI for digital resources used in a POLME. KMS shall be developed for the calculation of KMI and respective provision to users for selecting resources with regard to maturity statuses. Integrated in tools, KMI shall enhance knowledge workers' experience in dealing with information. Presenting calculated KMI for digital resources directly in POLME shall help in working off various KMA and setting up a basis for more successful KMGA. Hence, the outcome of this work shall provide the background for deciding whether it is possible and useful to calculate KMI for knowledge artifacts so that the maturity status of respective artifacts can be reflected directly in tools. The specific approach followed in this work shall provide knowledge workers with support in KMA so that maturing of knowledge artifacts can be fostered.

2.3.1 Relevant Considerations for Supporting Knowledge Maturing of Resources

To be more specific on the aim of this work, following considerations provide more details with regard to underlying theory and practical application of 'Knowledge Maturing'.

It is suggested to develop KMI in order to be able to provide maturing support services for knowledge workers in organizational learning environments [Schoefegger, Seitlinger and Ley, 2009a]. Moreover, it is strived for the presentation of different types of resources (at different maturity stages) in a uniform way [Schmidt, 2005a] as well as KMI are sought for judging micro-content [Schmidt, 2007].

KMI make 'Knowledge Maturing' observable and support 'Knowledge Maturing' aware tools as well as instruments for monitoring 'Knowledge Maturing' [Barnes et al., 2011]. It is supposed to encourage maturing of resources by creating awareness for them so that users become aware of the existence and need for improvement of those artifacts [Braun and Schmidt, 2007]. Explicit knowledge is a good starting point for assessing organizational knowledge and its maturing but little research is done for sophisticated concepts for maturing existing explicit knowledge [Riss et al., 2009]. It is reasonable to choose KMI for this end as the bottleneck in assessing the maturity of knowledge artifacts is the selection of qualified attributes [Schoefegger et al., 2009b]. From that, this work makes use of KMI in order to provide a means to make 'Knowledge Maturing' traceable as well as to make knowledge workers aware of knowledge artifacts on different maturity levels.

Chosen KMA are relevant for this work's goal. KMA 'reflect on and refine work practices or processes' together with other KMA would be most interesting to be supported by software or services [Kaschig et al., 2010]. Moreover, 'assess, verify and rate information', 'find relevant digital resources', 'familiarize oneself with new information' and 'keep up-to-date with organization-related knowledge' got considered as (very) interesting KMA too. Concluding that there also exist critical KMA like 'assess, verify and rate information', more support should be attributed to them and software services must be developed and enhanced to help overcome drawbacks by missing aid in crucial KMA [Kaschig et al., 2010]. From that, this work takes selected KMA into account in order to support knowledge workers. This work will be directly intervening in search processes and visualizing of information on organizational resources increasing the efficacy of to undertake KMA.

As each phase in the 'Knowledge Maturing Process' can be attached with concepts 'Drives', 'Motives', 'Actions' and 'Practices' observing what underlies respective phases [Riss et al., 2009], instances of those concepts in turn represent aims for this work from the 'Knowledge Maturing Process Model' perspective. 'Drives' basically describe underlying reasons for knowledge workers to engage in the 'Knowledge Maturing Process', e.g. acquiring new knowledge or defending knowledge they developed and shared. Hence, getting presented with calculated KMI upon resources helps users to reach their goals by bringing in a different perspective upon artifacts they work with. Instantiations of concept 'Actions' more directly describe what the calculation of KMI can do for users actually using KMI within their work environment. For example, KMI provide a means to judge / validate, experiment with and learn about knowledge artifacts. Additionally, chosen instances of concepts 'Practices' and 'Motives' like monitoring, curiosity and creativity give a hint where the result of this work shall help too.

KMGA got defined to express the relevance of KMA not undertaken by employees directly but brought in by organizational interventions in daily work. This form of guidance supplementing the 'Knowledge Maturing Process' either can take place by responsible persons or by automatic interventions by defined tools. It is suggested to foster guidance-oriented roles, activities and artifacts that are deemed to positively influence 'Knowledge Maturing' [Barnes et al., 2011]. The calculation and display of KMI allows to support various KMGA, especially 'recommend, suggest & advice', 'irritate & challenge', 'make aware',

‘evaluate & assess results’ and ‘monitor activities & progress’. By the application of KMI to resources, displaying KMI as additional information on resources and using KMI for search operations several defined KMGA can be supported.

This work will be integrated into a POLME in order to support the user directly where help is needed with regard to ‘Knowledge Maturing’. As resources in a system change regularly, it is necessary to elaborate on how resources develop over time and how this can be made visible to users [Weber et al., 2010]. Calculating and attaching KMI to resources allows gardening activities to focus on parts of the content contained in a POLME especially important (highly used) but of possibly poor quality (low readability) [Schmidt et al., 2009]. In order to be able to provide this work directly for users of a POLME, this work shall supply certain KMS.

To actually calculate KMI for resources, the concept of TI will be used. Within an organization the group as well as the individual implicitly assesses knowledge by using it in a certain manner. As knowledge usage assesses the integration of knowledge management activities into the operational work, such activities are already general indicators for the maturity of artifacts [Riss et al., 2009]. Moreover, it is necessary to inform about users’ activities [Lih, 2004]. From this and the fact that TI have a close relationship to KMA and KMI (as described above), it becomes reasonable using the concept of TI for the calculation of KMI.

Summarizing, this work elaborates on the question ‘**Whether it is possible to calculate KMI for knowledge artifacts (digital resources) with the help of TI within a POLME so that results presented to users provide a means to support KMA**.’

After basic concepts which will be used in this work got presented, the next chapter will explain additional concepts / approaches relevant in this context.

3 Measuring Maturity of Digital Resources Based on Usage Data

Following the aim of this work, first, a framework for keeping track of the multifaceted context of knowledge elements and second, certain KMS responsible for aggregating and storing of TI, for calculating KMI upon traced TI and for using results in respective POLME will be provided. In case of this work, instances of knowledge elements will be knowledge artifacts from within a POLME. On the one hand, knowledge artifacts can be defined as digital resources more generally, whereas TI have a close relation to similar concepts, e.g. 'Attention Metadata' on the other hand. Additionally, the KMI calculation procedure and its results have to be made available from within other KMS as well as from within software applications users are dealing with (POLME). For this end, this work makes use of the concept of 'Resource Profiles'. Respective concepts will be explained in more detail in the following.

3.1 Digital Resources

In general, a resource can be defined as follows: *"What makes something a resource is nothing more than the fact that somebody, at some time, considers it to be a resource."* [Downes, 2004, p. 3]. Further, a resource can be considered as *"[…] anything that, for whatever reason, someone has found necessary or useful to describe […]"* [Downes, 2004, p. 2].

Resources can be classified according to various criteria including the differentiation into digital and non-digital resources[1]. In the realm of 'Knowledge Maturing' wherein knowledge manifests and gets observable in (digital) knowledge artifacts, these knowledge structures can be called digital resources and will be abbreviated as resources in the following.

To enable people to create, store, locate and retrieve resources [Buchanan, 2006], it must be possible to distinguish one resource from another in a reliable manner; otherwise access to resources would be random [Downes, 2004]. As this work heavily builds on resources within the calculation process of KMI, so called 'Resource Profiles' are of relevance.

3.2 Resource Profiles

A 'Resource Profile' (RP) [Downes, 2004] is an aggregated description of a resource. From that, it becomes apparent that resources do have or do not have certain properties

[1] http://ltsc.ieee.org/doc/wg12/

distinguishing them from others. RP help in describing resources following that the sum of resources' properties denominates a resource according to various characteristics. A RP is a multi-faceted, wide ranging description of a resource which neither conforms to a particular scheme nor gets authored by any particular author [Downes, 2004]. For different resources, different sets of properties can be applied.

In relation to traditional resource descriptions, e.g. 'Learning Object Metadata'[2], possibly stored in dedicated repositories [Neven and Duval, 2002] focusing on describing underlying resources with defined properties for a specific purpose, RP integrate various information available from several sources at various locations [Downes, 2004]. From that, a RP provides a much more complete picture. A RP is made of and derived from contributions by many people. In its best case, the gathering of information residing in a RP happens automatically [Downes, 2004].

The RP provides an objective view upon what the resource constitutes and provides a means to select certain resources from others in order to satisfy certain needs of users as well as tools. In the case of this work, RP will be used in order to retrieve necessary information previously gathered from within a POLME for the calculation process of KMI as well as to store the results of calculation processes within the RP. The concept of RP allows describing a resource with regard to various properties available before, during and after the calculation process of KMI.

Further, the concept of RP allows describing resources independently of their types. As in the setting of this work resources in different stages of maturity and of different types are of relevance, it becomes apparent to design the container of available resources in the POLME with regard to the concept of RP.

According to the 'Knowledge Maturing Process Model', resources exist in different states of maturity and get improved over time. Hence, RP are sought to reflect resources on different levels of maturity as well as to foster discrimination of resources upon their maturity. Figure 5 shows examples of learning task materials which develop during their maturity improvements.

[2] http://ltsc.ieee.org/wg12/

The RP is sought to handle various types of resources. It is necessary to consider raw textual resources as well as textual resources incorporating multimedia content like typical email, office or PDF documents. Moreover, links to online resources as well as multimedia files like videos, music or images are proper resources representing content to be described by the RP in case of this work. Furthermore, containers of resources like folders are recognized as resources in this case too.

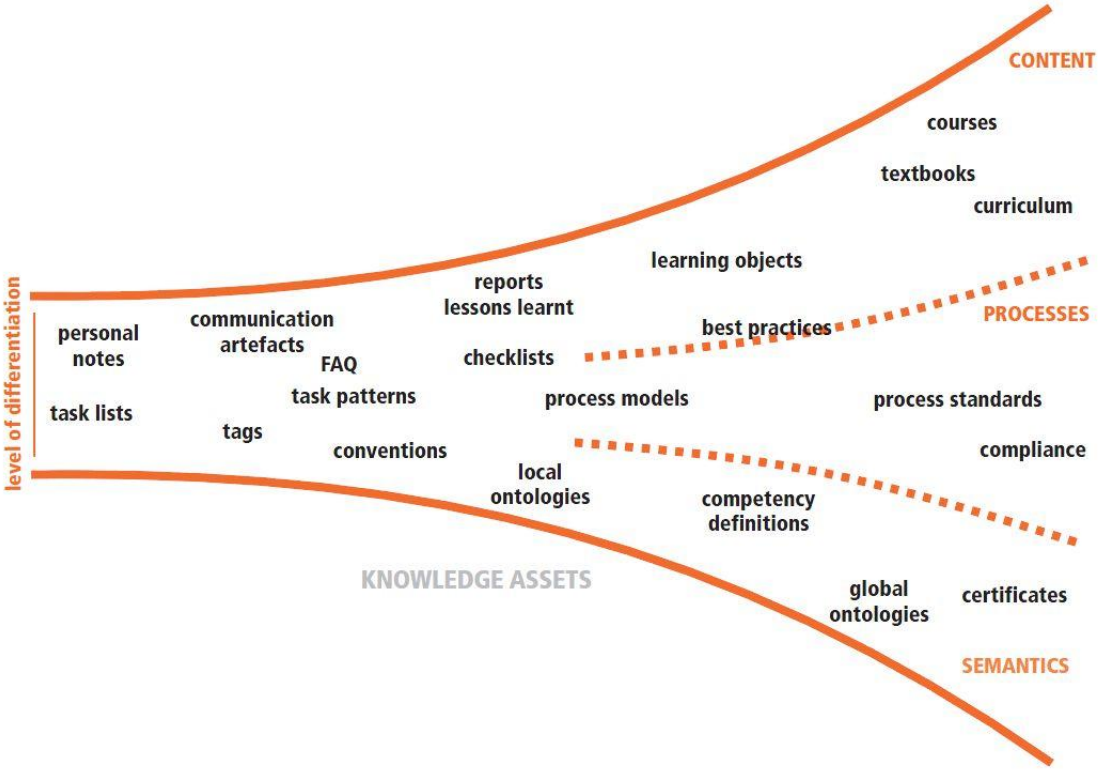

Figure 5: Maturity Development of Learning Material [Schmidt et al., 2009, p. 5]

This work tries to model resources with regard to certain maturity characteristics so that results directly can be used by knowledge workers within a POLME. On the one hand, the descriptions of resources in RP initially get filled during the calculation process of KMI and get enriched with calculated KMI on the other hand. A similar concept of describing system entities with regard to certain characteristics in order to gain deeper knowledge about respective subjects can be found in the realm of 'User Modeling'.

3.3 User Modeling

A 'User Model' (UM) is a representation of users' characteristics within a system [Brusilovsky and Millán, 2007] so that for example learning systems are able to personalize their behavior or adjust the 'Graphical User Interface' (GUI)[3] to the needs and requirements

[3] http://en.wikipedia.org/wiki/Graphical_user_interface

of the user. Related application areas of UM for example can be found in information retrieval tasks [Kelly and Teevan, 2003] and personalized recommendation systems [Schoefegger, Seitlinger and Ley, 2010] or when modeling the user's knowledge in certain domains [Lindstaedt et al., 2009]. It becomes apparent, that in 'User Modeling' the focus lies on determining the characteristics, preferences, knowledge and etc. of a certain user or person. In difference to this work, the focus does not lie on the characteristics of resources the users are dealing with but on the users themselves. This work aims for modeling resources with regard to certain characteristics yielding rich descriptions of resources considering their maturity.

As UM get used in the backend and / or as basis for, e.g. recommender systems [Burke, 2002], adaptive systems [Jameson, 2003] or extended knowledge services [Rath et al., 2008], it is necessary to gain as much information of users in order to provide desired functionalities. Data needed for inferring certain characteristics and preferences of users most likely have to be collected unobtrusively [Lindstaedt et al., 2009]. This work takes over this approach in order to be able to collect TI trigged by the use of a POLME which in turn will be used to calculate KMI. In the realm of unobtrusively extracting information either about the user itself or about the context of the user (including resources the user is dealing with) mainly two concepts relate to this work, i.e. 'Knowledge Indicating Events' and 'Attention Metadata'.

'Knowledge Indicating Events' (KIE) [Lindstaedt et al., 2009] were first introduced in the APOSDLE [Lindstaedt, Ley and Mayer, 2005] project. KIE get represented by certain events triggered by users of the (APOSDLE) software which in turn get used for detecting whether users have specific knowledge levels. The events available in the system get determined by the implementation of the GUI and get collected without interrupting the user during her work. This concept is very similar to the concept of TI from within the field of 'Knowledge Maturing' beside the fact that KIE got used to reflect the knowledge state of the user whereas TI originally identify certain user actions within a POLME making 'Knowledge Maturing' visible. As this work will use TI to calculate certain characteristics of resources, mainly events concerning the work with resources will be relevant.

Another approach dealing with events implicitly generated by users through application of software tools can be found in 'Attention Metadata'. 'Attention Metadata' in turn explains the nature of such events in more detail.

3.4 Attention Metadata

‘Attention Metadata‘ (AMD) is defined as the intersection of data about the user and the involved content and applications, hence represents the information on activities the user has carried out with digital content [Wolpers et al., 2006]. Rooting in information technology, this term got introduced by Steve Gillmore [Najjar, Duval and Wolpers, 2006].

From that, the AMD approach provides a means to collect and formalize user-related observations on how and what the user spends attention on [Wolpers et al., 2007]. For this approach, anything that captures the user’s attention during performing interaction with a tool is for particular interest [Tomadaki, Scott and Quick, 2007]. As the user deals with artifacts (documents) in a particular way specific only to her [Wolpers et al., 2007], digital resources like web pages, word files, emails and etc. are under focus in the context of AMD [Wolpers et al., 2007]. AMD helps to elicit for example what a user likes, reads, publishes and indicates what time she spent on a web page by exploiting rich information of the traced relation between the user and the content [Najjar, Duval and Wolpers, 2006].

Usage records gathered with the help of AMD including for example search activities [Najjar, Duval and Wolpers, 2006] enable improved personalized services [Wolpers et al., 2007]. Patterns emerging from the usage of tools can be the basis for recommendation of relevant information or visualization of user activities [Wolpers et al., 2007]. Moreover, it is possible to develop significantly improved and personalized information retrieval services by characterization of the use of knowledge in specific contexts [Memmel and Dengel, 2007].

With the help of AMD for information systems it is possible to, first, derive precise specifications about the usage and allocation of knowledge in certain working conditions, second, show the correlation of knowledge to the individual user [Wolpers et al., 2006], third, define requirements for the development of learning and knowledge supporting actions and fourth, allow the user of tons of information to find it [Chirita et al., 2005].

As TI and AMD are very specific to the environment where they can be observed but having two unifying more general concepts behind, the concept of ‘Attention’ as well as ‘Metadata’ should be explained shortly.

3.4.1 Attention

Generally, 'Attention' can be defined *"[...] as the set of processes enabling and guiding the selection of incoming perceptual information."* [Roda and Thomas, 2006, p. 6] for everyday handling of digital content [Wolpers et al., 2007]. This has to happen unobtrusively by observing the user behavior in handling her daily knowledge work, i.e. teaching, learning or handling a large amount of information in a personalized way [Wolpers et al., 2007]. Moreover, the computer is the actual source of observation in order to detect the context and content touched by the user [Wolpers et al., 2007].

3.4.2 Metadata

Metadata provide a way for digital artifacts to be discovered or located while many digital resources are non-textual and cannot be discovered via full-text searching [Wiley, 2000]. There exist two main types of metadata, objective and subjective metadata [Wiley, 2000]. The former represents properties of a subject which can be instantiated with meaningfully falsifiable values such as the author or file size. Latter represent properties to which meaningfully falsifiable values cannot be assigned like the artifacts' usefulness or inherent meaning.

Metadata should be generated automatically while the user works [Chirita et al., 2005] as people are not good in observing their own behavior [Doctorow, 2001]. To avoid this problem, automatic methods exist in order to create meaningful metadata, though some metadata with requirements to be true can only be created by humans [Doctorow, 2001].

Similarly to the AMD approach, this work collects user (usage) data in order to elicit and develop enhanced support in daily knowledge work with the help of software-based tools [Wolpers et al., 2006]. AMD is able to describe what users in community settings are dealing with in certain work situations [Tomadaki, Scott and Quick, 2007]. As the goal of this work is slightly different to the AMD approach, namely to describe resources with regard to their maturity instead of focusing on what users have done with resources in order to retrieve some kind of model about the user, the concept of AMD will be adapted so that usage data's underlying resources become much more important by describing them in RP.

As TI with focus on user activities but related to 'Knowledge Maturing' are similar to the concept of AMD, TI used in this work will be able to, first, detect the knowledge artifact with

which the user performs certain KMA and second, provide metadata with which the underlying resource is able to be described either directly or more sophistically with the help of from TI calculated KMI.

Based on the users' recent attention, it is possible to determine the relevance of specific content [Najjar, Duval and Wolpers, 2006] and construct recommendation procedures [Ochoa and Duval, 2006] or adapt systems to the users' information needs [Henze and Nejdl, 2003]. In the context of this work, resources won't be recommended relying on users' preferences based on a UM describing what users are interested in but with regard to calculated KMI of respective resources. With the help of KMI resources in turn will be made visible to users of a POLME so that direct access of resources with different maturity levels gets possible.

So far, the main concepts which will be used in this work have been described. The next chapters will focus on the setting for this work first, before describing developments realized in order to support the user of a POLME in certain KMA with the help of KMI calculated on usage data for resources. Evaluations done on results of this work will be described subsequently.

4 Calculation of Knowledge Maturing Indicators for Digital Resources

This chapter will describe implementations done for calculating KMI for resources based on usage data represented by TI. Additional developments done in order to actually make results of the calculation process available for supporting specific KMA in a certain POLME will be described in chapter 5.

Before describing the developments for the usage-based calculation of KMI for resources in detail, this chapter shows, first, the context of this work concerning the embedment of its implementations in a concrete project, second, the actual end-users for which respective development results of this work could be implemented and third, the specific POLME into which this work got integrated. Afterwards, the design for and realization of KMI calculation steps will be described. Chapter 5 describes how results got made available within search processes as well as for resource detail views.

4.1 Setting

First, concrete settings in which this work got involved shall be described.

4.1.1 MATURE Project

This work got embedded in the 'MATURE' project[4] responsible for conceptual research in the realm of 'Knowledge Maturing' and examination of its current application in real world settings. 'MATURE' is funded by the European Commission with € 9.1 million over 4 years and involves 12 partners from five different countries [Braun et al., 2009]. Beside the work on building a sound model of the 'Knowledge Maturing Process', 'MATURE' looks at learning on demand embedded into work processes responding to both requirements from the work situation and from employee's interests [Schmidt, 2005a].

4.1.2 MATURE Design / Development

Throughout 'MATURE' four interplaying strands report research results [Kaschig et al., 2010]. The empirical strand is conducting different forms of studies, the conceptual-technical strand is conceptualizing 'Knowledge Maturing' support and implementing tools, the integration strand is developing a flexible infrastructure and enabling loosely coupled solutions and the evaluation strand consists of participatory design activities and formative and summative evaluation.

[4] http://mature-ip.eu/

From that, for this work the empirical strand lied out basic theoretical concepts like the concept of KMA, KMGA, KMI and TI. The conceptual-technical part provided the environment for the integration and application of this work in a POLME. The evaluation strand as such plays a role concerning the provision of application partners where parts of this work got evaluated.

In principal, 'MATURE's' approach for designing and implementing prototypical software for the concept of POLME follows the 'Participatory Design Approach' [Muller and Druin, 2003] which gets combined with 'Deep Learning Design' [Ravenscroft and Boyle, 2010]. Design studies generated scenarios, feedback and initial ideas from application partners and evolved practical experience for supporting the 'Knowledge Maturing Process' [Ravenscroft, 2009]. Moreover, 27 'Use Cases'[5] followed in correlation with KMA helping to develop a set of 'Demonstrators' focusing on certain aspects of the 'Knowledge Maturing Process' linked to end-user feedback [Kaschig et al., 2010].

'MATURE's' application partners get involved in the design as well as in development phases, hence continuous feedback of various organizations involved is received. Feedback either comes directly from workshops wherein newest developments are presented and discussed or from questionnaires handed out to the 'Demonstrator' users indicating what users are interested in as the development goes on.

As this work got integrated in a certain POLME represented by a specific 'Demonstrator' applied to concrete end-users, chosen application partners and respective 'Demonstrator' will be described in the following.

4.1.3 Application Partners

For the setting of this work, two 'MATURE' application partners were of specific interest.

4.1.3.1 Connextions Kent

'Connexions Kent' (CK) in the UK beside other business activities is responsible for career guidance and personal development services applied to 13-19 year olds and those up to 25 with learning difficulties and disabilities [Attwell et al., 2008]. CK provides its services either directly in schools or colleges or at CK access points [Attwell et al., 2008] [Nelkner et al.,

[5] http://en.wikipedia.org/wiki/Use_case

2011]. In the context of the ‘MATURE’ project CK was involved in the continuous development of the POLME where this work got integrated.

Persons employed by CK are recognized as knowledge workers [Attwell et al., 2008] as they need to use different sources in order to obtain current and accurate information on the labor market on a routinely basis [Attwell et al., 2008] as well as knowledge and understanding required for career guidance is both heavily context dependent and dynamic [Nelkner et al., 2011]. Knowledge workers at CK extract and extrapolate key labor market information and interpret and manipulate these data for clients of their service [Attwell et al., 2008]. Moreover, they need to find appropriate and up-to-date information as fast as possible for the current work context [Ravenscroft, Schmidt and Cook, 2010]. Besides that, there is the organizational need of achieving a coherent and high quality organizational identity in the development of knowledge artifacts [Ravenscroft, Schmidt and Cook, 2010]. Special trained personal advisors [Nelkner et al., 2011] have to continuously extend and improve their knowledge for a better understanding of the labor market to be able to personally consult individuals on their job prospects [Schmidt et al., 2009]. They make use of a large body of formally documented knowledge artifacts (e.g. statistics and reports on job opportunities or labor market development in certain employment sectors and regions) but also draw on a considerable amount of informal knowledge derived from their experiences within concrete cases [Nelkner et al., 2011]. Each knowledge worker is self-responsible to manage work, information, contacts, relations to organizations and clients. Thus, well established knowledge as ‘knowing who’, ‘knowing what’ and ‘knowing where’ is the most important quality [Nelkner et al., 2011].

4.1.3.2 Structuralia

‘Structuralia’ (SA) is an e-learning provider located in Madrid, Spain. SA offers its clients over 160 e-learning courses primarily in the construction sector. Apart from the courses themselves, they also offer individual learning solutions to their clients including advice as to which of their courses may be relevant for them [Bradley et al., 2010]. In the context of the ‘MATURE’ project among other things SA mainly was responsible for providing an evaluation environment of developed ‘Demonstrator’ relating to this work.

For evaluation purposes of this work beside other evaluation work within the ‘MATURE’ project, a module of an online course on project management was chosen [Nelkner et al., 2011]. The course consists of 300 hours. It normally would run for around seven months and

is divided into 5 sub-modules. The course was provided to a group of employees from a large Spanish construction engineering company, i.e. 50 to 70 engineers. Each module consisted of a two or three day onsite course while most of the learning was done online. In particular one module of the seven modules of the overall program was chosen to be supported by the POLME.

The objective of the course was to provide knowledge, strategic methods and techniques needed for correct project management. The course was based on real case studies so that it would allow project managers to run past projects with reduced costs, shorter schedules and fewer resources [Nelkner et al., 2011]. The course was offered through a methodology that allows an intense participation and interchange of ideas with other students and teachers. It allowed development of team work on a compatible frame with professional duties of a group with expanded shifts aiming at collaboratively developing a shared understanding of the domain by collecting and critically discussing (controversial) information [Bradley et al., 2010].

As both application partners used the same POLME either continuously or within a defined evaluation period, the underlying 'Demonstrator' representing the POLME relevant for this work will be described in the next chapter.

4.1.4 Demonstrator 1

'Demonstrator 1' (DS1) subtitled with 'Assuring Quality for Social Learning in Content Networks' [Bradley et al., 2010] got implemented in order to be applied to the requirements of CK and SA. As such, DS1 provides the basis for the technical development of this work. The aim of the DS1 for CK is to support personal advisors in knowledge sharing, communicating and creating up-to-date, qualitatively high artifacts within their work and learning context [Nelkner et al., 2011]. Maturing of textual content is the basis for advisors to gain new knowledge or update knowledge about a certain topic in order to provide suitable advice across geographically dispersed locations [Nelkner et al., 2011].

DS1's main functionality with regard to the setting of CK is to collaboratively work on diverse contents. Moreover, it provides a place for retrieving interesting resources from the internet or even from a user's desktop in order to subsequently share it with colleagues or file them in the digital environment so that resources get (re-) findable. To (re-) find documents

the concept of ‘Tagging’[6] gets used in order to be able to attach certain keywords to resources categorizing artifacts with regard to, e.g. their content, topic they deal with, personal thoughts and etc. ‘Tagging’ in this setting also gets used for searching for resources in the system. Rating possibilities provide a mechanism to collaboratively assess documents. Additionally, discussion functionalities on artifacts enable the exchange of knowledge by question answer principles. To enable easy collecting and sharing of already existing or recently found contents, so called collections are available.

DS1 is constituted of a client and server side part. The following description of DS1 focuses on the client side part as the server side will be explained in more detail when it comes to the implementation of this work.

4.1.4.1 Widgets

DS1 was developed in a widget-style manner. The ‘Widget’ approach has been made popular in several domains over the last years represented by compact, specialized applications or application components [Stefaner et al., 2007]. Widgets in principal provide a simple GUI packed in single windows to provide a defined functionality. These little windows can be moved throughout the screen by drag and drop and can be adjusted in size so that they become easy to keep on screen in order to support the user during her work. Only certain widgets are supposed to be open when specific functionality is needed.

Sidebar

The ‘Sidebar’, which can be considered as the starting point for users using DS1, provides basic functionalities in order to keep the user’s installed widgets organized. From there, the user is able to install a variety of widgets according to his needs. The ‘Sidebar’ as shown in figure 6 shows all installed widgets in order to select and start desired functionality as well as to provide a place where widgets can be assigned to working sets. The user has to login through the ‘Sidebar’ first, in order to use the widget’s functionalities subsequently.

Login Widget

This widget gets opened when the user accesses DS1 for the first time or was logged out after the last work with DS1.

[6] http://en.wikipedia.org/wiki/Tag_(metadata)

Tag Cloud Widget

This widget provides an interesting overview of tags currently used in the system shown in a tag cloud. The tag cloud shows the actual vocabulary used by the users in order to denominate resources with regard to purpose, topic, assigned persons, content or just whatever users think describes the artifacts best. The presented keywords can be displayed according to their visibility in the system. In principal tags can be assigned to resources as either private or public labels. This allows users to keep certain descriptions of resources for themselves or to first try out which tags fit best to the resources in order to continually improve their description by tags. Later on, users should assign certain public tags to resources for the use by all users in the system. The tags get displayed according to the tag cloud principal where more frequently used tags for (different) resources get displayed by a bolder font. To get an overview of resources to which certain tags where assigned, tags displayed are clickable in order to open the 'Search Widget' where results of a tag-based search will be shown.

Figure 6: Sidebar

Search Widget

The 'Search Widget' as shown in figure 7 provides the tag-based search functionality for resources in the system. The user can either type certain keywords (tags) in a predefined box manually or choose tags from within the system from a list on the left side of the widget in order to start the search by clicking the search button. The search results will contain all kinds of resources available in the system. The type of the found resource gets displayed by a little

icon placed at the left side of a search result entry. Each search result entry contains the 'Uniform Resource Identifier' (URI)[7], the assigned tags and the overall rating info of respective found resource. Another field within a search result entry contains the identifiers of users in the system who contributed to the resources.

The user is able to open a search result directly by clicking on the URI of the resource displayed. Moreover, the user is able to rate a resource directly from within the search result by selecting and applying a certain number of stars from the overall five-star-rating display field. Hovering a certain period of time over the search result entry with the help of the mouse pointer lets a little toolbar pop out from within the right side of the search result. This toolbar contains extended functionality in order to add the found resource to a collection, start a discussion about the resource or assign or change tags to / of the resource.

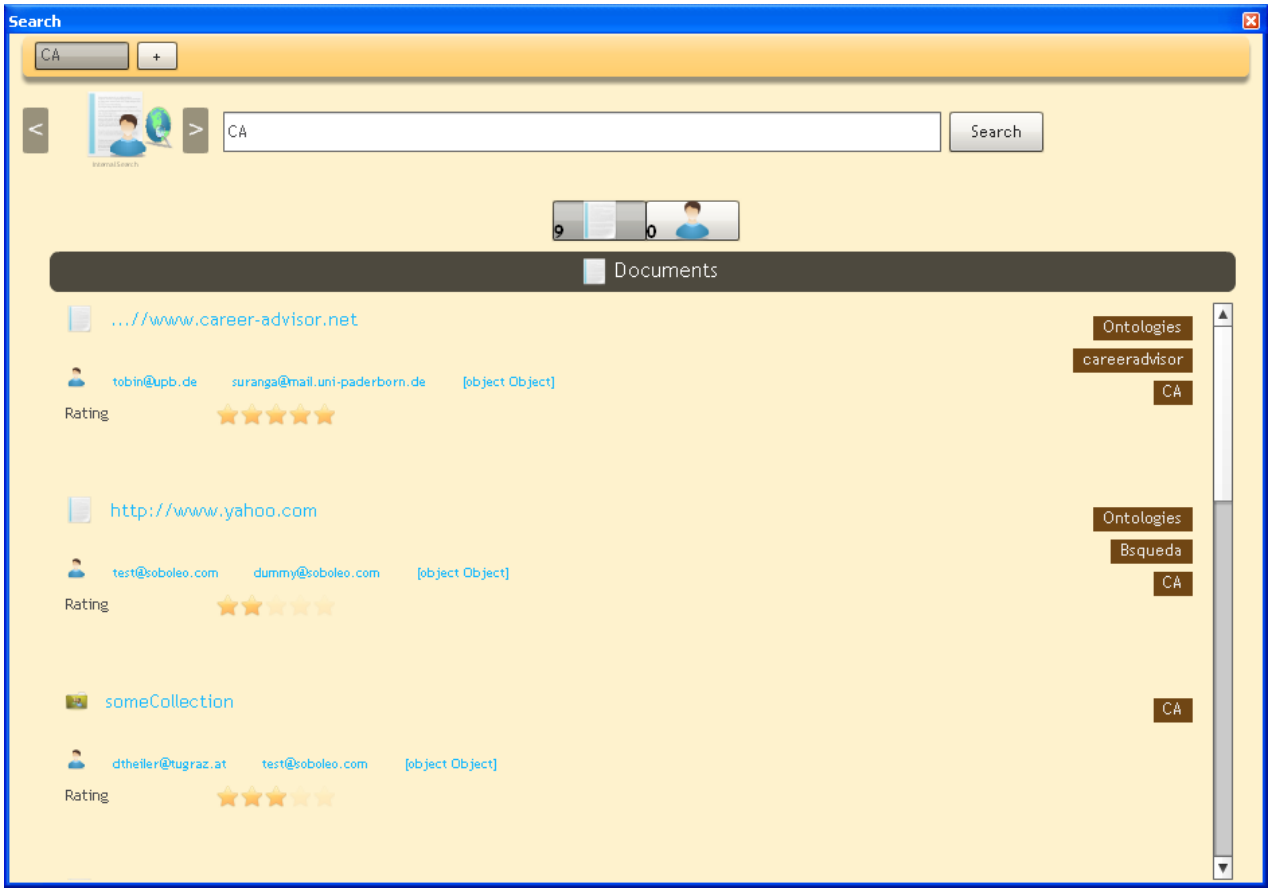

Figure 7: Search Widget

The list on the left side containing used tags from within the system gets structured according to a user defined collaboratively evolving hierarchy of tags, which can be edited via the 'Taxonomy Widget'.

[7] http://en.wikipedia.org/wiki/Uniform_resource_identifier

Tagging Widget

This widget as shown in figure 8 allows assigning and deleting private and public tags to or from a selected resource. The 'Tagging Widget' displays all current assigned tags in each category respectively. It gets also displayed which tags got assigned so far by the current user and how often a certain tag was assigned by the users in the system to this resource in total. To select a certain resource for tagging, e.g. the URI of a resource (e.g. from the web browser's address field), a local file residing on the user's desktop or other resources in the system like collections can be dragged directly onto a dedicated field at the top of this widget.

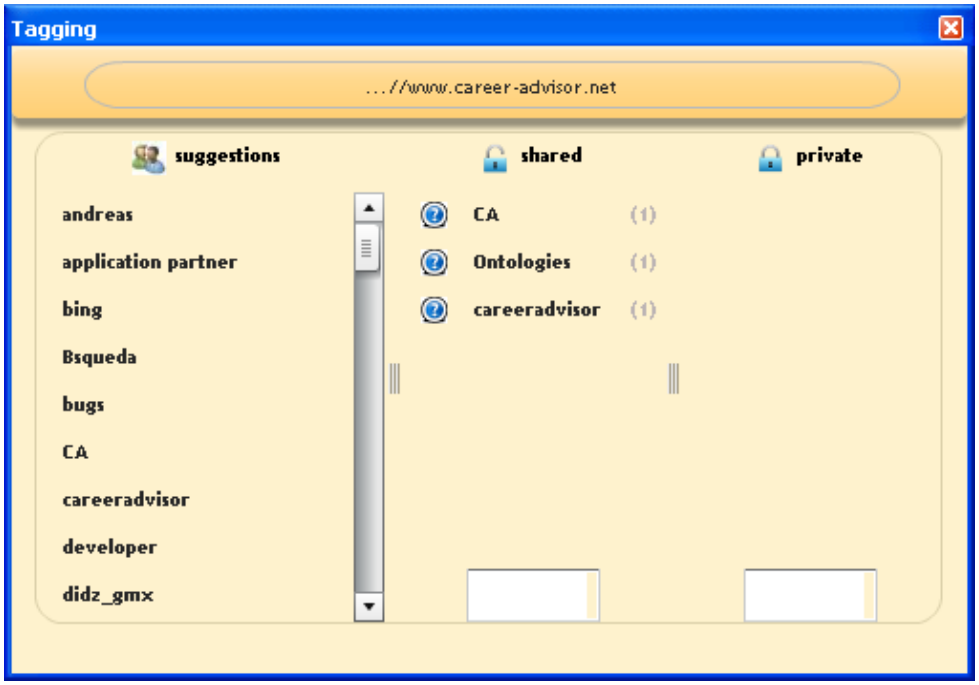

Figure 8: Tagging Widget

Taxonomy Widget

The 'Taxonomy Widget' in principal provides various functionality used in other contexts too from which only a selection is considered important for this work. The main purpose of this widget is to organize tags in the system according to their structure concerning their intended meaning. The users are able to choose from currently existing hierarchies to which tags were already assigned. If the user thinks a tag belongs to another (sub-) concept in the hierarchy, she is able to drag and drop the focused tag onto the target concept in order to subsume the tag under the target concept. As all users are able to change the tag structures within the system, a collaboratively designed vocabulary structure evolves describing resources together with their contents. As described above, this hierarchy gets displayed in the 'Search Widget' in order to provide a more sophisticated search input method.

Collection Widget

With the help of the 'Collection Widget' as shown in figure 9 the user is able to create private as well as public folders to collect her resources, to collaboratively work on those resources when respective folders get subscribed by other users within the system and to organize contained resources according to the user's need. First of all, collections (folders) can be created to be able to store newly retrieved or often used resources in a certain place with regard to the purpose of respective resources. To provide a place where resources brought into the system remain private to the user, it is possible to create folders only visible to the user.

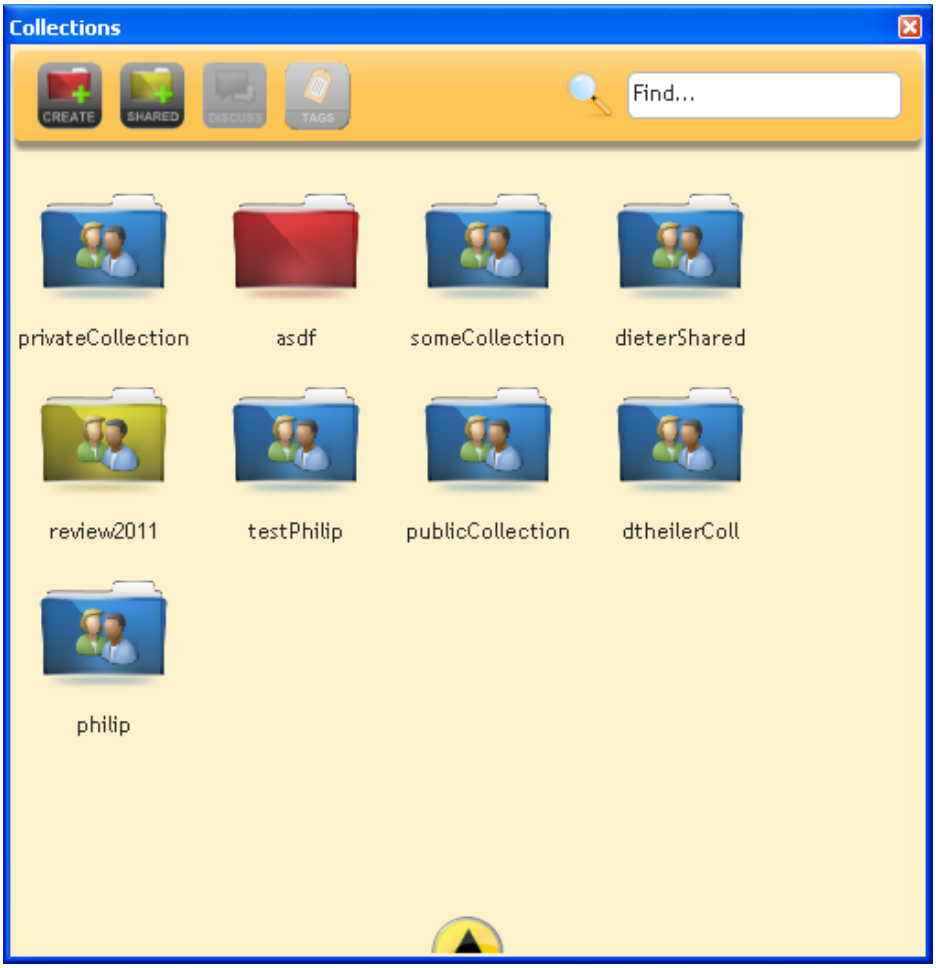

Figure 9: Collection Widget

The user can switch the collection to public mode whenever he decides that he wants to share a folder with other users. The user is able to drag all imaginable types of resources into such a folder. He is able to put documents from his own desktop and links to homepages or online videos and images in his folder. To provide access for other users to local files, these contents get uploaded to the server whenever a user drops an item into a collection from her desktop. The items in a collection can be renamed, tagged and rated from the context menu of each item. It is also possible to re-arrange items in the folder by drag and drop. If the user decides

to make a folder public for all users in the system, he just has to select the corresponding entry in the context menu. Similarly to the 'Search Widget', the 'Collection Widget' provides the possibility to start a discussion either on a collection itself or upon a collection item (entry in a collection). As the user initially just sees collections created by her, she can search for public collections in the system; found public collections can be chosen and subscribed. This allows the user to see respective collections together with own collections within widget. Moreover, the user is able to work with subscribed collections in the same way as he would be able to do with his own. By this way, contents of a subscribed collection can be changed collaboratively by users in the system. If the user decides that he wants to print the content of a collection to PDF, he is able to do so by selecting desired functionally from the toolbar.

Discussion Widget

The 'Discussion Widget as shown in figure 10 provides a place for collaboratively exchanging knowledge upon resources. The left side of this widget presents available discussions from which the user is able to choose desired ones. On the right side of the widget the user sees all current comments within a selected discussion. Users are able to ask questions or type answers or comments by the corresponding box on the right side.

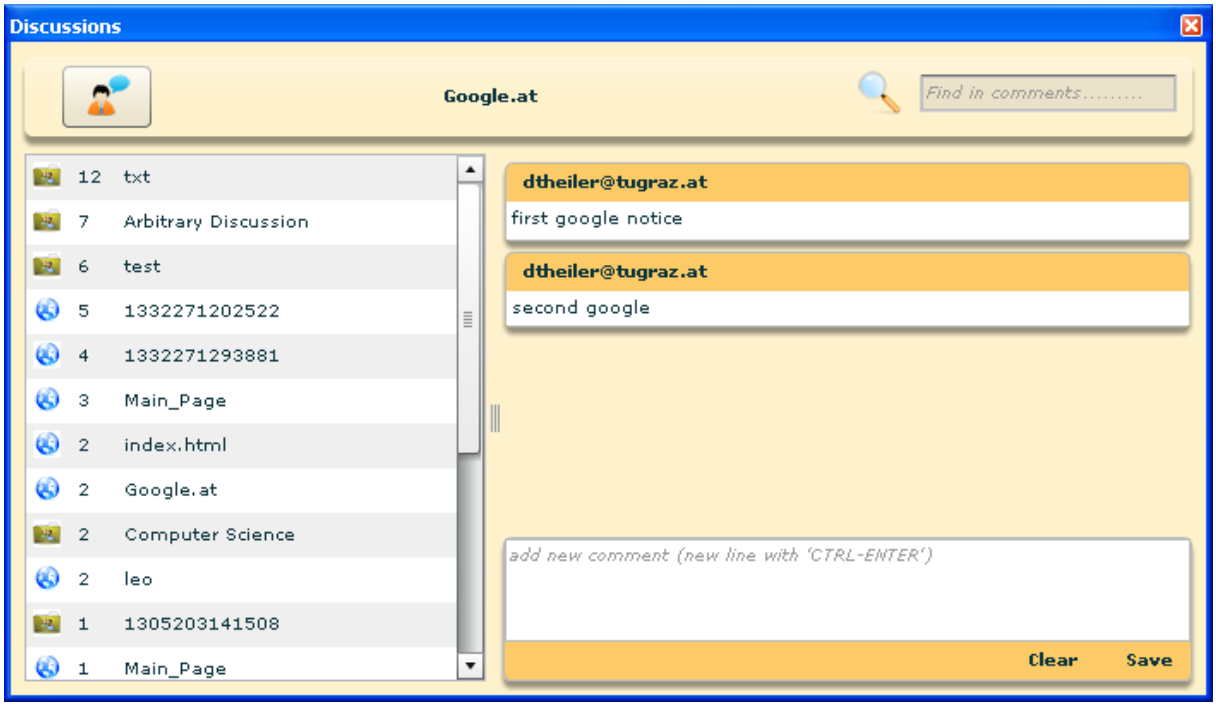

Figure 10: Discussion Widget

As the basic setting for this work got described by now, following chapters will focus on the design of the calculation process of KMI first, before describing technical developments.

4.2 Design

This work's aim is to calculate KMI for resources used in a POLME based on usage data. In its concrete case, the POLME used gets represented by DS1. DS1 provides several possibilities for working with various types of resources and for improving their maturity with regard to the concept of KMA. Considering the setting for CK, DS1 allows for resources to be developed in quality so that reliable and accurate career guidance services get possible.

As this work intends to elaborate on the possibility of automatically calculating KMI for resources contained in a POLME so that resources on different maturing stages can be presented in a uniform way as well as it gets possible to judge content with regard to its maturity level, a specific approach will be presented taking into account necessary changes in DS1. Each step in the design as well as implementation builds on results of previous steps so when sequentially applied in possible different contexts, outcomes can be tailored to respective specific environments.

4.2.1 Sought Outcomes

From the KMGA perspective, the result of calculating KMI for resources based on usage data shall yield the basis for 'monitor activities & progress', 'evaluate & assess results' and 'make aware'. As TI can reflect users' activities within a POLME, these will be used as the basis for the calculation process. The outcome of the calculation process represents an aggregation of past activities, hence will back KMGA 'monitor activities & progress'. The result of the calculation sequence will be KMI attached to resources which have undergone certain TI. Hence, available KMI make results of KMA upon explicit knowledge in resources visible so that 'evaluate & assess results' will be inherent to this work's results. Finally, 'make aware' can be executed with the help of calculated KMI in order to show users as well as domain experts interesting conditions in the knowledge base of the organization where DS1 gets applied.

To be more concrete on the actual outcome of the stepwise approach taken, the calculation process for DS1 yields a client side library handling tracing actual TI upon resources, a 'Usage Logging Knowledge Maturing Service' (ULKMS) on server side responsible for receiving and storing traced TI and a 'Resource Modeling Knowledge Maturing Service' (RMKMS) responsible for the calculation process of KMI with the help of stored TI. In principal, this work will be split in a client side and server side part. The client side part

resulting in a library named ‘Mature Components’ (MC) will be responsible for providing the basic environment necessary to trace chosen TI from within DS1 and to send these data to the server with the help of the ULKMS. The ULKMS on server side will be responsible for accepting traced data from the client side of DS1 and storing it into a ‘Triplestore’ (TS)[8] ‘Database’ (DB)[9]. Together, MC and the ULKMS provide the framework for collecting and storing usage data upon resources within DS1. Within these implementations, the client side of DS1 got analyzed with regard to available TI. From the fact that TI can be mapped to KMI and vice versa and that TI make KMA traceable in concrete settings (e.g. POLME), it is reasonable to rely on TI in the KMI calculation process. To be able to integrate usage data for the calculation process which cannot be directly related to TI, additional events got introduced. Together, available TI and events necessary for the calculation process additionally, will be denoted by ‘User Events’ (UE). Both, the fact that UE in general reflect user operations possible with a POLME and the fact that TI in turn make KMA in a concrete POLME traceable, provide a means for KMI to reflect the status of knowledge created and shared within a concrete software application. As resources are of major interest to make the collective knowledge construction process visible [Schmidt et al., 2009], taking into account various resource types makes it easier to identify solid and reliable knowledge. Moreover, the basic infrastructure for dealing with the communication between the client side and server side part got implemented so that respective KMS got able to work. Relying on these implementations, the RMKMS calculates selected KMI for resources after identifying used resources with the help of traced UE. In analogy to ‘User Modeling’, ‘Resource Modeling’ is sought to be the discriminative identifier for what the actual process of calculating KMI for digital resources shall be called. For that, first, the set of KMI had to be analyzed with regard to KMI actually being able to be represented by usage-based matters. Moreover, resource-based KMI got chosen describing the maturity of resources within the set of KMI for various knowledge manifestations, i.e. artifacts, cognifacts and sociofacts. Second, UE got assigned to groups manually so that resulting UE groups are able to reflect the semantics of chosen KMI. Third, as chosen KMI cannot be directly calculated upon respective UE groups representing specific UE aggregations for KMI, ‘Intermediate Indicators’ (II) got introduced. Respective indicators – mainly calculated upon resources’ UE frequencies within established UE groups – allow for a mapping between UE groups and KMI. Fourth, besides using UE for the usage-based approach in calculating KMI for resources, RP provide a means to represent respective resources with regard to UE and calculated II and KMI. The sum of RP available within DS1

[8] http://en.wikipedia.org/wiki/Triplestore
[9] http://en.wikipedia.org/wiki/Database

gets represented by introduced ‘Resource Model’ (RM). From that, the RM allows for the results of the calculation process to be stored in working memory and to be further applied within DS1; as such, the RM represents the core of the RMKMS.

Before describing the concrete implementations for the KMI calculation process, conceptual as well as technical considerations shall be described.

4.2.2 Conceptual

4.2.2.1 Resource Types

First, it was necessary to define basic resource types used in DS1 so that respective differentiation allowed for identifying possible TI upon concrete resource types. In principal, DS1 provides the possibility to use each type of digital resource for the work with career guidance-related content, i.e. textual as well as multimedia content. Moreover, DS1 provides access to contained resources through a possible grouping in collections, which can be considered as resource containers like folders in a file system. Furthermore, the concept of discussions has to be represented in a third type of resource. From a more technical point of view, those resource types have been defined as follows:

Digital Resource

The type ‘Digital Resource’ represents the most general type specification for a resource. All other resource types are subsumed under this type. A resource of type ‘Digital Resource’ can be anything that has got an URI like a web page or an image or video contained in a web page. Moreover, as various local files from the user’s desktop can be used in DS1, a link to a local file which has to be uploaded to the server subsequently in order to be accessible by others, is also considered as of type ‘Digital Resource’ in the context of this work.

Collection

The resource type ‘Collection’ is seen as descendent of type ‘Digital resource’. A collection represents a compilation of different resource types, i.e. a collection can contain another collection as well as a discussion or a resource of type ‘Digital Resource’. The items of a collection get denominated as collection items even though a collection item does not exist as independent type but gets represented by other resource types as well.

Discussion

A resource of type ‘Discussion’ represents a container for the deliberative collaboration on another resource type or another discussion. The ‘Discussion’ type is a descended of type ‘Digital resource’. As such, a discussion contains an unlimited number of discussion comments or so called discussion entries.

Next, as for certain resource types various TI will be defined, it is necessary to describe TI and their relation to UE together with necessary parameters.

4.2.2.2 Transition Indicators and User Events

TI reflect certain user activities within the POLME by which, first, users’ activities within the system get visible, second, defined KMA get traceable and third, KMI can be get calculated upon. In this work, these user activities get related to actual resources involved by the user performing certain actions in DS1. Concrete instances of TI only come to live by certain user operations through the POLME. For the sake of this work, available TI get extended by user activities not directly relating to certain TI. Both TI and these additionally introduced events get denoted as UE. Moreover, each TI / user activity represents a defined operation (event) on a resource at a certain point in time, hence the parameters of a UE look as follows.

4.2.2.2.1 Parameters

User

The user denominates the actual person triggering an operation by performing a certain action with the help of DS1. The action under focus gets observed by the part of implemented framework responsible for handling UE in order to recognize the currently logged in user.

Resource

The resource denotes the unique identifier for the resource upon a certain operation was performed. As there exist UE not used for the purpose of the calculation of KMI, the resource parameter will be left empty for those UE as long as no specific resource is concerned by the UE. UE not involving a certain resource, e.g. navigational operations within the POLME had to be integrated for different application scenarios but won’t be used in this work.

Time

For chronological operations, the timestamp of when a UE got triggered is important. It denotes in which order certain operations took place in order to allow the calculation of KMI

for resources. For long lasting operations, more than one UE is necessary to denote the duration of a certain operation.

Content

The content of a UE denotes additional information of what was used or changed by applying the current operation. This parameter takes an unlimited number of entries concerning information of what got used in order to perform the current operation. From this parameter, the calculation process gets able to conclude further aspects of the UE on a resource as well as additional descriptions of what the UE type conveys.

From that, it becomes apparent that the content parameter might be misused for transferring additional data including the whole set of resources' properties rather than chosen attributes of the operation triggering the UE. The content parameter must contain only small pieces of relevant information which give the UE more expressiveness as well as precious information necessary to interpret the UE in a specific context.

Type

The type of a UE denominates the actual kind of a TI which was triggered by the user performing on a certain resource. From that, the type together with the time parameter uniquely identifies the UE. The UE type incorporates a specific meaning besides just identifying the UE's type.

4.2.2.2.2 Types

According to the functionalities provided by DS1, a broad range of UE types could be derived from the analysis of possible KMA within DS1. From that, it became clear which TI could actually be used for the calculation of KMI on respective resource types. Hence, only UE got considered as TI and used for the calculation of KMI actually representing user operations being able to indicate maturity development of resources. Subsequently, a list of UE which root directly in TI could be arranged with regard to the resource type they consider. As further developments of the RMKMS yielded, additional UE events had to be defined in order to allow certain KMI to be calculated. For a complete list of derived UE please refer to the appendix.

After available UE got derived from analyzing DS1, it was necessary to define the list of KMI which will be supported by this work.

4.2.2.3 Knowledge Maturing Indicators

'Knowledge Maturing' theory provides a list of possible KMI to be applied when judging an organization's maturity level. In the case of this work, not the whole organizational context should be considered but digital resources within an organization representing explicit manifestations of knowledge available. From the set of KMI, mainly two KMI types can be considered for this work, i.e. activity and artifact-related KMI. Activity-based KMI are able to be related to TI from within a POLME as they are capable of considering maturity development of resources by looking at certain activities possible by the set of tools. Additionally, for the aim of this work resource-related KMI will be chosen, as this work focuses on the maturity development of resources excluding cognifacts and sociofacts more or less considering different aspects of different maturing elements available. As knowledge workers depend on information available within an organization in order to successfully perform daily work, digital artifacts are of great importance for knowledge workers. Hence, following chosen set of KMI gets used to be applied to digital resources within DS1.

Analyzing possible artifact-related KMI with regard to their capability of being expressible by usage-based matters yielded the set of to calculate KMI in table 3:

Knowledge maturing indicators
An artifact has been created/edited/co-developed by a diverse group
An artifact was prepared for a meeting
An artifact was created by integrating parts of other artifacts
An artifact has been the subject of many discussions
An artifact was changed in type
An artifact has achieved a high degree of awareness among others
An artifact is used widely
An artifact was selected from a range of artifacts
An artifact became part of a collection of similar artifacts
An artifact was made accessible to a different group of individuals
An artifact is referred to by another artifact
An artifact was presented to an influential group of individuals
An artifact has been accessed by a different group of individuals
An artifact has been used by an individual
An artifact was changed

An artifact has been accepted into a restricted domain
An artifact has been recommended or approved by management
An artifact has become part of a guideline or has become standard
An artifact has been rated high
An artifact has been assessed by an individual

Table 3: Knowledge Maturing Indicators

As certain chosen KMI shall be calculated upon traceable UE from within DS1, the process for the calculation of KMI shall be explicated in the next chapter.

4.2.2.4 Calculating Knowledge Maturing Indicators

In order to provide a means to calculate KMI upon to trace usage data, respective UE have to be assigned to certain groups basically reflecting given semantics inherent to chosen KMI and allowing TI to be mapped to KMI. Grouping of UE is reasonable when taking into account that certain KMI reflect outcomes of developments in maturity rooting in certain knowledge workers' activities. Moreover, maturity developments often consist of sequentially applied operations; as such are not made out of single user actions. Hence, grouping of UE makes it possible to integrate several KMA performed upon resources during the calculation process of KMI. It is also reasonable to use several UE for possible different KMI as certain instances of KMA (TI) yield developments in maturity observable by various KMI. In turn, grouping of UE simplifies relating outcomes of maturity developments to activities performed with resources as certain KMI root in either single or multiple TI. To be able to vary the mapping between TI / UE and KMI or to add or remove certain UE, respective KMI semantics as well as available UE from within the POLME have to be considered. The definition of respective UE groups for this work is done once, though further developments could consider possible changes in relating KMI back to TI / UE. Table 4 shows established UE groups which define the basis for the calculation of KMI.

User event groups	
changing	**adding and deleting**
addPrivateCollectionItem	addDiscussionComment
addSharedCollectionItem	addPrivateTag
createPrivateCollection	addSharedTag
createSharedCollection	removePrivateTag

addDiscussionComment	removeSharedTag
removePrivateCollectionItem	changeCollection-AddPrivateCollectionItem
removeSharedCollectionItem	changeCollection-AddSharedCollectionItem
removePrivateCollection	changeCollection-RemovePrivateCollectionItem
removeSharedCollection	changeCollection-RemoveSharedCollectionItem
renamePrivateCollection	**tagging**
renameSharedCollection	addPrivateTag
renamePrivateCollectionItem	addSharedTag
renameSharedCollectionItem	removePrivateTag
changeCollection-AddPrivateCollectionItem	removeSharedTag
changeCollection-AddSharedCollectionItem	**initial collection collaboration**
changeCollection-RemovePrivateCollectionItem	subscribeCollection
changeCollection-RemoveSharedCollectionItem	subscribeCollectionItem-SubscribeCollection
changeCollection-RenamePrivateCollectionItem	**initial discussion collaboration**
changeCollection-RenameSharedCollectionItem	createDiscussion
createDiscussion	startDiscussion
sharing with community	**remove initial collection collabroation**
shareCollection	unSubscribeCollection
shareCollectionItemShareCollection	unSubscribeCollectionItem-UnSubscribeCollection
createSharedCollection	**selected from others**
addSharedCollectionItem	startDiscussion

addSharedTag	viewEntityFromSearchResults
startDiscussion	addResourceToCollection-FromSearchResults
createDiscussion	rateResourceFromSearchResults
collaborate collection	**remove from collection**
structureSharedCollection	removeSharedCollectionItem
renameSharedCollection	removePrivateCollectionItem
removeSharedCollectionItem	**add to collection**
changeCollection-AddSharedCollectionItem	addPrivateCollectionItem
changeCollection-RemoveSharedCollectionItem	addSharedCollectionItem
changeCollection-RenameSharedCollectionItem	**person association**
structureSharedCollectionItem-StructureSharedCollection	exportCollectionItem
renameSharedCollectionItem-RenameSharedCollection	addPrivateCollectionItem
addSharedTag	addSharedCollectionItem
removeSharedTag	shareCollection
recommend	subscribeCollection
exportCollectionItem	unSubscribeCollection
shareCollection	createPrivateCollection
createSharedCollection	createSharedCollection
rateEntity	rateEntity
shareCollectionItemShareCollection	startDiscussion
addSharedTag	renameDiscussion
addSharedCollectionItem	addDiscussionComment
assess	viewEntity
rateEntity	renamePrivateCollection
addPrivateTag	renameSharedCollection
removePrivateTag	addSharedTag
removeSharedTag	addPrivateTag

addSharedTag	removePrivateTag
awareness	removeSharedTag
appearsInSearchResult	removePrivateCollectionItem
viewEntity	removeSharedCollectionItem
rateEntity	renamePrivateCollectionItem
addSharedTag	renameSharedCollectionItem
exportCollectionItem	structurePrivateCollection
addSharedCollectionItem	structureSharedCollection
changeCollection-AddSharedCollectionItem	**start discussion**
addDiscussionComment	startDiscussion
addDiscussionTargetComment-AddDiscussionComment	**appear in search result**
subscribeCollection	appearsInSearchResult
subscribeCollectionItem-SubscribeCollection	**export collection item**
replaceSharedTag	exportCollectionItem
structureSharedCollection	**collaborate discussion**
removeSharedTag	renameDiscussion
structureSharedCollectionItem-StructureSharedCollection	addDiscussionComment
viewing entity	addDiscussionTargetComment-AddDiscussionComment
viewEntity	renameDiscussionTarget-RenameDiscussion
change type	addSharedTag
shareCollection	removeSharedTag
subscribeCollection	**rating**
exportCollectionItem	rateEntity
unSubscribeCollection	**organizing in collection**
createPrivateCollection	structurePrivateCollection
createSharedCollection	structureSharedCollection
shareCollectionItem-	structurePrivateCollectionItem-

ShareCollection	StructurePrivateCollection
subscribeCollectionItem-SubscribeCollection	structureSharedCollectionItem-StructureSharedCollection
unSubscribeCollectionItem-UnSubscribeCollection	
createDiscussion	

Table 4: User Event Groups

The calculation of KMI mainly gets based upon the resources' actual number of UE reached from within certain UE groups. As more or less concrete instances of KMA get applied to resources, the actual frequency with which resources get developed in maturity in relation to others is sought to express given resources' maturity states.

As giving more or less value to certain kinds of TI / UE would yield unbalanced and possible false decisions when judging the maturity of resources, it is reasonable to not give preference to certain instances of UE but to integrate and value all undergone activities equally so that only the frequency of several undergone activities becomes important.

Actually relating certain resources to others within a POLME with regard to UE frequencies in respective groups allows for denoting the resources' maturity states upon the ongoing change in the overall knowledge contained in the system. Moreover, taking the frequency of UE for resources in relation to others into account (when calculating KMI) is sought to work, as KMI are not defined with the help of single numeric values but with qualitative descriptions of developments in maturity. As such, using frequency values of UE in respective UE groups does not consider predefined thresholds when it comes to deciding whether a resource reached a certain level of maturity but considers the actual overall condition of the knowledge base.

Further, this approach is sought to be able to express the maturity of resources regardless to whether resources get further developed with regard to content (knowledge) during certain periods of time, as reached low, moderate or high levels of acceptance within the community nevertheless can express the actual maturity of respective resources. This can be concluded from the fact that various UE and upon based KMI are able to express whether resources get accessed, used, shared and etc. more or less frequently, hence provide a means to express a certain resource's acceptance and popularity with the system yielding certain maturity states.

In turn, this shall allow for distinguishing between evolving or stable resources too when taking into account various UE types. From that, certain resources will or will not be able to reach certain KMI at certain points in time giving insight into the actual maturity state of the resource by different perspectives represented by KMI.

For that, frequencies with which a resource got focused by UE in certain UE groups get calculated. It is sought to calculate maturity states of resources at certain points in time, tough previously calculated KMI have to be taken into account when it comes to describing the resource with regard to past KMA. It is necessary not to 'forget' calculated UE frequencies for resources between different calculation periods to consider already reached maturity states of resources. Hence, not only resource's relations within the actual calculation period get expressed but resources' maturity developments spanning over more than a single calculation period.

In order to calculate KMI for resources, II get introduced to allow a mapping between calculated UE frequencies for resources in groups and KMI. From the fact that defined UE are always specific to the actual environment (POLME) where they can be triggered, II reflect KMI within a concrete setting based on defined UE groups. Whereas KMI convey maturity states by more abstract (textual) descriptions, II are the concrete instances of UE frequencies per UE groups for resources (in relation to other resources). II basically will reflect certain characteristics of resources actually denoting them as more or less mature within a concrete knowledge base at a certain point in time. Each instance of II possibly can represent one or more KMI whereas in turn single KMI often can be expressed by more than one II. This condition again roots in the fact that II are able to represent the knowledge contained in respective POLME on a finer granularity level than KMI would be able to, as II directly origin in UE (groups) specific to the environment. Following these considerations, a mapping of calculated II on resources to KMI must be done in order to be able to assess the maturing element (resource) with the help of KMI. Moreover, using II as intermediate properties of resources between UE groups and KMI lets the mapping between concrete TI and KMI go more flexible and easier.

In principal three types of II will be calculated. As not necessarily all resources will receive UE in specific groups, negative II indicate that a resource never has had a single UE in respective UE group. Positive II indicate that a resource has had at least one UE in respective

UE group. As negative and positive II can only indicate whether a resource has or has not received UE with in a UE group, discriminative II reflect indicators based on certain thresholds. Respective thresholds have to be calculated upon the overall UE frequencies of resources in certain UE groups, as such basically representing the mean values of UE frequencies per groups. It is important to calculate thresholds with regard to defined types of resources taking into account that only resources of same types can be meaningfully related to each other considering maturity states. This follows from the fact that mainly only certain resource types will receive certain UE so that relating resources of different types would not consider resource types' missing capability to be focused by certain UE. From that, discriminating II are capable of relating UE frequency characteristics of certain resources to others in the system. This then can reflect different maturity states of resources as certain resources received different numbers of KMA. Actual instances of II used to indicate the maturity of resources based on concrete UE groups get shown in table 5.

Intermediate indicators		
Negative	**Positive**	**Discriminating**
iINCollaborateDiscussion	iIPCollaborateDiscussion	iIDCollaborateDiscussion
iINShareCommunity	iIPShareCommunity	iIDShareCommunity
iINSelectFromOther	iIPSelectFromOther	iIDSelectFromOther
iINCollect	iIPCollect	iIDCollect
iINRateHigh	iIPRateHigh	iIDRateHigh
iINRecommend	iIPRecommend	iIDRecommend
iINChange	iIPChange	iIDChange
iINChangePerson	iIPChangePerson	iIDChangePerson
iINOrganizeCollection	iIPOrganizeCollection	iIDOrganizeCollection
iINChangeByAddOrDelete	iIPChangeByAddOrDelete	iIDChangeByAddOrDelete
iINPresentAudience	iIPPresentAudience	iIDPresentAudience
iINGotRated	iIPGotRated	iIDGotRated
iINTag	iIPTag	iIDTag
iINViewPerson	iIPViewPerson	iIDViewPerson
iINCollectSimilar	iIPCollectSimilar	iIDCollectSimilar
iINView	iIPView	iIDView
iINMadeOutOfOthers	iIPMadeOutOfOthers	iIDMadeOutOfOthers
iINCollaborateCollection	iIPCollaborateCollection	iIDCollaborateCollection
iINSearchResults	iIPSearchResults	iIDSearchResults

iINReferredBy	iIPReferredBy	iIDReferredBy
iINAsses	iIPAsses	iIDAsses
iINDiscuss	iIPDiscuss	iIDDiscuss
iINAssociatePerson	iIPAssociatePerson	iIDAssociatePerson
iINChangeType	iIPChangeType	iIDChangeType
iINStandard		iIDStandard
iINUseWide		iIDUseWide
iINAware		iIDAware

Table 5: Intermediate Indicators

It will be reasonable to calculate II in order to be able to do a mapping between TI and KMI as the specification of respective UE groups was sought to be too detailed in order to be mapped to KMI directly. The actual mapping of II instances to the set of chosen KMI gets shown in table 6. As II could be mapped differently to different KMI in different POLME, the mapping was done considering DS1 as POLME actually used within this work.

Mapping of intermediate indicators to knowledge maturing indicators	
Knowledge maturing indicators	**Intermediate indicators**
An artifact is used widely	iIDUseWide
An artifact was changed	iIDChange
	iIDChangeByAddOrDelete
An artifact has been accepted into a restricted domain	iIDCollect
	iIDPresentAudience
An artifact has been the subject of many discussions	iIDDiscuss
An artifact has been accessed by a different group of individuals	iIDCollaborateCollection
	iIDCollaborateDiscussion
	iIDChangePerson
	iIDViewPerson
An artifact has achieved a high degree of awareness among others	iIDAware
	iIDViewPerson
An artifact has been recommended or approved by management	iIDRecommend
An artifact has been used by an individual	iIDAssociatePerson
An artifact has been rated high	iIDRateHigh
An artifact was made accessible to a different group of individuals	iIDShareCommunity

An artifact was selected from a range of artifacts	iIDSelectFromOther
	iIDViewPerson
An artifact became part of a collection of similar artifacts	iIDCollectSimilar
An artifact was presented to an influential group of individuals (An artifact was prepared for a meeting)	iIDPresentAudience
An artifact has been assessed by an individual	iIDAsses
	iIDGotRated
	iIDOrganizeCollection
An artifact is referred to by another artifact	iIDReferredBy
An artifact was created by integrating parts of other artifacts	iIDMadeOutOfOthers
An artifact has become part of a guideline or has become standard	iIDStandard
An artifact was changed in type	iIDChangeType
An artifact has been created/edited/co-developed by a diverse group	iIDChangePerson
	iIDOrganizeCollection
	iIDCollaborateCollection
	iIDCollaborateDiscussion

Table 6: Mapping of Intermediate Indicators to Knowledge Maturing Indicators

The mapping between II and KMI considers the actual resource type, the respective UE group underlying the calculation of II as well as the possible assignment of to calculate KMI to one or more II.

So far, underlying conceptual thoughts concerning the calculation of KMI for resources got described. The next chapters will focus on technical considerations with regard to the implementations for the calculation of KMI.

4.2.3 Technical

Apart from this work's target, an earlier version of DS1 got evaluated during development phase. Respective evaluation shall be described shortly in order to lay out important constraints and requirements for an adequate implementation and application of KMI for resources before it will be explained how the result of this work will be provided by KMS and underlying RM.

4.2.3.1 Performance Evaluation

During an earlier development phase not being part of this work, DS1 got evaluated at the 'University of Innsbruck'[10] yielding relevant constraints and requirements for this work. The focus of respective evaluation lied on observing the performance of DS1 considering its general performance with regard to connection speeds and usability issues, server reaction times to client queries and the up- and download speed of local files put into collections. For this work, mainly server reaction times and general connection speeds became relevant.

The evaluation was done during two classes of an information technology course with 25 participants split in two groups. Each group performed certain tasks with the help of DS1 at the same time, tough the tasks were differing. After the half of time spent with DS1 tasks got interchanged between the groups. The tasks primarily were made up of requirements considering certain functionalities of DS1 in order to find resources for a certain topic, file them in collections and collaboratively work on the content put into the system. Additionally, the students were asked to label resources with various tags and perform a number of searches within the system in order to retrieve content put in by their colleagues. Lastly, the users had to start and develop discussions upon their collected materials with users of the other group respectively.

For this work, it was important to see that the actual implementation of the communication between client and server was not satisfying enough in order to rely on used technology for further implementations in the realm of this work. In order to have a defined area wherein to measure communication capabilities of DS1, file up- and download was chosen. The evaluation yielded that times for uploading files were too high in order to deploy the system in a real world setting like that at CK considering the bandwidth of the intranet at CK being much slower than at the 'University of Innsbruck'. The vast amount of time the client needed to up- and download collection entries to the server could be traced back to the amount of additional data sent needed for 'Extensible Markup Language' (XML)[11]-based 'Simple Object Access Protocol' (SOAP)[12] backed calls to currently used 'Webservices' (WS)[13] implemented by an 'Apache Axis' (AA)[14] framework on server side. For uploading one single file additionally about one third of the file's size was needed in order to exchange synchronization

[10] http://www.uibk.ac.at/
[11] http://www.w3.org/XML/
[12] http://www.w3.org/TR/soap/
[13] http://www.w3.org/2002/ws/
[14] http://axis.apache.org/axis/

data for the SOAP call itself. As at a certain point in time more than at least five students were actively working with the system, DS1 wasn't able to handle all users' queries in a sound time span. Additionally, the actual way of implementation within the client part of DS1 didn't reuse data already retrieved from server for certain queries but performed a new outgoing request although data could have been available already. Hence, the amount of data sent between client and server actually at least doubled, which made the system even slower.

As this work had to be integrated into DS1 using the same connection for handling client server communication, it was necessary to replace the WS-based communication by a more sophisticated approach. Moreover, it was badly necessary to implement client components being able to cache certain data retrieved from server in order to avoid unnecessary data exchange. A short manually performed code review yielded that broad refactoring of client code was needed in order to integrate planned functionalities of this work.

4.2.3.2 Technical Preconditions

This work uses both client and server side of DS1 in order to implement / integrate the calculation of KMI and their respective presentation in DS1. From that, the basic technical environment given shall be described in the following. The client application of DS1 as well as the server side got implemented with regard to basic requirements from within the 'MATURE' project for CK as application partner on the one hand and for SA for evaluation purposes of the POLME approach on the other hand.

4.2.3.2.1 Client

The client side of DS1 got implemented with the help of 'Adobe'[15] 'Flex'[16] 'Software Development Kit' (SDK)[17] 4.0. 'Flex' is made out of a combination of 'Actionscript'[18] and 'Macromedia eXtensible Markup Language' (MXML)[19]. MXML is a declarative markup language[20] used for describing GUI in rich internet applications. This combination made it somewhat easier to separate the actual implementation of the GUI from components responsible, first, for handling the data displayed and second, for the processing of data as well as for the management of server requests and results. From that, it was reasonable to stay with this development kit and implement upcoming changes with the help of these tools.

[15] http://www.adobe.com/
[16] http://flex.org/
[17] http://en.wikipedia.org/wiki/Software_development_kit
[18] http://actionscript.org/
[19] http://en.wikipedia.org/wiki/MXML
[20] http://en.wikipedia.org/wiki/Markup_Language

Another reason to stay with 'Flex' for client side implementations was the very sophisticated support of 'Sockets' [21] which were sought to replace slower WS implementations. Additionally, it was not sure at this point in time whether the widget-based approach of DS1 should be fostered or changed to a web-based implementation. First, 'Flex' provides low level 'Socket' interfaces which were ready to use from within the SDK. Second, 'Flex' is either usable as 'Air' [22] application running as desktop application or as web-based application executed with the help of 'Flash Player' [23] with almost no changes necessary.

From that, 'Flex' provided most opportunities to keep the widget-based approach as well as to support a faster low level client server communication regardless of which application scenario would be strived for in future.

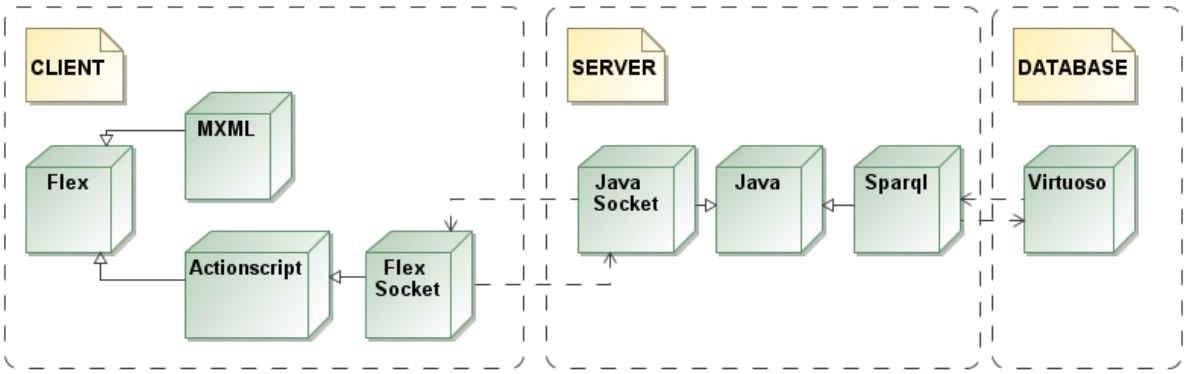

Figure 11: Technical environment

4.2.3.2.2 Server

The server side of DS1 got implemented in 'Java' [24] with the help of the AA WS framework for the communication with client side. Reasons to use WS were seen in the ease of use by provision of dedicated support by the AA framework as well as the possibility to use loosely coupled arrangements of services for clients' needs. The connection from within the 'Webserver' [25] to the DB represented by a TS implemented by a 'Virtuoso' server (VS) [26] was written in 'Java' too. To query the content stored in the TS, SPARQL [27] got used in order to set off statements to the DB server instance.

[21] http://en.wikipedia.org/wiki/Network_socket
[22] http://www.adobe.com/products/air.html
[23] http://www.adobe.com/products/flashplayer.html
[24] http://java.com/en/
[25] http://en.wikipedia.org/wiki/Web_server
[26] http://virtuoso.openlinksw.com/
[27] http://www.w3.org/TR/rdf-sparql-query/

From the current technical situation, the server side implementation of to come changes and implementations for this work had to be done in 'Java', though it was planned to replace WS implementations by 'Sockets' in order to solve performance issues. The overall situation of tools which will be actually used gets shown in figure 11.

As the performance evaluation has shown, current implementation with regard to reaction times was not able to allow any further load on already overstrained communication channel. From that, it became necessary to remove currently used WS communication and replace it by a lower level communication which was sought in a 'Socket' implementation. After some initial tests considering connection speeds and ease of use, the 'Socket' approach was sought to be implemented. The reason not to use any other technology for client server communication was seen in the fact that all other communication strategies from within the 'Flex' SDK again would have been using 'Hypertext Transfer Protocol' (HTTP)[28]-based communication. As WS failed (as described above) any similar approach has not been reasonable. Another reason for not to use any other 'Flex'-related technology like 'Action Message Format' (AMF)[29] for 'Hypertext Preprocessor' (PHP)[30], i.e. 'amfPHP'[31] was derived from the fact that such kinds of communication strategies would have required certain types of servers, following broad changes necessary on server side. In the case of, e.g. amfPHP, a PHP server would have been required; another way taking into account using given 'Java' on server side from within 'Flex' would have required, e.g. a 'BlazeDS'[32] server instance. As the initial requirement was to use 'Java' on the server side, the 'Socket' implementation seemed to be most appropriate and reasonable according to necessary changes.

4.2.3.3 Knowledge Maturing Services and Resource Model

As all the UE to be used for the calculation process of KMI for resources have to be retrieved from within DS1, the ULKMS has to be provided accepting given UE and permanently storing them in order to be accessible later on. Respective service is seen as 'Representation' service following theory from within the realm of 'Knowledge Maturing'. Actually implemented as 'Usage' service, this service is responsible for gathering user interaction data,

[28] http://www.w3.org/Protocols/
[29] http://en.wikipedia.org/wiki/Action_Message_Format
[30] http://php.net/
[31] http://www.silexlabs.org/amfphp/
[32] http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

i.e. data that is generated through the use of and interaction with a POLME and its contained knowledge artifacts.

The actual process of calculating KMI shall be handled within a ‘Modeling’ service following theory. In the case of this work, the RMKMS gets introduced to characterize each knowledge artifact (resource) in a POLME with the help of key-value pairs. As the ‘Knowledge Maturing Process’ is most intuitively recognized in the case of content objects, the RMKMS is seen as an abstraction service providing contextual information about resources [Kump et al., 2011].

To keep all properties of resources represented by key-value pairs (including calculated KMI) accessible, this work follows the idea of RP. As RP can be used to represent (digital) resources with regard to various additional (meta-) data, the actual container keeping together all the RP for resources within DS1 will be denoted as RM. As such, the RM represents the core container of resources within the RMKMS. To keep the RM updated for fast access of resource information, a working memory-based approach will be chosen. It is sought to store calculated characteristics of resources in server’s disk memory later on. The RM has to provide defined methods to, first, handle updates and second, query it for certain resources according to DS1’s needs.

In order to update RP contained in the RM on a regularly basis, it was decided to automatically build an updated RM instance once a night. The decision not to update the RM each time new UE could be used in order to re-calculate defined KMI came from possible performance loss of server’s processing power.

4.3 Implementation

After describing the setting of this work together with conceptual thoughts and technical preconditions, concrete implementations done for the calculation process of KMI for resources will be described in the following.

4.3.1 Client Side User Event Collection

The calculation of KMI builds on TI generated from within desired POLME. From that, it was necessary to, first, analyze DS1 with regard to available TI defining the basic entities for the overall KMI calculation process and second, develop a framework which is able to trace and send respective TI to the server side in order to get stored accordingly. As this work gets applied to a specific POLME, possible user operations available through the GUI of DS1 got

analyzed with regard to the resource type they consider. In the following, the actual framework implementing handling TI aggregation and sending UE to server will be shown. As various implementations in DS1 had to be done enabling including the framework and using the ULKMS, changes in DS1 will be described as well.

4.3.1.1 Mature Components Library

The evaluation of DS1 yielded that some of the components used within the client side of DS1 had to be changed and extended. In order to realize those upcoming changes, it was sought to centralize various server calling methods and distribution of their results into a single component outsourced into a library. Moreover, redundant code fragments in DS1 got moved to the library, hence were made available for all code parts at a single code location.

The design of the MC library got laid out with regard to the need for tracing and sending defined UE to the server part of DS1. Therefore, first it was badly necessary to introduce the new client server communication approach in the form of 'Socket' communication. Furthermore, the code in DS1 had to be cleaned up considering redundant implementation fragments. MC allowed for removing all methods and corresponding result handlers needed for the current client server communication directly implemented into DS1; respective code got re-implemented and relocated into dedicated places in MC. MC from then on was made responsible for handling all outgoing server requests and respective results.

Additionally, MC implemented caching possibilities needed to make certain client side data available for faster access. As the same infrastructure got reused in other projects independently from this work, MC had to be implemented with regard to defined interfaces through which data could be accessed and distributed.

To already get an idea of how the library could be used, figure 12 shows most important classes implemented within the library. The basic entities of MC get represented by class 'GlobalConstants' containing instances of library interfaces, class 'ServerManager' for delegating server querying and updating to sub-managers, classes 'ServerCallingManager' and 'ServerResponseParserManager' for handling the actual calls and results of server requests and responses, caching classes used in certain managers, buffering classes for latching server call parameters and classes representing server result instances. A detailed description of classes and the basic workflow within the library will described subsequently after changes allowing using the library from within DS1 will be described in the following.

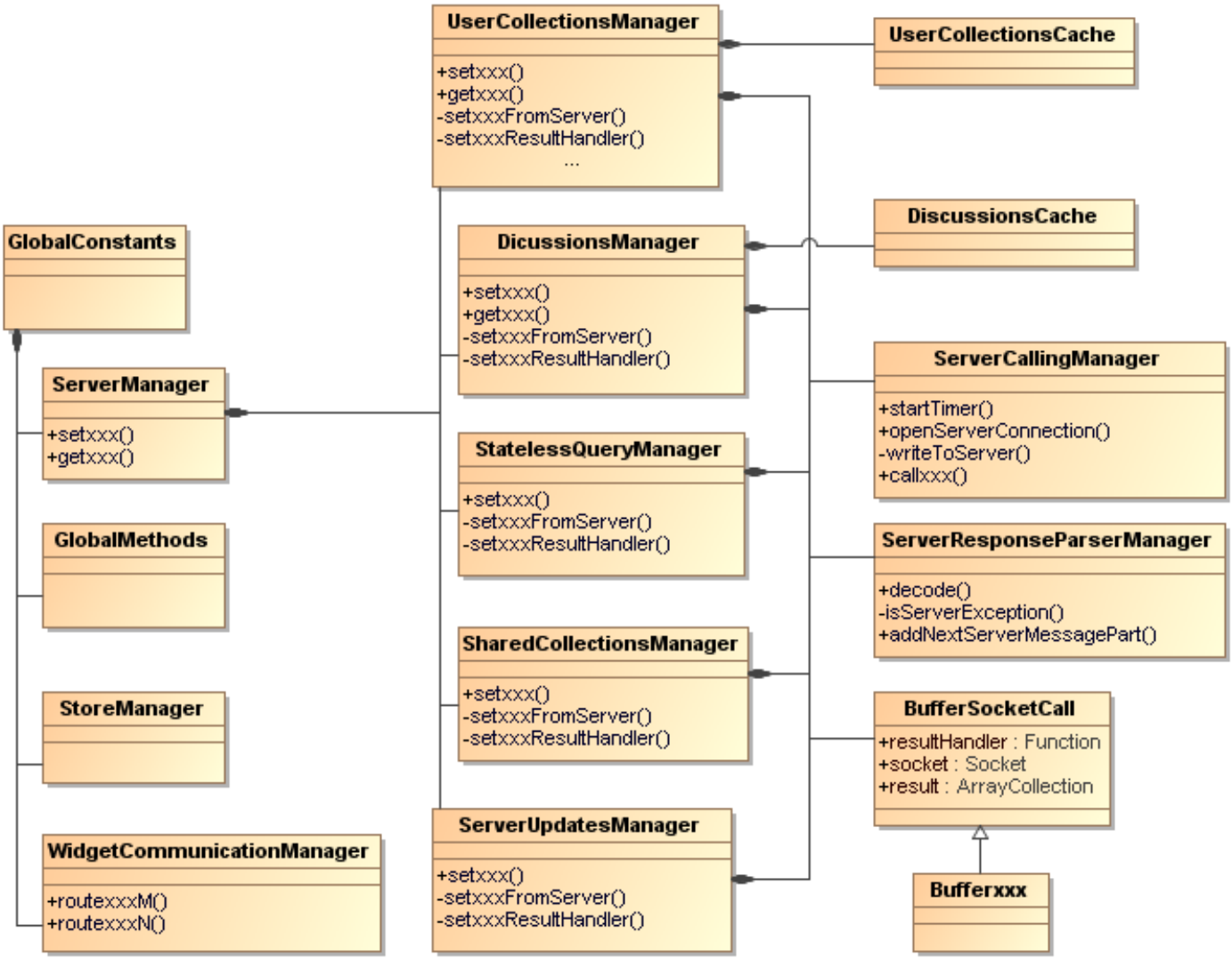

Figure 12: Mature Components Library Classes

4.3.1.2 Changes in Demonstrator 1

As both the performance evaluation described above and initial code analysis yielded, extensive DS1 client side implementations had to be done in order to set up the environment capable of handling TI collection and sending to server.

4.3.1.2.1 Socket Communication

At the time MC got introduced all server requests and updates got handled directly from within various places in DS1 code. A certain WS declaration in each code file where server access was needed was defined in the following way:

```
<fx:Declarations>
      <s:WebService id="wsTaxonomy" wsdl="xxx?wsdl">
            <s:operation
```

```
                    name="interpretKeywordQuery"
                    resultFormat="object"
                    result="keywordQueryResultHandler(event)"
                    fault="wsTaxonomyFaultHandler(event)"
                />
        </s:WebService>
</fx:Declarations>
```

In order to use a certain operation of the service, the service had to be called the following way:

```
wsTaxonomy.interpretKeywordQuery(„someKeyword“);
```

When the asynchronous call to the server returned, either the defined result or fault handler got called by the ‘Flex’ framework. From there, the result in a defined format had to be parsed for further processing. The actual structure of operations defined in underlying ‘Web Services Description Language’ (WSDL)[33] files and implemented counterparts on server side often required a number of subsequent calls to get all properties of a certain object. For example, a first request retrieved all the URI for existing collections; a second one was able to retrieve all the URI of a certain collection’s entries. To retrieve certain properties of a collection item another call was necessary. From that, the structure of WS operations and their result handlers got very complex, beside the fact that the parameters for WS operations were not defined directly in WS declaration in MXML but could only be found in the underlying WSDL file.

Re-factoring

To not only introduce a new communication strategy but to be able to consider possible side effects in DS1, DS1 code got re-factored in order to extract code parts depending on the results of client server communication. For this, code depending on server results was brought as close as possible to the result handler of the WS operation call first. Code locations for handling the results of server requests often were based at different locations due to the fact that a lot of event dispatching and handling was used for the distribution of results. This strategy was mainly followed because widgets themselves actually got implemented very modularly. On the one hand, this made the code very readable concerning the graphical layout of items it implemented, on the other hand this often split data handling processes which should have stayed together. Hence, complex and confusing event handling processes got

[33] http://www.w3.org/TR/wsdl

partly removed and replaced by appropriate result handling at places where asynchronous server calls returned. From that, it was easier and now possible to apply necessary new communication.

Introducing New Data Types

The next step in the changing process of the client side of DS1 considered the actual data which got requested or updated from within DS1. As mentioned above, often data requested by a single WS operation didn't suffice to retrieve all data necessary at a certain point in time. In order to help overcome this, a new set of data types got introduced in combination with new server requests allowing for a more sophisticated use of resources.

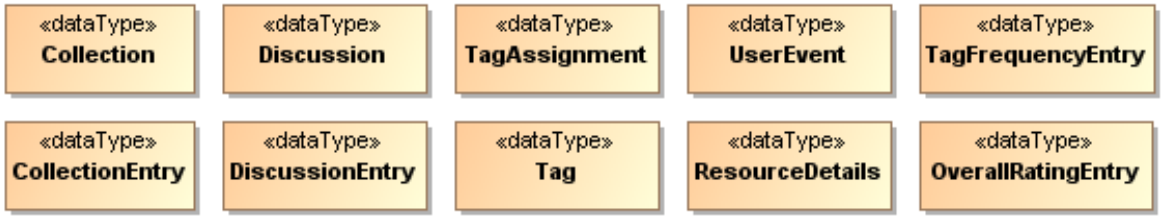

Figure 13: New Data Types

The new set of entities shown in figure 13 got developed considering actual server side representations of entities. As not all new data types introduced could be mapped directly to currently used data type instances on server, certain accommodations according to the handling of in- and output data types for new communication between client and server had to be done as well. From the client side perspective, new data types were able to represent entities the user got to deal with, hence using respective data types for communication between client and server yielded reducing server calls in order to get desired data. For this, the calling structure and distribution of results had to be changed in order to use less calls. Moreover, data binding for graphical elements had to be adapted.

After re-factoring the client code considering both the consolidation of split code parts and outsourcing of redundant sections into the MC library code for dealing with WS completely got replaced by using 'Sockets'.

Introducing Sockets

It was planned to keep implementation of server requests as simple as possible from within DS1. Therefore, requesting data from the server or calling an updating method now only requires a dedicated result handler (if necessary at all) and parameters important for instantiating the server call. A simple requesting operation looks as follows:

```
GlobalConstants.server.setServerTime();
```

In this example no result handler is needed, because corresponding manager in MC is able to set the result to a dedicated variable within MC. Another exemplary requesting operation for retrieving all collections within the system looks as follows:

```
GlobalConstants.server.setUserCollections(resultHandler);
```

The result handler in this case has to be defined the following way within DS1:

```
function resultHandler(collections:ArrayCollection){…}
```

The result handler expects a list of objects of class 'Collection'. From this parameter definition, it becomes clear that the client code has to know which type of parameter is required in order to catch the result of the server call. For this, various server calls implemented got documented with the help of 'ASDoc'[34] in class 'ServerManager' in order to describe what certain method expect as input parameters as well as what they return.

In the case of method 'setUserCollections', DS1 is able to retrieve requested result instantly as well as is able to access the cached result by calling corresponding methods on class 'ServerManager'.

```
GlobalConstants.server.getUserCollections();
```

This possibility allows for retrieving cached data from within MC. Therefore, operations retrieving anything from the server directly got denoted in the form of 'setxxx' whereas methods retrieving certain data from MC caches have to be called by 'getxxx'.

To provide an example for updating the server following method should give a hint how it looks like:

```
GlobalConstants.server.addUserCollection(resultHandler, collectionName,
      collectionType);
```

[34] http://livedocs.adobe.com/flex/3/html/help.html?content=asdoc_1.html

Above method requires two additional parameters ‘collectionName’ and ‘collectionType’ representing the collection’s name to be created and its type (private or public).

The code structure within DS1 could be kept very simple, as these kinds of methods and additional result handlers basically represent the whole client code necessary for requesting and updating the server side.

4.3.1.2.2 Usage Logging Knowledge Maturing Service

DS1 got analyzed very well in order to locate available TI. To be able to use those defined UE within server side, sending various UE had to be integrated directly into the client code where certain TI take place. As not all calls to the ULKMS for various UE could have been realized together within respective server calls within MC, UE handling got implemented independently of other methods considering standard server requesting and updating functionalities. Though, the basis for handling UE got laid down by new ‘Socket’ approach. From that, UE handling could be done similarly to already described server operations:

```
GlobalConstants.server.saveUserEventxxx(resource, content);
```

Initially, it was planned to provide just one method with an additional parameter denominating a certain UE type for sending it to server. In order to keep the call as simple and meaningful as appropriate, each UE gets stored by an extra method implemented in MC. Additionally, this approach simplifies identifying places in DS1 where specified TI can be triggered by GUI interaction.

For the calculation of certain KMI, additional UE had to be implemented allowing a more sophisticated calculation upon UE data, e.g.:

```
GlobalConstants.server.saveUserEventStructurePrivateCollectionItemStructure
      PrivateCollection(resource, content);
```

For defined UE often more than one event had to be saved at a certain occasion, so that UE handling for certain TI got packed into dedicated methods in DS1, e.g.:

```
private function saveUserEventStructureCollectionContent(…){…};
```

4.3.1.3 Mature Components Library Implementations

After changes necessary directly in DS1 code got described, this chapter shows the actual core functionalities of the MC library implementing ‘Socket’ communication, caching of client side data and calling the ULKMS on server side.

The overall class structure shown in figure 12 already can give a hint where dedicated parts for the ‘Socket’ communication strategy used for sending UE to server got implemented. The overall work flow implemented in the library for dealing with DS1 server requests can be shown in figure 14.

In the following, most important classes used in DS1 shall be described briefly before the internal handling of DS1 client requests to server with regard to the sequence diagram shown in figure 14 will be described.

GlobalConstants

This class contains all necessary instances of classes responsible for working off server communication duties and globally needed constants. Globally accessible parts of the library get offered in a static way. ‘GlobalConstants’’ properties get initialized by calling ‘GlobalConstants.initializeGlobalConstants();’ at the beginning of respective client program using the library. This initializes all class variables being accessible from outside the library. Following class instances can be accessed via class ‘GlobalConstants’: ‘GlobalMethodManager’, ‘ServerManager’, ‘WidgetCommunicationManager’ and ‘StoreManager’. As the ‘Flex’ SDK neither allows private constructors nor more than one constructor per class, no true ‘Singleton’[35] pattern could have been implemented preventing the possible existence of more than just one instance of ‘GlobalConstants’ and certain other entities in the library. From that, it seemed most appropriate to use globally accessible static properties to access kinds of ‘Singleton’ objects. As ‘Flex’ does not provide a way to implement concurrency[36], it seems apparent to use ‘GlobalConstants’ as a static entry point providing central access to the library from this perspective too.

[35] http://en.wikipedia.org/wiki/Singleton_pattern
[36] http://en.wikipedia.org/wiki/Concurrency_(computer_science)

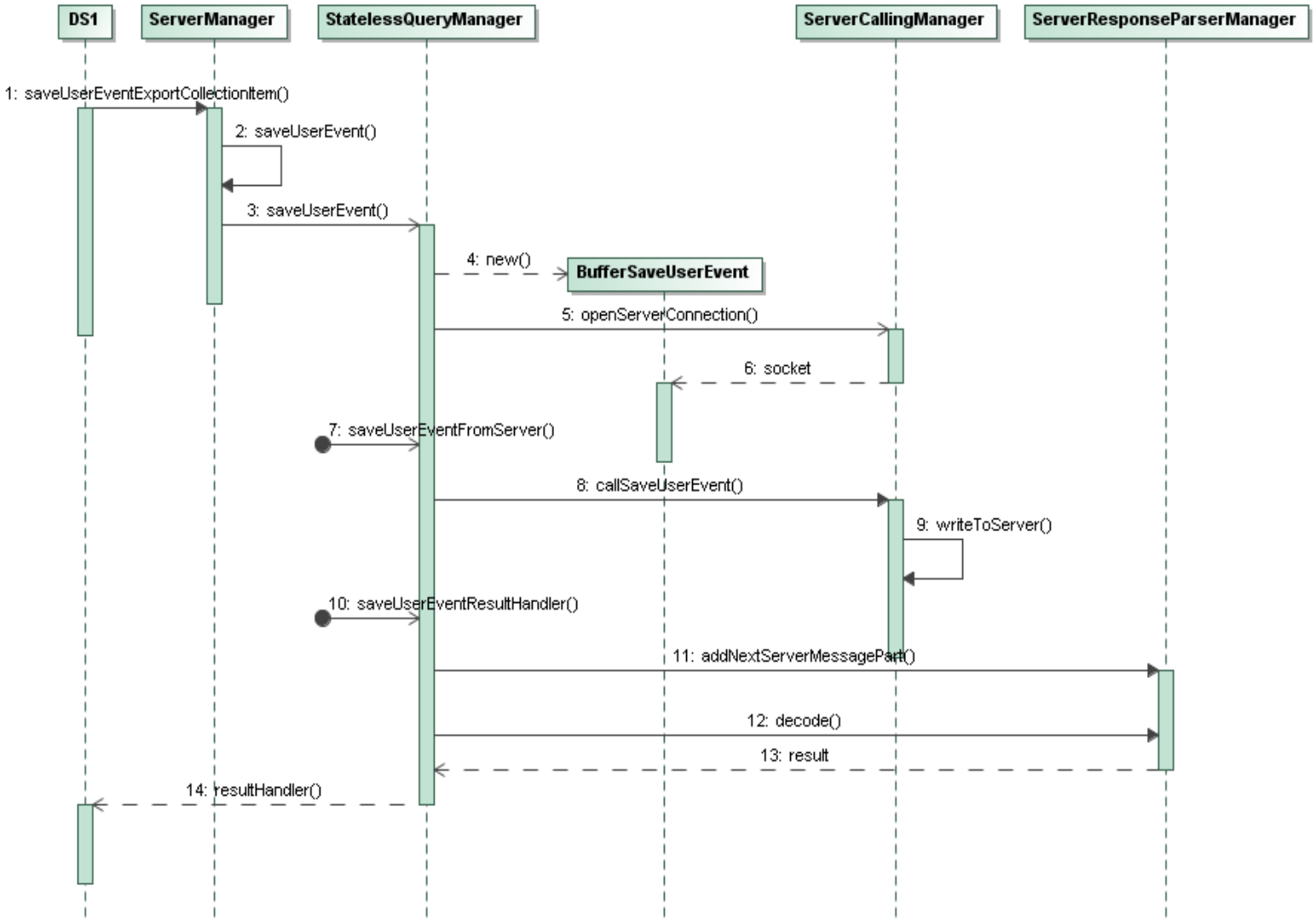

Figure 14: Mature Components Sequence for Sending User Events

ServerManager

From the class diagram of MC in figure 12 it becomes visible that all interfaces for server requests got available by an instance of class 'ServerManager'. To provide a means to delegate specific work to other class instances within the library, 'ServerManager' subsequently forwards received server calls to registered managers in order to handle certain responsibilities. 'ServerManager' basically contains following managers working off assigned tasks: 'StatelessQueryManager', 'ServerUpdatesManager', 'UserCollectionsManager', 'SharedCollectionsManager' and 'DiscussionsManager'.

As caching got implemented, only 'ServerUpdatesManager' and 'StatelessQueryManager' need to request the server each time specific data is requested from within the client. 'UserCollectionsManager', SharedCollectionsManger and 'DiscussionsManager' are able to

cache retrieved data; hence only need to call server functionality when data gets updated by the client and new data from server is required.

StatelessQueryManager

This class is responsible for server queries and updates when it's neither needed nor possible (reasonable) to cache data. Basic functionalities assigned to this class consider user log in, tagging of resources, providing basic / specific information upon resources, searching for resources, file up- and download, rating of resources and storing of UE.

ServerUpdatesManager

This class is responsible for keeping the client informed about certain changes on specific data on server.

UserCollectionsManager

Class 'UserCollectionsManager' is responsible for queries and updates concerning collections and their entries. As the user most time basically works with the same collection objects during a single session within DS1, initial data queried from server gets cached and updated locally respectively by certain properties with this class. Additionally, only changes made to collection and collection entry objects get sent to server.

SharedCollectionsManager

This class is responsible for querying all the public collections in the system providing it to the client program.

DiscussionsManager

This class handles data concerning discussions the same way as the 'UserCollectionsManager' instance does for collections belonging to the user.

Each manager instance kept by 'ServerManager' in turn contains an instance of 'ServerResponseParserManager' and 'ServerCallingManager'.

ServerCallingManager

This class provides methods getting called from within various managers in the library to perform outgoing requests to the server.

ServerResponseParserManager

This class is responsible for parsing all the results of various server calls in order to provide defined objects to its parent managers which subsequently forward results to the client program.

UserCollectionsCache and DiscussionsCache

These classes are responsible for caching data concerning collections, collection entries, discussions and discussion entries (comments). Instances of this class are held by the 'UserCollectionsManager' and by 'DiscussionsManager' respectively filling these caches with results of according server calls. The content of 'UserCollectionsCache' and 'DiscussionCache' gets queried and updated by 'UserCollectionsManager' and 'DiscussionsManager' respectively in order to access data related to collections and discussions. Several methods are provided in each class for simpler access to object's concerned properties.

BufferSocketCall

From this class derived subclasses get used to temporally store parameters and method references necessary for asynchronous 'Socket' calls. For each duty (method) of a dedicated manager a corresponding buffer class exists, providing an initial set of properties implemented by the base class and a set of parameters specific to the actual type of server call implemented by the deriving buffer class.

Data Entity Classes

Several classes shown in figure 13 get used to represent certain entities like collections or tags respectively. Instances of those classes mainly get constructed by class 'ServerResponseParserManager' as well as in the client program of DS1 when handling data to be shown in the GUI.

GlobalMethods

Methods of this class are directly available for the client program. This class defines helper methods which get used in the client program as well as in the library.

WidgetCommunicationManager

This class provides a means to send updating messages within the client program. It gets used either to inform parts of the software about changes done in various locations or to initiate

certain processes within the client which otherwise would have to be started either directly via certain method calls or dispatching and catching certain workflow event types.

StoreManager

This class is responsible for storing and retrieving certain data which has to be encrypted and stored on the client's computer.

In the following, steps necessary for dealing with client server communications can be shown.

4.3.1.3.1 Socket Communication

Class 'ServerManager' defines the overall interface for using any server functionality from outside MC. From that, each method implemented by this class is available by property 'server' in class 'GlobalConstants' and exemplarily got defined as follows:

```
public function addUserCollection(resultHandler:Function,
      collectionName:String, collectionType:String){…}
```

From within certain methods, 'ServerManager' decides which registered manager class is responsible for handling a certain request. In the case of the above example, class 'UserCollectionsManager' is responsible for handling the update of collection objects which looks as follows:

```
userCollectionsManager.addUserCollection(resultHandler:Function,
      newCollectionName:String,newCollectionType:String);
```

The body of each defined method in class 'ServerManager' basically forwards the request or update to a dedicated manager instance held by 'ServerManager'.

Forwarding Demonstrator 1 Calls to Manager Instances

In order to handle calls forwarded by class 'ServerManager', each manager instance basically defines three methods; one public method for accepting and handling the forwarded call and two private methods – one for delegating the invocation of server functionality and one handling the result sent from server. To stay with the example from above, corresponding public method in class 'UserCollectionsManager' looks as follows:

```
public function addUserCollection(resultHandler:Function,
      newCollectionName:String, newCollectionType:String):void{

      if(bufferAddUserCollection.result != null){
            caller.startTimer(waitForTriggerTimerHandler);
      }else{
            bufferAddUserCollection =
                  new BufferAddUserCollection(
                        GlobalConstants.SM_ADD_USER_COLLECTION,
                        new ArrayCollection(),
                        resultHandler,
                        caller.openServerConnection(
                              addUserCollectionFromServer,
                              addUserCollectionResultHandler),
                        newCollectionName,
                        newCollectionType);
      }

      function waitForTriggerTimerHandler(event:TimerEvent):void{

            if(bufferAddUserCollection.result != null){
                  caller.startTimer(waitForTriggerTimerHandler);
            }else{
                  addUserCollection(
                        resultHandler,
                        newCollectionName,
                        newCollectionType);
            }
      }
}
```

The integrated ‘Socket’ framework in ‘Flex’ allows for ‘Asynchronous Communication’[37] with the server. From that, the connection between client and server has to be initialized first in order to get used for exchanging data. Hence, it becomes apparent that it is possible to call, e.g. method ‘addUserCollection’ twice even before the first call returned from server, although explicit concurrency is not available within ‘Flex’. To prevent sending second request’s data over the communication cannel opened for first call, a blocking mechanism got implemented considering these circumstances.

[37] http://en.wikipedia.org/wiki/Asynchronous_communication

Call Sequence Handling

The mechanism for taking care of possibly intervening calls for a certain method in a manager instance uses buffers, first, to observe calls preformed and not returned yet and second, to store data and parameters needed for a server call. Hence, for each public method defined in a manager class a related buffer instance has to be defined. For exemplary method 'addUserCollection' corresponding buffer class 'BufferAddUserCollection' gets derived from 'BufferSocketCall'. Class 'BufferSocketCall' basically defines properties which get used by all derived buffer classes in order to store the result handler provided by DS1, the initialized 'Socket' instance and a data array for handling server results. The latter property is responsible for observing whether a server call gets currently performed. By questioning whether the result property of the buffer is null at the begin of method 'addUserCollection', it gets decided whether a possible second call to this method should be performed instantly or scheduled to a later point in time after the first call will have finished.

```
if(bufferAddUserCollection.result != null){
      caller.startTimer(waitForTriggerTimerHandler);
}else{…}
```

If the second call can't be performed instantly, a timer gets started in order to wait for the first call to be finished (when the server pushes data to the client). The else part of corresponding if clause actually performs the request. The method 'startTimer' implemented in class 'ServerCallingManager' uses defined 'Anonymous Function'[38] as parameter handling the timer's expiration.

```
if(bufferAddUserCollection.result != null){
      caller.startTimer(waitForTriggerTimerHandler);
}else{
      addUserCollection(
            resultHandler,
            newCollectionName,
            newCollectionType);
}
```

[38] http://en.wikipedia.org/wiki/Anonymous_function

The timer handler checks whether the method is ready to be called by the second ‘thread’ or has to be set back in time. The result property of the buffer instance only gets set to null at the time the server returns for the first call and result handling got worked off respectively.

Initializing Parameter and Result Handler Storage

Data needed for the server request has to be buffered in respective buffer property of corresponding manager class (which was already used for polling a server call) in order to be able to provide it after the initialization sequence for the ‘Socket’ call completed. Therefore, the buffer gets filled as follows according to ongoing example:

```
bufferAddUserCollection =
      new BufferAddUserCollection(
            new ArrayCollection(),
            resultHandler,
            GlobalConstants.SM_ADD_USER_COLLECTION,
            caller.openServerConnection(
                  addUserCollectionFromServer,
                  addUserCollectionResultHandler),
            newCollectionName,
            newCollectionType);
```

In this concrete case, buffer class ‘BufferAddUserCollection’ contains two additional properties, i.e. label and type. From that, the parameters of the constructor become apparent; the first one provides the container for the call’s result, second one takes the result handler provided by DS1, third one gets set with the actual ‘Socket’ instance returned by method ‘openServerConnection’, the fourth and fifth one take the name and type for the new collection.

Initializing the Server Connection

Class ‘ServerCallingManager’ is responsible for functionalities considering actual communication with the server via the ‘Socket’ class implemented by ‘Flex’ SDK. In the following, provided method body shows how the ‘Socket’ gets created and how the dedicated handler which gets called after the connection with the server got established gets assigned to the ‘Socket’ instance. Moreover, it gets shown how the result handler for receiving data from server gets assigned to the ‘Socket’ and necessary server address and port get provided.

```
public function openServerConnection(connectHandler:Function,
      resultHandler:Function):Socket{

      var socket:Socket =  new Socket();
      socket.addEventListener(Event.CONNECT,connectHandler);
      socket.addEventListener(ProgressEvent.SOCKET_DATA,resultHandler);
      […]
      socket.connect(GlobalConstants.SOCIAL_SERVER_ADDRESS,GlobalConstants.
            PORT_SOCIAL_SERVER);
      return socket;
     }
```

Returned ‘Socket’ instance gets stored in the dedicated buffer for a certain server call so that it gets accessible in the connect handler as well as in the result handler for setting off the actual call to the server and for handling the server’s return message respectively.

Sending Data to the Server

After the initializing sequence for the server connection has been finished, the ‘Socket’ returns to the defined handler wherein the actual call to the server gets sent.

```
private function addUserCollectionFromServer(event:Event):void{
      caller.callCreateCollection(bufferAddUserCollection);
}
```

The call to the server representing DS1’s actual request gets handled by a dedicated method within class ‘ServerCallingManager’. For each kind of request DS1 sets off to the server, a separate method got defined in class ‘ServerManager’ and in respective manager instance, though methods in class ‘ServerCallingManager’ could be used from within different methods in manager instances. The reason for not implementing the actual server call in originating manager itself lies in the fact that DS1’s client side perspective of certain server functionality not necessarily reflects what the server used in DS1 is actually able to do. From that, it is possible to hide parts of certain server operations’ functionalities. This allows for more tailored application of server functionalities as well as hiding certain functionality from the client.

The method provided in class ‘ServerCallingManager’ for actually handling the request to the server receives the buffer for the request containing the ‘Socket’ instance and certain parameters to be sent over already established connection provided by the ‘Socket’.

```
public function callCreateCollection(buffer:BufferSocketCall):void{
      var request:ServerRequestParameter = new ServerRequestParameter(
            buffer.methodName,4);
      request.addParameter(buffer.collectioName);
      […]
      writeToServer(socket,request);
}
```

The parameters containing requested data or identifiers for entities to be updated on server get packed into an instance of class ‘ServerRequestParameter’ representing needed container structure for transmitting and handling the request on server side. Afterwards, method ‘writeToServer’ works off pushing data to the server.

```
public function writeToServer(socket:Socket,
      request:ServerRequestParameter):void{

      socket.writeUTFBytes(JSON.encode(request));
      socket.writeByte(0);
      socket.flush();
     }
```

It was sought to use ‘JavaScript Object Notation’ (JSON)[39] encoding for representing data contained in non primitive data types from within ‘Flex’, e.g. the self defined ‘Collection’ object. To use JSON was most reasonable because of its small amount of data overhead for representing complex object structures in relation to, e.g. other encoding formats like XML.

The ‘Socket’ transmits the data according to the ‘Universal Character Set Transformation Format 8-bit’ (UTF-8)[40] standard which guarantees appropriate character encoding so that server side counterparts receive data in standardized format. To complete and signalize the end of the data stream sent to the server, an additional zero gets send.

[39] http://json.org/
[40] http://utf8.com/

Receiving Data from Server

The ‘Socket’s’ attached result handler got located in each responsible manager respectively and defines the second private method necessary for each server call in respective manager. The basic structure of a method handling the server result looks as follows:

```
private function addUserCollectionResultHandler(event:ProgressEvent):void{
      if(responder.addNextServerMessagePart(buffer) == true){
            var result:Object = responder.decode(buffer.result.getItemAt(1)
                  as String) as Object;
            buffer.resultHandler();
            bufferAddUserCollection.result = null;
      }
}
```

As it cannot be expected that the server answers requested data during a single transmission, the result handler can be called more than one time. From that, method ‘addNextServerMessagePart’ with instance of class ‘ServerResponseParserManager’ adds available data to the result object contained in the buffer as long as all requested data got received. After the result is complete, instance of class ‘ServerResponseParserManager’ is responsible for decoding data described in JSON format. To complete the whole server request, the result handler provided by the client program gets called in order to either distribute the server result or to just update that a certain server call has finished.

Adding Received Data to the Result

The buffer holding data and results for a certain server call gets filled up with data received from the ‘Socket’.

```
public function addNextServerMessagePart(buffer:BufferSocketCall):Boolean{
      while (buffer.socket.bytesAvailable == true) {
            buffer.result += buffer.socket.readUTFBytes(
                  buffer.socket.bytesAvailable);
      }
      try{ JSON.decode(buffer.result);
      }catch(e:Error){ return false; }
      return true;
     }
```

To assert whether all data was already received, currently received data is tested against its JSON syntax. If the data string is complete, hence could be correctly decoded by 'Flex'-integrated JSON's decode operator, no exception will be thrown; otherwise an error suggests that further data is necessary to complete the JSON encoded result string.

Decoding Received Result

The decode method of class 'ServerResponseParserManager' is able to decode the received result string into defined data objects within MC. As it is possible that a certain error happened while the server handled the request, the result is tested against errors first. To decide whether the result contains data for a specific request, the object retrieved from an initial JSON decoding process contains a property called 'methodName' denominating the request type. The server result string gets decoded by defined methods subsequently and returned to the calling method in respective manager instance informing about the received result.

```
public function decode(messageToDecode:String):Object{
	var decodedJsonObject:Object = JSON.decode(messageToDecode);
	isServerException(decodedJsonObject);
	if(decodedJsonObject.methodName == GlobalConstants.SM_xxx){
		return decodexxx();
	}
	[…]
}
```

So far, the core functionalities according to the 'Socket' implementation in MC got described so that data caching and using the ULKMS can be described next.

4.3.1.3.2 Data Caching

MC provides the possibility to cache data for example for collections in the system. To allow for a sophisticated and reliable caching approach, data updates from DS1 have to be done on server side first to subsequently perform caching in MC. To get data from caches' instances, certain getter methods questioning caches got implemented. Following example shows how a certain cache gets initialized with data from server, gets updated and queried by DS1. To grasp for example retrieving all the collection objects within the system, caching of collection objects looks as follows implemented in the result handler for call 'setUserCollections' in class 'UserCollectionsManager'.

```
userCollectionsCache =
      new UserCollectionsCache(
            responder.decode(
                  bufferSetUserCollections.result.getItemAt(1) as String));
```

Querying Cached Data

The instance of class ‘UserCollectionsCache’ contained in class ‘UserCollectionsManager’ storing all initially received collections is able to provide certain operations dealing with extracting relevant information from the cache’s internal structure. For example, methods ‘canAddCollection’ or ‘canAddCollectionEntry’ help to check whether collections or collection entries can be added to the cache. Further, methods ‘getCollectionByUri’ and ‘getCollectionEntries’ provide a means to retrieve certain collections as well as all their entries.

To be able to query the ‘UserCollectionsCache’ instance from within DS1, methods in class ‘ServerManager’ got provided forwarding queries to ‘UserCollectionsManager’ which in turn sets off certain operations to its instance of ‘UserCollectionsCache’.

Updating Cached Data

To update certain properties of, e.g. collections, methods like ‘setCollectionEntryNewPosition’ or ‘addCollection’ got implementend in ‘UserCollectionsCache’. Method ‘addCollection’ for example gets called whenever DS1 requests to add a new collection by calling method ‘addUserCollection’ on class ‘ServerManager’; the collection gets added on server first before the cache in ‘UserCollectionsManager’ gets updated.

```
var collectionUri:String=responder.decode(bufferAddUserCollection.result);
userCollectionsCache.addCollection(
      new OurCollection(
            collectionUri,
            new Array(),
            bufferAddUserCollection.label,
            bufferAddUserCollection.type,
            GlobalConstants.USER_NAME));
```

4.3.1.3.3 Usage Logging Knowledge Maturing Service

The basis for storing UE to the server for the calculation of KMI for resources got laid down by the implementation of 'Socket' communication already. From that, it was possible to implement dedicated methods in the MC library in order to invoke the ULKMS on server side. The actual implementation of the ULKMS using the 'Socket' communication will be explained in subsequent chapters. As the basics for using the 'Socket' communication channel got already explained, it remains open how to use the end point of the ULKMS within client side. As basically whole needed server functionality is available from within class 'ServerManager', also storing of UE got integrated in this class. DS1 is able to call provided UE storing methods in class 'ServerManager' in order to send the UE to the server. Class 'ServerManager' delegates requests to class 'StatelessQueryManager' which in turn handles the UE accordingly. Similarly to descriptions above, 'StatelessQueryManager' calls designated method 'callAddUserEvent' in class 'ServerCallingManager' in order to invoke the ULKMS as follows:

```
request = new ServerRequestParameter(buffer.methodName,5);
request.addParameter(GlobalConstants.USER_NAME);
request.addParameter(buffer.eventKey);
request.addParameter(buffer.eventContent);
request.addParameter(buffer.resourceURI);
request.addParameter(GlobalConstants.USER_KEY);
writeToServer(buffer.socket,request);
```

As defined by the parameters of a UE, it is necessary to provide the identifier of the user triggering the UE, the UE type ('eventKey'), the UE content and the involved resource.

This chapter described overall implementations done for enabling the ULKMS to be called via the new 'Socket' communication. Additionally, changes necessary for tracing UE directly within the client side of DS1 got described. The next step in the development process after collecting the UE from within DS1 was the implementation of the ULKMS. Just before this service could come to life, it was necessary to implement needed counterpart for 'Socket' communication on server side described in the next chapter too. These implementations will lay down the basis for modeling resources with regard to KMI and other usage-based characteristics respectively.

4.3.2 Server Side User Event Collection

First, implementations on server side had to be done in order to replace the WS framework with components necessary for ‘Socket’ communication. Next, new methods had to be provided in order to enable the core functionality of the ULKMS, i.e. accepting and storing defined UE sent from within DS1 to the TS. As the components for calculating KMI got integrated into the existing server side framework of DS1, a short overview of the overall package structure on server side should be given first, before jumping into details of ‘Socket’ communication changes and implementation of the ULKMS.

4.3.2.1 Server Side Framework Structure

As not all parts within the server application are relevant for this work, not the whole structure of server’s packages shown in figure 15 will be explained here. The focus of server side development for ‘Socket’ implementations lies on packages ‘service’ and ‘socket’. Functionalities for the calculation of KMI for resources got implemented below the ‘model’ package in subsequent packages.

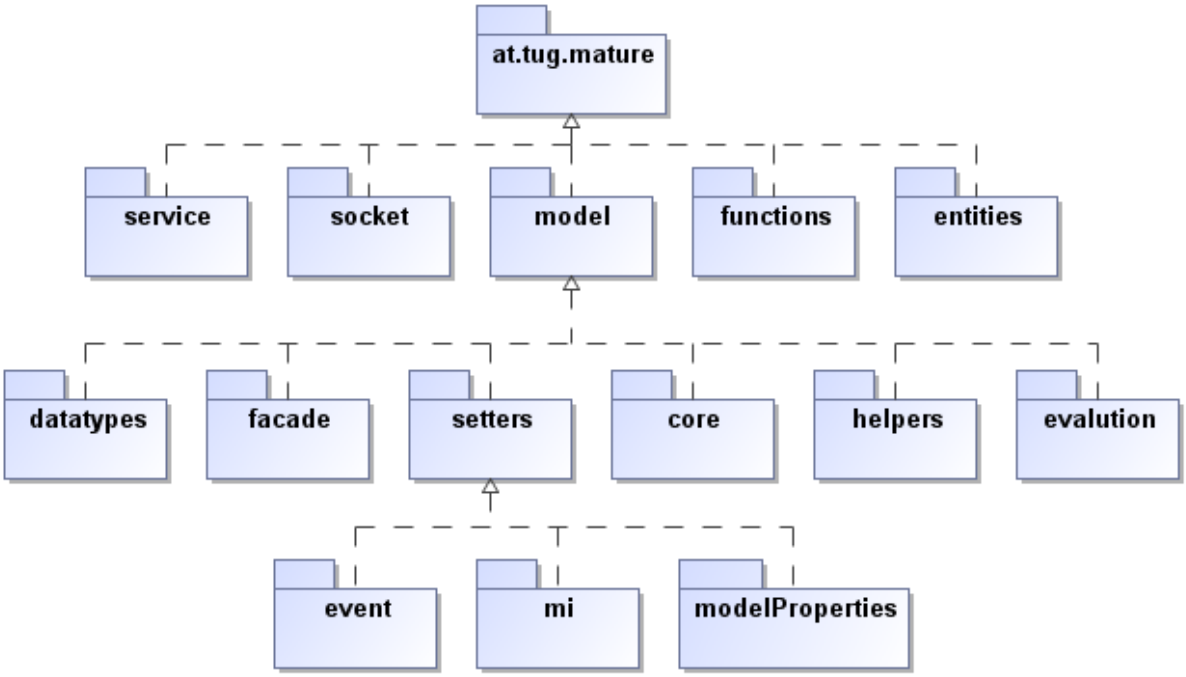

Figure 15: Server packages

Service and Socket Packages

The ‘service’ package contains class ‘SocketInterface’ which defines operations accessible from within DS1 client side through ‘Socket’ communication. Moreover, class ‘CoreServiceFacade’ hides the actual implementation of server functionalities and dedicated

functionalities only accessible from within the server application. The ‘socket’ package is responsible for keeping all classes responsible for the actual client server communication via ‘Sockets’.

Model Package

The ‘model’ package contains components necessary for the concrete KMI calculation for resources and shall be described in more detail later on in order to keep confirm with the sequence of development steps actually done.

Functions and Entities Packages

As these packages basically neither got changed nor extended by this work, except for integrating the ULKMS, they won’t be described in detail. The ‘function’ package mainly contains implementations relating to existing functionality within the client side of DS1. The ‘entities’ package contains representations of data types used within server side of DS1.

4.3.2.2 Socket Implementations

In order to use WS within DS1, the server side had to be run from within a ‘Webserver’. To allow using the concept of ‘Sockets’ within server side, a different type of server severing answers to clients’ requests had to be used. To provide respective server with implemented ‘Socket’ concept just has to be exported as ‘Java Archive’ (JAR)[41] file in order to be subsequently started from command line at a certain place from within the physical server machine:

```
> java –jar Server.jar configFile
```

Necessary parameters for starting the server as well as for accessing certain ‘Graphs’[42] within the DB have to be provided by a configuration file via command line argument. Beside other parameters not relevant for the ‘Socket’ communication, the configuration file had to contain the port on which the server should actually be run and had to set whether the calculation process of KMI for resources should be executed directly after starting the server.

In order to give an overview of the basic implementation sequence necessary for new ‘Socket’ communication approach, the sequence diagram for handling client requests gets shown in figure 16.

[41] http://en.wikipedia.org/wiki/JAR_(file_format)
[42] http://en.wikipedia.org/wiki/Graph_database

The current implementation of providing WS operation functionalities was done in a dedicated class. Respective class got renamed to 'SocketInterface' and changed accordingly to implement methods finally executing the clients' requests after handling 'Socket'-related processing of requests in preceding classes. As already can be shown with the help of the sequence diagram in figure 16, various client requests for accessing certain functionality within the RMKMS get also handled via class 'SocketInterface'. The actual implementation of operations accessible by the client side got hidden from external access by a 'Façade'[43] pattern. Class 'CoreServiceFacade' then delegates certain duties to defined classes in packages 'functions' and 'model' implementing core server functionalities as well as the calculation of KMI for resources.

To enable DS1 to actually use defined interfaces, class 'SocketServer' implements the counterpart of the 'Sockets' used in 'Flex' using the 'ServerSocket' class from within 'Java'. The main entry point for the server application after getting started got defined within class 'ServerMain' holding an instance of class 'SocketServer'.

SocketServer

Class 'SocketServer' accepts and delegates client requests to subsequent class instances where initializing respective instance in class 'ServerMain' looks as follows:

```
SocketServer server = new SocketServer(port);
Thread serverThread = new Thread(server);
serverThread.start();
```

Class 'SocketServer' has to be run in a separate 'Thread'[44] allowing the server side, e.g. working of requests while other responsibilities could be done in parallel. New client requests will be accepted in overwritten run method of class 'Thread'.

```
ServerSocket server = new ServerSocket(port);
while(true){
      new Thread(
            new SocketHandler(
                  server.accept(),++socketRequestCounter)).start();
```

[43] http://en.wikipedia.org/wiki/Facade_pattern
[44] http://en.wikipedia.org/wiki/Thread_(computing)

}

In turn, each new client request will be handled in a separate thread allowing for executing different client requests in parallel. Further, processing of a client's request will be done in class 'SocketHandler' after method accept of class 'ServerSocket' returns a new instance of class 'Socket' available from within 'Java'. The 'Socket' class actually represents the communication channel to the client requesting the server.

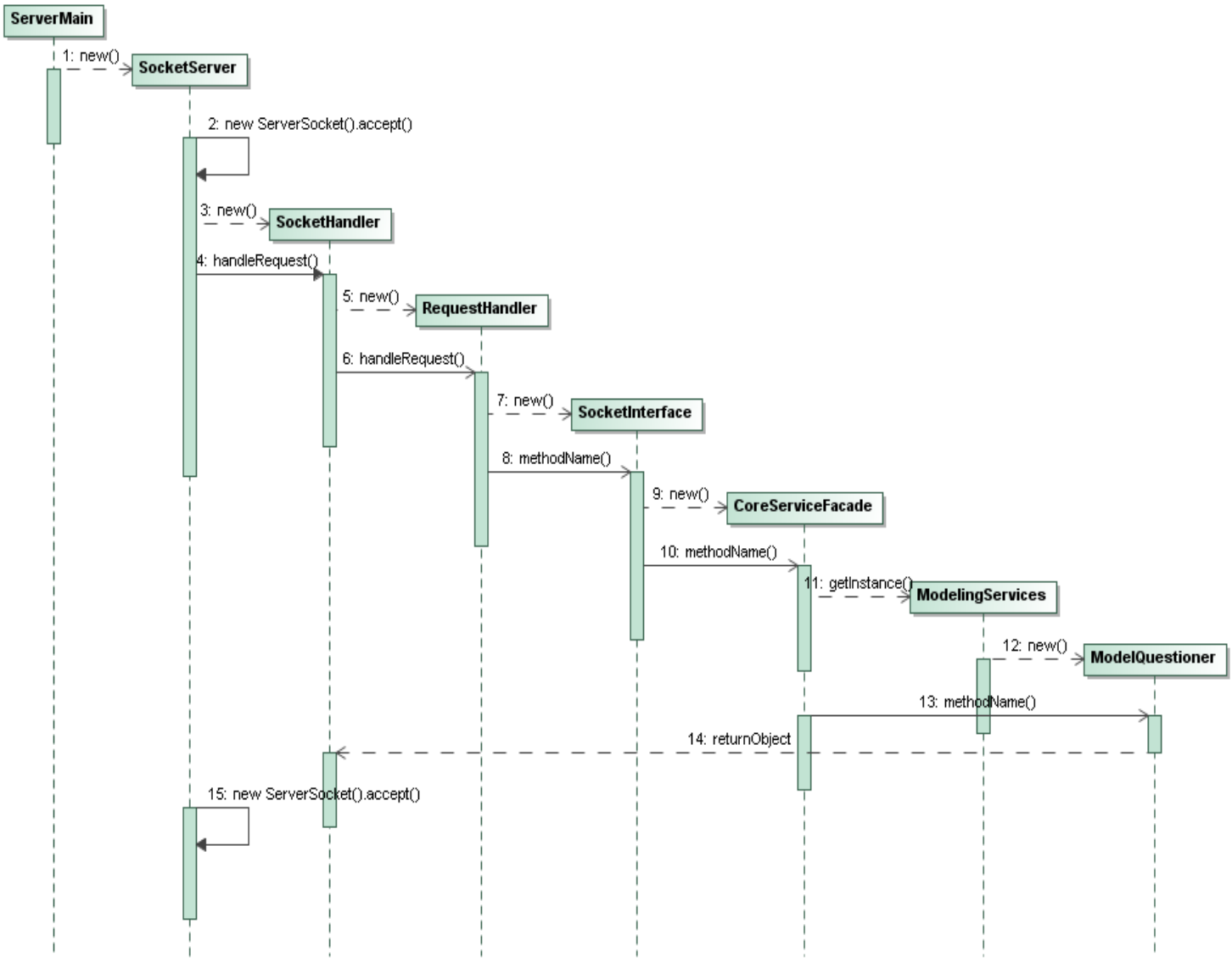

Figure 16: Handling Client Requests Sequence

SocketHandler

As each DS1 client connection must be handled in a single thread, respective instance of class 'SocketHandler' is responsible for working off the client's request. After needed in- and output streams for receiving and responding data get set, method 'readRequest' reads from the

input stream into an instance of class ‘StringBuffer’ as long as the terminating NUL[45] gets sent by the client.

```
while(true){
      if(inputStream.read(readBuffer, 0, 1)== '\0'){
            socketHelper.setRequest(requestBuffer.toString());
            return true;
      }
      requestBuffer.append(readBuffer[0]);
}
```

The ‘setRequest’ method of class ‘SocketHelper’ stores the JSON encoded request to a member and retrieves the method name from the request identifying the actual operation to be executed within server side. Afterwards, the request gets checked by class ‘SocketHelper’ whether it is concerning a certain execution type on server as the server for DS1 gets used for various other client side applications too. This allows for deciding whether the request asks for up- or downloading files to a ‘Web-based Distributed Authoring and Versioning’ (WebDAV)[46] repository affiliated to the VS, checking the server’s policy file for web-based client applications or standard querying and updating of server side content. In the case of what is required for this work, following code provides the entry point for parsing and executing the client’s request:

```
writeResponse(
      new RequestHandler().handleRequest(socketHelper.getRequest()));
```

RequestHandler

This class handles required parsing of the request encoded in JSON format. Method ‘handleRequest’ first converts the JSON string into a JSON object from which certain attributes of the request get extracted:

```
JSONObject object = new JSONObject(requestString);
JSONArray typeArray = object.getJSONArray("types");
JSONArray valueArray = object.getJSONArray("values");
JSONArray methodName = object.getString("methodName");
```

[45] http://en.wikipedia.org/wiki/Null_character
[46] http://www.webdav.org/

As the client request got structured the way that for each parameter of the request corresponding type got transmitted additionally and the name of the operation to be executed got defined, it is possible to decide which method of class 'SocketInterface' has to work off respective request. To be able to call desired method on class 'SocketInterface' 'Dynamic Binding' [47] was used in order to retrieve the actual method instance from class 'SocketInterface'.

```
SocketInterface socketInterface = new SocketInterface();
Class parameterTypes[] = new Class[typeArray.length()];
Object argumentList[] = new Object[valueArray.length()];
Object returnObject = new ReturnObject();

for (int counter=0; counter < typeArray.length();counter++){
      parameterTypes[counter]=Class.forName(typeArray.getString(counter));
}
for (int counter=0; counter < valueArray.length();counter++){
      argumentList[counter] = valueArray.getString(counter);
}

Method method =
      Class.forName("at.tug.mature.service.interfaces.SocketInterface").get
            Method(methodName, parameterTypes);

Object result = method.invoke(socketInterface, argumentList);
returnObject.setMethodName(methodName);
returnObject.setObject(result);

return returnObject;
```

Parameters necessary for invoking desired method of class 'SocketInterface' get created from the parameters sent with the client request in order to subsequently let desired method in class 'SocketInterface' execute according functionality. This returns the result to method 'handleRequest' which in turn packs the result and returns it to class 'SocketHandler' sending it back to the client with the help of the client 'Socket' output stream.

So far, the description of implementations for the 'Socket' approach on server side, a short overview of the package structure within server and an overview for the overall sequence of

[47] http://en.wikipedia.org/wiki/Dynamic_dispatch

accessing operations available by the server was given. Next, a description of implementations for the ULKMS responsible for accepting and storing traced UE within the client side of DS1 will be given.

4.3.2.3 Usage Logging Knowledge Maturing Service

The service retrieving UE triggered by DS1 basically uses the above described 'Socket' communication to transmit various UE from client to server. Subsequently, UE get stored into a dedicated 'Graph' within the TS of the VS.

UE Retrieving

The ULKMS basically works off an 'addUserEvent' message sent from client via already presented workflow. Subsequently, class 'SocketInterface' forwards the operation to class 'CoreServiceFacade' which actually delegates the request to a dedicated worker class contained in the 'functions' package. The method representing the UE retrieving part of the service looks as follows:

```
public boolean addUserEvent(String user, String actionType, String content,
      String uri, String key) {

      return eventFunctions.addUserEvent(user, new Date().getTime(),
            actionType, uri, content);
}
```

UE Storing

Class 'CoreServiceFacade' delegates DS1's request to class 'EventFunctions' within 'functions' package storing received UE to a dedicated 'Graph' in the TS named '_eventgraph'. To access the 'Graph's' content and actually create a connection to the DB server, an URI represented by class 'URI' gets used to identify the respective 'Graph'. The connection between the server and VS gets created with the help of a 'Factory'[48] pattern in order to be able to easily exchange the DB backend.

```
Connector connector = ConnectorFactory.getConnector();
```

[48] http://en.wikipedia.org/wiki/Factory_method_pattern

The instance of class ‘Connector’ gets used to add the UE parameters to the ‘Graph’ in the form of ‘Triples’[49]. From that, method ‘addStatement’ of class ‘Connector’ takes the subject, predicate and object for each triple necessary to store a single UE to dedicated ‘Graph’. Additionally, the URI of the ‘Graph’ has to be provided each time an ‘addStatement’ method call gets done.

```
URI graphUri = new URI(SVocabulary.MATURENS + "_eventgraph");
URI currentEvent = new URI(SVocabulary.MATURENS + "event_" + time);

connector.addStatement(
        eventURI,                currentEvent,            currentEvent,
   SVocabulary.hasEvent,    SVocabulary.hasResource,  SVocabulary.belongsTo,
       currentEvent,              pageURI,                  user,
        graphUri);               graphUri);               graphUri);

      currentEvent,                                       currentEvent,
    SVocabulary.RDFtype,          pageURI,            SVocabulary.timestamp,
   SVocabulary.typeEvent,   SVocabulary.RDFtype,     createLiteral (time),
        graphUri);          SVocabulary.typeRESOURCE,      graphUri);
                                  graphUri);

      currentEvent,              currentEvent,
  SVocabulary.actionType,     SVocabulary.content,
createLiteral(actionType),  createLiteral(content),
        graphUri);                graphUri);
```

As the above code shows, the UE parameters user, content, resource and timestamp get assigned to the event’s unique identifier within the ‘Graph’ for UE.

This chapter described the implementation of basic preconditions for setting up the ULKMS as well as implementing the service itself. This lays down the basis for creating the second KMS to be developed by this work, i.e. the RMKMS. Following chapters basically will describe the implementation of the KMI calculation for resources with the help of the RMKMS.

4.3.3 Resource Modeling Knowledge Maturing Service

After choosing the set of KMI to calculate and grouping of UE so that resulting groups can be used for the mapping between KMI and TI, the environment wherein the actual calculation

[49] http://en.wikipedia.org/wiki/Resource_Description_Framework

process of KMI gets handled had to be set up. First, relevant packages previously shown in figure 15 get described before most important classes needed for the RMKMS incorporating functionalities for the calculation of KMI get figured in respective class diagram in figure 17. Afterwards, implementations for the KMI calculation process will be described.

The process of calculating KMI for resources in this work relates to the calculation of certain properties of resources which in turn get stored in respective RP. From that, the environment necessary for the RMKMS has to be set up accordingly allowing working off the actual KMI calculation process as well as providing access to results of the calculation process by the RM.

4.3.3.1 Overview

4.3.3.1.1 Packages

As figure 15 depicts, following packages got introduced in order to structure respective RMKMS implementations. The ‘model’ package contains relevant implementations handling modeling of resources further split into six sub-packages representing certain functionality realms.

DataTypes

Classes contained in this package represent certain entities used for the calculation process of KMI. The most important class in this context gets represented by class ‘ModelResource’. This class represents a single RP with the help of calculated properties reflecting certain characteristics of a resource as well as providing a means for calculating further resource-related aspects.

Evaluation

The ‘evaluation’ package contains classes needed for analyzing certain aspects of calculated KMI as well as defines methods within classes used for supporting the evaluation process of calculation KMI for resources.

Facade

From within this package, the KMI calculation process is able to access core server functionalities defined by class ‘CoreServiceFacade’ in the ‘service’ package. By methods defined in class ‘ModelQuestioner’, class ‘CoreServiceFacade’ is able to access the RP of certain resources within the system. Moreover, as will be shown later on, class

‘ModelQuestioner’ supports searching for resources in the system upon respective RP. Additionally, this package contains classes responsible for periodically calling the KMI calculation process executing updates of RP.

Helpers

This package consist of classes helping to either convert between certain entity types within the modeling process or execute certain console logging operations.

Core

The ‘core’ package contains class ‘Model’ as well as class ‘ModelSearcher’. Class ‘Model’ represents the container for all the resources in the system, hence can be called the RM within the RMKMS. Moreover, class ‘Model’ provides the entry point for starting the KMI calculation process as well as a means to update corresponding RP. Class ‘ModelSearcher’ is responsible for performing certain search activities triggered by DS1 and complements found resources with properties accessible through the ‘Model’ class. Additionally, this package contains affiliated classes globally needed for certain calculation processes as well as class ‘ModelQuestioner’ – the start point for calling the KMI calculation process as well as the access point for querying and updating the RM containing the results of respective calculation process.

Setters

As updating of various RP during the calculation of KMI comes equipped with the help of TI, class ‘ModelEventSetter’ provides a means to retrieve latest UE stored within the TS. Moreover, this class attaches UE to corresponding resources for subsequent KMI calculation. Class ‘ModelResourcePropertySetter’ gets used to retrieve certain resource properties through attached UE. Those properties subsequently get used to calculate further characteristics of resources like KMI. Furthermore, this package contains all classes responsible for calculating and attaching KMI to RP of respective resources, hence class ‘ModelIIThresholdSetter’ is responsible for calculating certain thresholds needed in order to decide whether a resource is able to reach certain II, classes ‘ModelIISetter’ and ‘ModelMISetter’ then actually attach respective indicators to the resources.

4.3.3.1.2 Classes

To provide an overview of classes implemented within the RMKMS, the class diagram in figure 17 can give a more detailed insight into which entities had been used for handling the service's responsibilities.

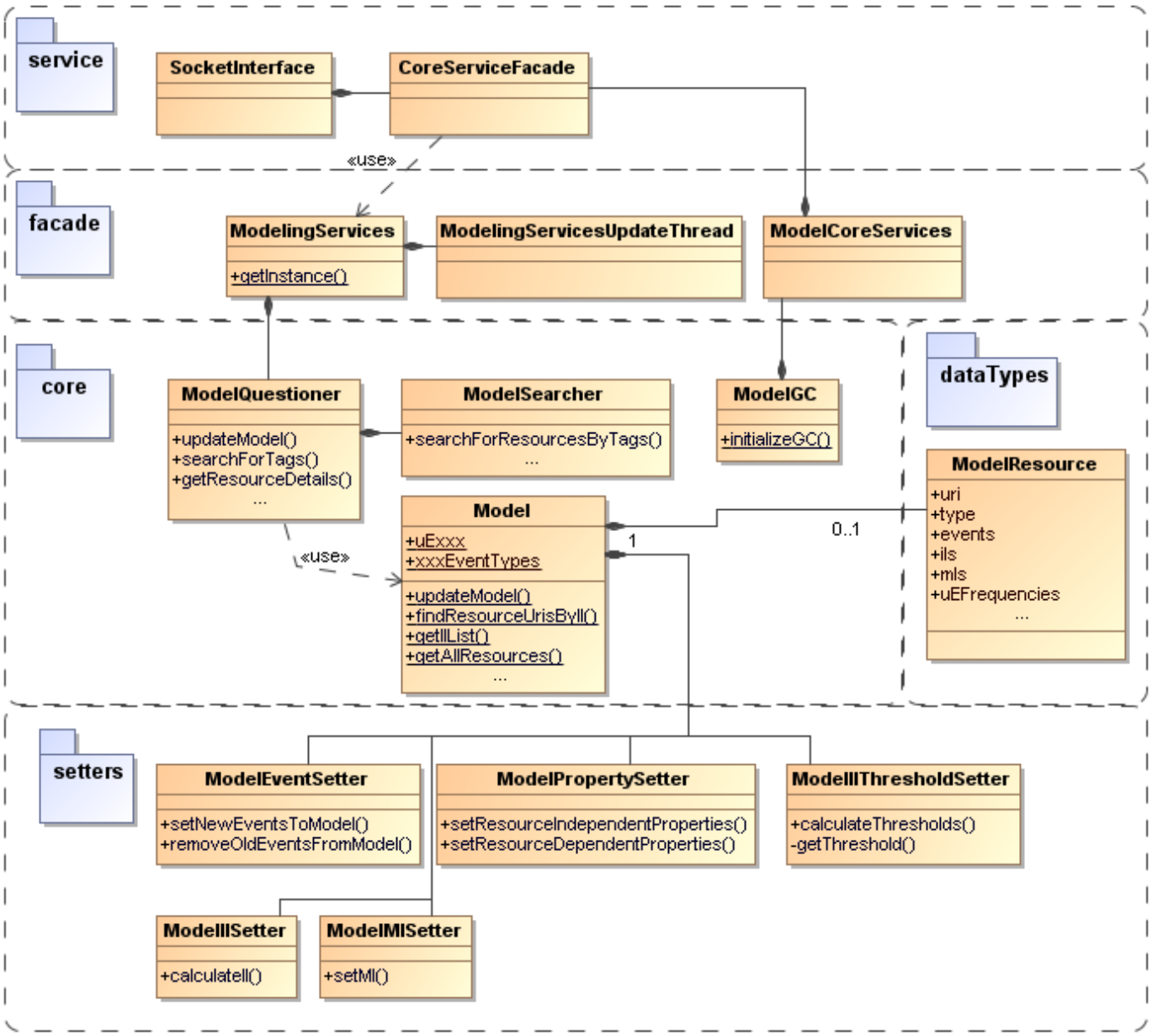

Figure 17: Resource Modeling Knowledge Maturing Service Classes

As figure 17 suggests, class 'Model' together with class 'ModelResource' defines core classes representing the RM underlying the RMKMS. The 'Model' class keeps all RP in a dedicated statically assigned 'Hash table'[50] property in order to provide easy access to the resources during the calculation of KMI as well as later on. A resource and its profile get represented by class 'ModelResource' basically keeping all relevant properties of a resource accessible by

[50] http://en.wikipedia.org/wiki/Hash_table

questioning the ‘Model’ class. Additionally, class ‘ModelResource’ keeps certain properties helping calculating various characteristics of a resource.

Class ‘ModelQuestioner’ is responsible for accessing the RMKMS from client side as well as directly from within the server. Class ‘ModelQuestioner’ can be accessed by triggering method ‘getModelQuestioner’ on ‘Singleton’ class ‘ModelingServices’. ‘ModelQuestioner’ is able to accept queries and updates to the RM represented by class ‘Model’. Responsible methods will be described later on. In order to support extended search operations from within DS1, class ‘ModelSearcher’ helps to perform certain search operations based on RP after extracting various resources from the TS.

As UE necessary for calculating KMI as well as for other resource-related calculations get not directly stored within RP at first instance but get put into the VS’s DB with the help of the ULKMS first, class ‘ModelCoreServices’ is responsible for the provision of UE to calculation processes from within the TS. Moreover, class ‘ModelCoreServices’ provides a means to query server side functionality for the RMKMS provided by class ‘CoreServiceFacade’.

4.3.3.1.3 Knowledge Maturing Indicator Calculation Sequence

Besides keeping all RP represented by the RM, class ‘Model’ is responsible for the calculation of KMI itself. Hence, class ‘Model’ provides certain statically accessible methods in order to either trigger the start of certain calculation processes or retrieve various RP contained in the RM within the RMKMS.

In order to deal with different calculation steps, class ‘Model’ got supported with various helper classes responsible for handling defined steps in the KMI calculation process. Those helper classes are declared in the ‘setters’ package below the ‘model’ package supporting retrieving and attaching new UE to resources in RM and calculating of KMI for resources. The whole sequence of generating updated RP including the calculation of KMI can be described by the sequence diagram shown in figure 18.

To actually trigger the calculation of KMI by the RMKMS, method ‘updateModel’ in class ‘ModelQuestioner’ has to be called in order to forward the request to class ‘Model’ where a new calculation processes gets started subsequently.

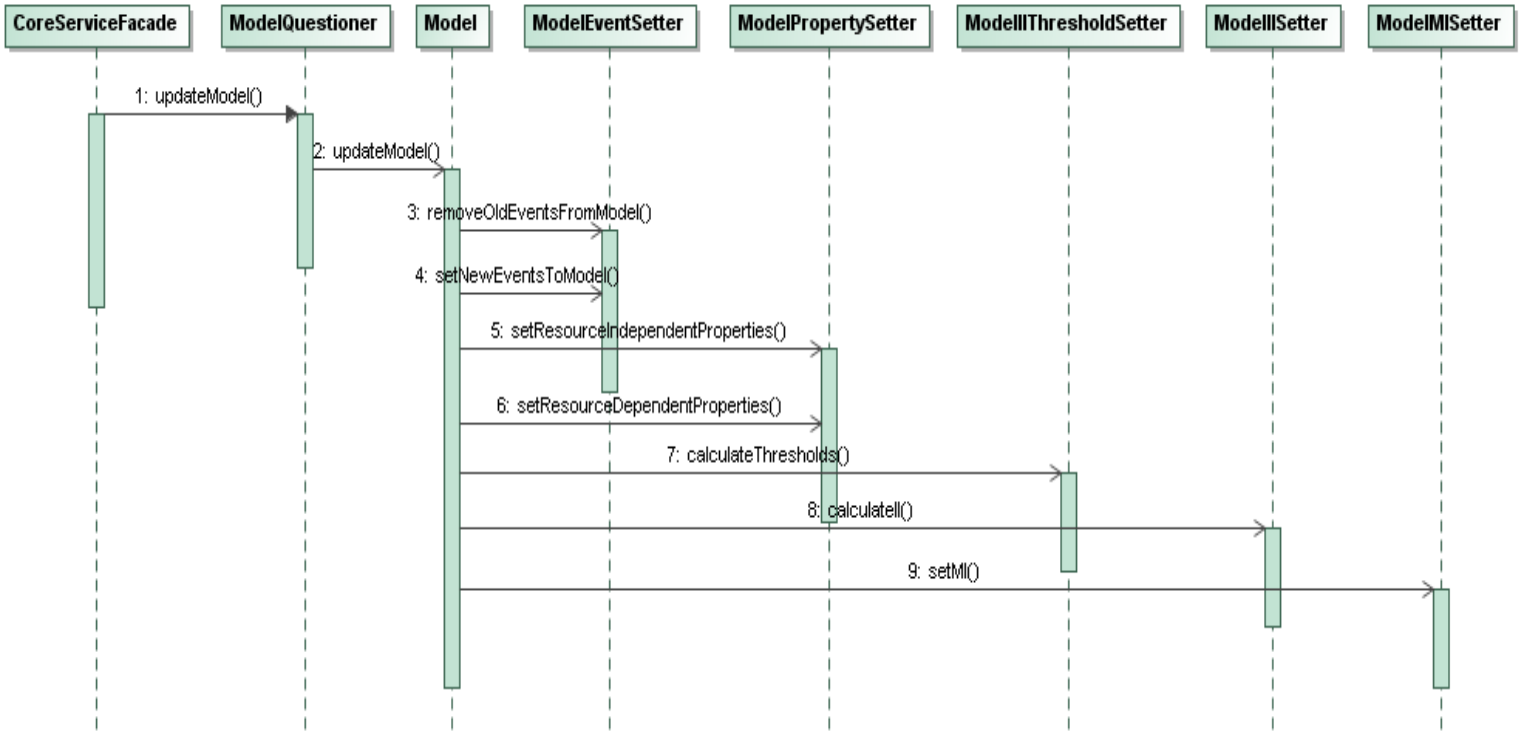

Figure 18: Knowledge Maturing Indicator Calculation Sequence

As respective RP were sought to be updated once a night in order to integrate new usage data generated during a day of DS1's application, an updating mechanism got provided in order to refresh the resources' characteristics and to re-calculate KMI for respective resources. Hence, class 'ModelingServiceUpdateThread' is responsible for calling method 'updateModel' in class 'ModelQuestioner' once a night setting up the service if not created yet or updating established RM containing all the RP with calculated KMI.

So far, an overview of the environment necessary for the RMKMS for calculating KMI for resources as well as classes used for representing results of calculation processes in respective RP in the RM got shown. Moreover, the overall sequence for the calculation process of KMI (with will be described in detail in the following) together with responsible class instances could be shown.

In the following, the first step in the calculation process of KMI responsible for identifying resources with regard to traced UE gets described.

4.3.3.2 Extracting Resources from User Events

Following the overall KMI calculation process depicted in figure 18, respective process has to be started by calling method 'updateModel' on class 'Model'. This subsequently allows the

RMKMS to begin with updating resources with regard to KMI. From that, the next step is to attach collected UE to either already existing or new resource representations in the RM. Instances of class ‘ModelResource’ are responsible for representing resources used in DS1 within the RMKMS.

Method ‘removeOldEventsFromModel’ of class ‘ModelEventSetter’ gets called in order to remove all UE from current RP used for previous calculations. UE used for KMI calculations get assigned directly to RP of respective resources so that property ‘events’ of instances of class ‘ModelResource’ get reset in order to start a new calculation. As identifying resources in the system works directly with the help of collected UE, whether new representations of resources should be created gets decided upon the ‘resource’ parameter of new UE.
For retrieving possibly new UE from TS and assigning them to resources, method ‘setNewEventsToModel’ gets called on the instance of class ‘ModelEventSetter’ held by class ‘Model’.

```
Model.sortedEventsSinceLastUpdate =
      Event.sortUserEventList(
            ModelGC.typeConverter.createUserEventListFromGraphEventList(
                  getNewEvents(lastUpdateTime)));
```

Given parameter ‘lastUpdateTime’ gets used in order to restrict retrieving UE to a certain time span. Property ‘lastUpdateTime’ initially gets set by class ‘Model’ and gets updated each time a KMI calculation process finishes. To decide for how long back in time the RMKMS should include collected UE, parameter ‘lastUpdateTime’ must be set accordingly. After an initial calculation of KMI for all RP, new calculation processes always include events since the last update. Above code fragment from method ‘setNewEventsToModel’ shows retrieving of UE for all the resources in a certain time frame beginning at ‘lastUpdateTime’ and reaching up to the time the calculation got started. First, method ‘getNewEvents’ retrieves collected UE stored within TS with the help of class ‘ModelCoreServices’, second, an instance of class ‘ModelTypeConverter’ prepares UE so that they can be applied to ascending sorting with regard to time and third, UE get assigned to globally accessible ‘sortedEventsSinceLastUpdate’ property in class ‘Model’.

```
eventsPerResource =
      ModelGC.typeConverter.getUserEventsPerResourceFromUserEventList(
            Model.sortedEventsSinceLastUpdate);
```

```
for(String resourceUri : eventsPerResource.keySet()){
     foundOrCreatedResource = findResourceOrAddNewOne(resourceUri);
     foundOrCreatedResource.events = eventsPerResource.get(resourceUri);
}
```

To be able to process new UE according to their affiliation with resources, they get assigned to corresponding resource URI first. Afterwards, possible new or already existing resources will be updated with new UE by method ‘findResourceOrAddNewOne’. Respective method is responsible for adding newly created resources to the globally accessible resources property in class ‘Model’. From that, it becomes obvious that the RMKMS only takes resources into account which had been focused by certain TI from within DS1, i.e. had been used for certain KMA.

As for now resources – either used the first time by users or focused by new user activities within DS1 – got updated with regard to possibly new UE, the next step basically will focus on calculating necessary frequencies of UE in previously stated groups so that further calculation of KMI can rely on those numbers when it comes to deciding whether various resources reach certain KMI.

4.3.3.3 User Event Frequency Calculation per User Event Groups per Resource

The calculation of certain KMI for resources mainly gets based upon the number of assigned UE from within certain groups. For that, the frequency with which a resource was used (‘evented’) in a certain UE group gets calculated. From the fact that those frequencies of UE in a certain group for respective resources indicate denominating characteristics of resources worth storing in corresponding RP, UE frequencies shall also be called properties of resources among other possible properties of resources. Moreover, the RP does not ‘forget’ calculated UE frequencies for resources but takes already calculated frequencies in possibly deseeding calculation periods into account. This enables the KMI calculation process to continually consider increasing or decreasing event frequencies spanning over more than one calculation period. Otherwise, if calculated properties would be reset each time a new calculation starts, possible KMI accomplishments from earlier periods would be ignored.

After new UE got assigned to existing or newly created resource representations, now frequencies in UE groups have to be set. Moreover, general resource properties directly derived from UE get assigned to respective resources in this step. For the assignment of those

properties to resources mainly class ‘ModelResourcePropertySetter’ is responsible. When assigning properties derived from UE to resources, each resource in RM gets updated by method ‘updateModel’ in class ‘Model’ calling dedicated methods in class ‘ModelResourcePropertySetter’ as shown by following code fragment.

```
for (ModelResource resource : resources.values()){
      resourcePropertySetter.setResourceType (resource);
      […]
      resourcePropertySetter.setResourceIndependentProperties (resource);
}
resourcePropertySetter.setResourceDependentProperties();
```

First of all, it is necessary to assign the resource type to respective resources in order to be able to differentiate following KMI calculation with regard to resources’ types. To decide whether the resource belongs to a certain type, assigned UE have to be examined. Furthermore, other more general resource properties, e.g. the author of a resource or its editors, get applied to respective RP, though for the case of KMI calculation especially methods ‘setResourceIndependentProperties’ and ‘setResourceDependentProperties’ are of importance.

As the calculation of KMI for resources mainly relies on frequencies of certain UE types, UE got assigned to certain groups helping to calculate respective indicators. From that, frequency properties needed for the calculation process represent the number of times a resource received certain TI in certain groups which in turn basically represent underlying semantics of chosen KMI. Those properties get set in method ‘setResourceIndependentProperties’ accordingly.

Independent Properties

Method ‘setResourceIndependentProperties’ is responsible for either increasing or decreasing UE frequency counters in respective UE groups. Class ‘ModelPropertySetterHelper’ actually implements setting UE frequencies to resources. For that, instances of UE assigned to resources get checked whether they belong to a certain group in order to increase or decrease respective frequency properties of resources. Basically, class ‘ModelPropertySetterHelper’ defines counting methods respecting previously stated UE groups so that up- and down counting of respective resource properties happens in accordance with defined UE groups. Corresponding methods get declared as follows:

```
propertySetterHelper.increaseUEFrequenciesxxx(ModelResource resource,
      Event event){…}
propertySetterHelper.decreaseUEFrequenciesxxx(ModelResource resource,
      Event event){…}
```

The actual implementation of method ‘increaseUEFrequenciesChanging’ counting up the resource properties which for example help to calculate change-related KMI looks as follows:

```
if(Model.changingEventTypes.contains(event.actionType) == true){
      resource.uEFrequencies.put(
            ModelUEFrequencyType.uEFrequencyChange,
            resource.uEFrequencies.get(
                  ModelUEFrequencyType.uEFrequencyChange) + 1);

      if(resource.personsChanged.contains(event.user) == false){
            resource.personsChanged.add(event.user);

            resource.uEFrequencies.put(
                  ModelUEFrequencyType.uEFrequencyChangePerson,
                  resource.uEFrequencies.get(
                        ModelUEFrequencyType.uEFrequencyChangePerson) + 1);
      }
}
```

Relating to the code above, first, it gets checked whether the current UE has a type indicating that it belongs to respective UE group (represented by ‘Model.changingEventTypes’) in order to be taken into account when considering the calculation of respective KMI depending on the change of a resource by a user. Second, the frequency property indicating how often a change happened to a resource gets increased by one. As another KMI relies, e.g. on how often a resource was changed by different persons, this method gets also used to set another frequency property for respective resource. Additionally, method ‘increaseUEFrequenciesChanging’ not only increases the number of times a resource got ‘evented’ by certain UE types but registers respective users triggering these UE to be able to calculate respective KMI.

Dependent Properties

Method ‘setResourceDependentProperties’ is responsible for increasing or setting certain frequency properties to resources upon already calculated properties from within the method ‘setResourceIndependentProperties’. Following methods within class ‘ModelPropertySetterHelper’ work off those responsibilities:

```
propertySetterHelper.increaseUEFrequencyxxx(ModelResource resource){…}
propertySetterHelper.setUEFrequencyxxx(ModelResource resource){…}
```

The actual implementation of setting the number of resources which make up a certain resource looks as follows:

```
if(ModelGC.method.isResourceOfTypeCollection(resource.type) == true){
      resource.uEFrequencies.put(
            ModelUEFrequencyType.uEFrequencyMadeOutOf, 0);
      resource.uEFrequencies.put(
            ModelUEFrequencyType.uEFrequencyMadeOutOf,
            getCollectionEntryUris(
                  Model.resources,
                  resource.resourceUri).size());
}
```

First, it has to be checked whether the current resource is of a certain type so that respective property only gets set to resources being able to contain other entities. In difference to independent properties, respective dependent properties will be reset here each time a new calculation process starts as dependent properties get based on calculated independent properties. From that, to calculate the frequency of sub-entities for a certain resource, it is first necessary to reset respective property to be able to reflect this resource’s property at a certain point in time. Method ‘getCollectionEntryUris’ returns all URI being already attached to respective resource. Respective URI got assigned to the resource while calculating independent properties for respective resources from new UE. From that, the property expressing the number of current sub entities of a resource of type collection can be set.

As this step calculated certain UE frequencies in respective groups for all the resources in the RM, the next step will use the concept of II. Most of those indicators are capable of relating a certain resource to all the other resources in the system based on calculated frequency

properties. From that, it is necessary to calculate certain thresholds upon which the possible assignment of II to resources can be decided shown in the next chapter.

4.3.3.4 Intermediate Indicator Threshold Calculation with Regard to User Event Frequencies

Whether discriminating II can be assigned to certain resources depends on thresholds calculated for respective II. These thresholds depend on the overall frequencies of UE in respective groups for all resources. Therefore, II thresholds have to be calculated first. Following the overall KMI calculation sequence in figure 18, this step gets worked off by class 'ModelIIThresholdSetter'. Method 'calcuateThresholds' called from within method 'updateModel' in class 'Model' is responsible for the calculation of II thresholds per resource type. Upon the calculated threshold it can be decided whether certain resources reach defined discriminating II.

```
Model.thresholdCollectionOrganizeCollection =
      multiply(
            getThreshold(
                  ModelResourceType.collection,
                  ModelUEFrequencyType.uEFrequencyOrganizeCollection),
            Model.FACTOR_COLLECTION_ORGANIZE_COLLECTION);
Model.thresholdDiscussionOrganizeCollection =
      multiply(
            getThreshold(
                  ModelResourceType.discussion,
                  ModelUEFrequencyType.uEFrequencyOrganizeCollection),
            Model.FACTOR_DISCUSSION_ORGANIZE_COLLECTION);
Model.thresholdDigitalResourceOrganizeCollection =
      multiply(
            getThreshold(
                  ModelResourceType.digitalResource,
                  ModelUEFrequencyType.uEFrequencyOrganizeCollection),
            Model.FACTOR_DIGITAL_RESOURCE_ORGANIZE_COLLECTION);
```

Method 'calcuateThresholds' basically defines threshold calculation calls for each resource type and for each UE group as shown exemplarily in the above code fragment. Calculated thresholds get multiplied with certain factors accordingly in order to be able to adapt the threshold to the desired discrimination strength of a certain II. Before a threshold can be adapted as shown, method 'getThreshold' in class 'ModelIIThresholdSetter' is supposed to calculate the actual threshold. Method 'getThreshold' takes two parameters: one for

denominating the resource type and one for denominating the respective UE group for which the threshold gets calculated. With the help of the UE group identifier (e.g. ‘ModelUEFrequencyType.uEFrequencyOrganizeCollection’) it is possible to access the actual value of a certain frequency property when applied to a resource. The first part of method ‘getThreshold’ implemented in ‘ModelIIThresholdSetter’ basically looks as follows:

```
ArrayList<Integer> distinctFrequencies = new ArrayList<Integer>();

for(ModelResource resource : Model.resources.values()){
      if(
            ModelGC.method.isOfSameResourceType(
                  resource.type,resourceType) == true &&
            distinctFrequencies.contains(
                  resource.uEFrequencies.get(ueGroupId)) == false){

            distinctFrequencies.add(resource.uEFrequencies.get(ueGroupId));
      }
}
```

The calculation of a threshold for a certain II relies on all the frequency values of resources of a certain type in the RM for a given UE group. Therefore, all the values for a given frequency property get stored to property ‘distinctFrequencies’. Actually, the number of resources in the RM itself won’t be taken into account when it comes to the calculation of certain thresholds. As early analyzing of resources in DS1 yielded, a lot of resources were brought into DS1 once but not receiving any further user activities anymore. From that, it becomes obvious that taking the value of the total number of resources into account when calculating thresholds would yield very low threshold values. This in turn would weaken the discriminative power of II as nearly all resources would be able to reach indicators based on very low thresholds. From that, only distinct frequency values get stored accordingly.

The second part of method ‘getThreshold’ actually deals with calculating the threshold upon collected distinct frequency values in property ‘distinctFrequencies’:

```
int index = -1;
Integer[] distinctFrequenciesArray =
      distinctFrequencies.toArray(new Integer[distinctFrequencies.size()]);
```

```
java.util.Arrays.sort(distinctFrequenciesArray);

if(distinctFrequenciesArray.length % 2 == 0){
      index = distinctFrequenciesArray.length / 2;

      if(
            index >= 0 &&
            index  < distinctFrequenciesArray.length){
            return distinctFrequenciesArray[index];
      }
}else{
      index = distinctFrequenciesArray.length / 2;

      if(
            index + 1  >= 0 &&
            index + 1   < distinctFrequenciesArray.length){

            return distinctFrequenciesArray[index + 1];
      }
}
return 1;
```

Calculating the threshold for discriminating II upon certain UE groups happens upon the ascended sorted distinct frequency values of resources stored to property ‘distinctFrequeciesArray’. In case of an even frequency value list’s length, the threshold gets set to the value from the index representing half the list’s length plus one, in the case of an odd length, the threshold gets set to the value from the index representing half the list’s length. If the list length is less than two, just one gets returned as threshold.

From that, the threshold for discriminating II out of a certain UE group approximately represents the respective mean frequency value considering all resources in the RM without actually using same values twice.

This step was able to show how the threshold calculation for discriminating II works. This yields the basis for actually assigning defined II to resources respecting their UE frequencies in groups. Using this basis for the assignment of II to resources will be explained in the next step of the KMI calculation process.

4.3.3.5 Assigning Intermediate Indicators to Resources with Regard to Thresholds

After thresholds for deciding whether certain resources could reach defined discriminating II got calculated, actual assignment of II to resources gets done by class ‘ModelIISetter’ as shown by the sequence diagram depicted in figure 18. Method ‘calculateII’ in class ‘ModelIISetter’ works off the assignment of certain II to resources called by method ‘updateModel’ from within class ‘Model’. With regard to setting the frequency values of UE for certain groups for resources shown in a previous step, following method declarations get used in order to assign II to resources within class ‘ModelIISetter’:

```
setIIxxx(ModelResource resource){…}
```

To exemplarily show how those methods work, method ‘setIIOrganizeCollection’ was chosen to examine how respective II get set within.

```
if(resource.uEFrequencies.get(
      ModelUEFrequencyType.uEFrequencyOrganizeCollection) == 0) {

      resource.iIs.add(ModelIIType.iINOrganizeCollection);
}else{
      if(
            intermediateIndicatorSetterHelper.exceedsThreshold(
                  resource,
                  resource.uEFrequencies.get(
                        ModelUEFrequencyType.
                              uEFrequencyOrganizeCollection),
                  Model.thresholdCollectionOrganizeCollection,
                  Model.thresholdDiscussionOrganizeCollection,
                  Model.thresholdDigitalResourceOrganizeCollection)
            == true){

            resource.iIs.add(ModelIIType.iIDOrganizeCollection);
}else{
            resource.iIs.add(ModelIIType.iIPOrganizeCollection);
}
```

It becomes apparent that all three types of II get set with regard to the actual values of respective frequency properties of resources and the actual threshold for discriminating II. As discriminating II ‘iIDOrganizeCollection’ depends on respective threshold, method

‘exceedsThreshold’ defined in class ‘ModelIISetterHelper’ checks whether the resource reaches desired II. II in method ‘setIIOrganizeCollection’ express whether the resource was organized in a collection at least once, never had undergone underlying UE or was organized in a collection very often according to defined UE in respective group.

All methods in ‘ModelIISetter’ handling different II rely on respective frequency properties of resources except method ‘setIIStandard’ and method ‘setIIUseWide’ taking into account already calculated II and the total number of UE received respectively.

This step showed how II got assigned to resources having certain numbers of UE in respective groups. The last step in the calculation process of KMI for resources actually assigns KMI to resources with the help of calculated II which will be shown in the next chapter.

4.3.3.6 Knowledge Maturing Indicator Assignment with Regard to Intermediate Indicators

In order to assign the results of calculated II with regard to KMI mappings to resources, the actual assignment of the textual representation of resulting KMI to resources has to be done in order to fill respective RP with derived KMI.

Method ‘setMI’ in class ‘ModelMISetter’ maps calculated II for resources in the RM to their actual representation by KMI. KMI assigned to resources get stored in respective property ‘mIs’ within instances of class ‘ModelResource’. Following code fragment exemplarily shows how KMI get mapped to calculated II within method ‘setMI’.

```
if(resource.iIs.contains(ModelIIType.iIDMadeOutOfOthers == true)){
      maturingIndicatorSetterHelper.addIndicator(
            resource,
            ModelMIType.An_artifact_was_created_by_integrating_parts_of_oth
                  er_artifacts);
}
```

KMI properties of respective resources get assigned with regard to the resource type of underlying resource so that the mapping done in table 6 gets applied with respect to different resource types.

Described implementations for calculating KMI for resources in DS1 yielded the implementation of a client side framework being able to use the ULKMS implemented within

the server part of DS1 relying on implemented ‘Socket’ client server communication. Actual calculation of KMI got implemented by the RMKMS using the idea of representing resources within the system with regard to certain properties yielded from respective calculation process upon UE. As the RMKMS as described so far won’t be able to provide its results to the client side of DS1, following chapters will describe how the RMKMS got able to be accessed by the client side before describing done evaluations on results of respective service as well as summarizing results of this work.

5 Resource Modeling Knowledge Maturing Service for Search and Summary

The aim of this work, namely to elaborate on the possibility of calculating usage-based KMI for resources within a POLME, yielded the possibility to integrate results directly in DS1 to be used by users. This in turn enables this work to support defined KMA by directly intervening in search processes and visualizing of resource information with regard to resources' maturity. From the fact that representing knowledge artifacts should depend on levels of maturity so that the user can decide on the importance on a resource while browsing content [Weber et al., 2009], the RM provides certain data used for a more detailed view upon resources in DS1 and re-implemented search processes within DS1. The basic environment for storing the profiles of resources including calculated II and KMI was done working-memory-based.

To provide users of DS1 more details on resources as well as the possibility to search with the help of properties representing the maturity of resources, it got decided to not directly show KMI for resources but more flexible II. This becomes reasonable by II's capability of more likely expressing the actual maturity of resources respecting users' understandings of the concept of maturity in a specific POLME. Moreover, this was reasonable as provided textual representations of II more closely reflect results of maturing processes within DS1 as they allow users to relate II to actual tool functionality on a more concrete level than KMI would have been able to.

As II remain stable in their semantic as well as in their programmatic declaration as long as the underlying environment (POLME) does not change, these properties of resources in the RM are able to describe resources independently of whether they get used for KMI calculation or not. From that and from the fact that II are more powerful to express the actual meaning of KMI for the user in a certain POLME as II are closer to the actual user activities the POLME supports, II will be used to represent resources in DS1 when it comes to actually supporting concrete KMA. II get used to help users in search processes as well as to enable judging resources with regard to maturing related characteristics instead of using KMI for this purpose directly. Doing so gives users of DS1 the chance to derive the status of used resources from a 'Knowledge Maturing' perspective considering their understanding of what 'Knowledge Maturing' actually could mean in their concrete working environment.

As searching for information is very common, typically not that much effort is made in manual labeling resources so that they become findable. Enabling software must enhance search processes with functionalities allowing filtering resources with respect to maturity levels. From that, knowledge workers will receive support in finding relevant resources by directly showing resources with regard to II. Similarly to the AMD approach, calculated II being stored in respective RP provide detailed information of resources so that certain resources can be recommended within search processes. This addresses the missing aid in several KMA, e.g. ‘find relevant digital resources’ or ‘familiarize oneself with new information’.

Moreover, II not only can be used for search process but for displaying extended information upon resources as well. Again, this supports users in certain KMA, e.g. ‘asses, verify and rate information’ as well as ‘keeping up-to-date with organization-related knowledge’. The automatic provision of II helps to give knowledge workers more information on resources they deal with in order to reflect on outcomes of work processes, make relevance judgments individually and get interesting knowledge presented automatically.

From that, calculated II got integrated into search as well as at places where more information than just, e.g. the rating or tags of a certain resource, was desired. Hence, the RMKMS and respective calculation results stored in the RM got made able to provide certain information on resources within DS1 directly to users. For this approach, the provision of information followed the same communication principle as for example the ULKMS was based on. Methods accessing the RMKMS in order to retrieve various RP from the RM got implemented in class ‘SocketInterface’ and class ‘CoreServiceFacade’ so that class ‘ModelQuestioner’ got able to answer queries by directly questioning the ‘Model’ class.

As search processes within DS1 actually relied on user assigned tags for stored resources within the TS and as the structure of services in DS1 required calculated II to be stored to TS first before used in search processes, respective storing of calculated II got implemented too. As requirements for search processes in DS1 changed independently from this work, storing of II to TS got obsolete hence implementations done will not be described here. The actual search processes together with implementations necessary for querying the RM for resources will be described in the next chapters.

5.1 Searching for Resources with the help of Intermediate Indicators

To enable searching for certain resources with the help of calculated II, the RM was made accessible from within DS1. This allows users to search not only by entered keywords but by II getting displayed in the 'Search Widget' of DS1 accordingly. Hence, users got able to select II for searching for resources the same way as they would do by user assigned tags. After a user triggers a search, the 'Aggregated Resource Search Service' (ARSS) – as conceptual overlay to the RMKMS – provides resources with chosen II as well as with possible tags chosen.

The 'Search Widget' within DS1 had to be redesigned in order to integrate new search possibilities. An initial mockup for the new 'Search Widget' can be figured as shown in figure 19.

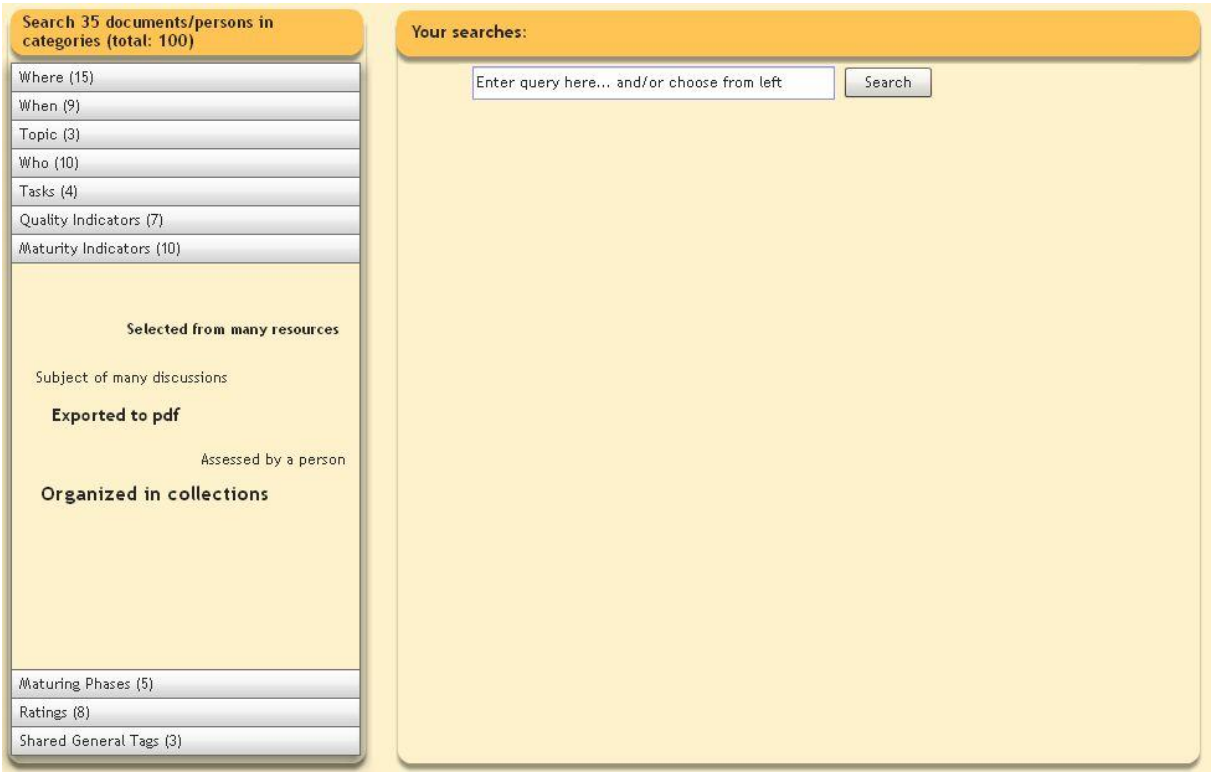

Figure 19: Extended Search Widget Mockup

Besides the possibility to search by II and manually entered keywords, it was planned to enable the user to search by several other pre-defined categories too, though most of them got not relevant for this work except category 'Shared General Tags'. This category should be able to show 'User Generated'[51] tag hierarchies enabling users to browse through all the tags ever assigned to various resources in DS1. From that, the overall current search process as it was conducted currently had to be changed too. The new search strategy including both the

[51] http://en.wikipedia.org/wiki/User-generated_content

user generated hierarchy of tags and calculated II to be selected could be figured as shown in figure 20.

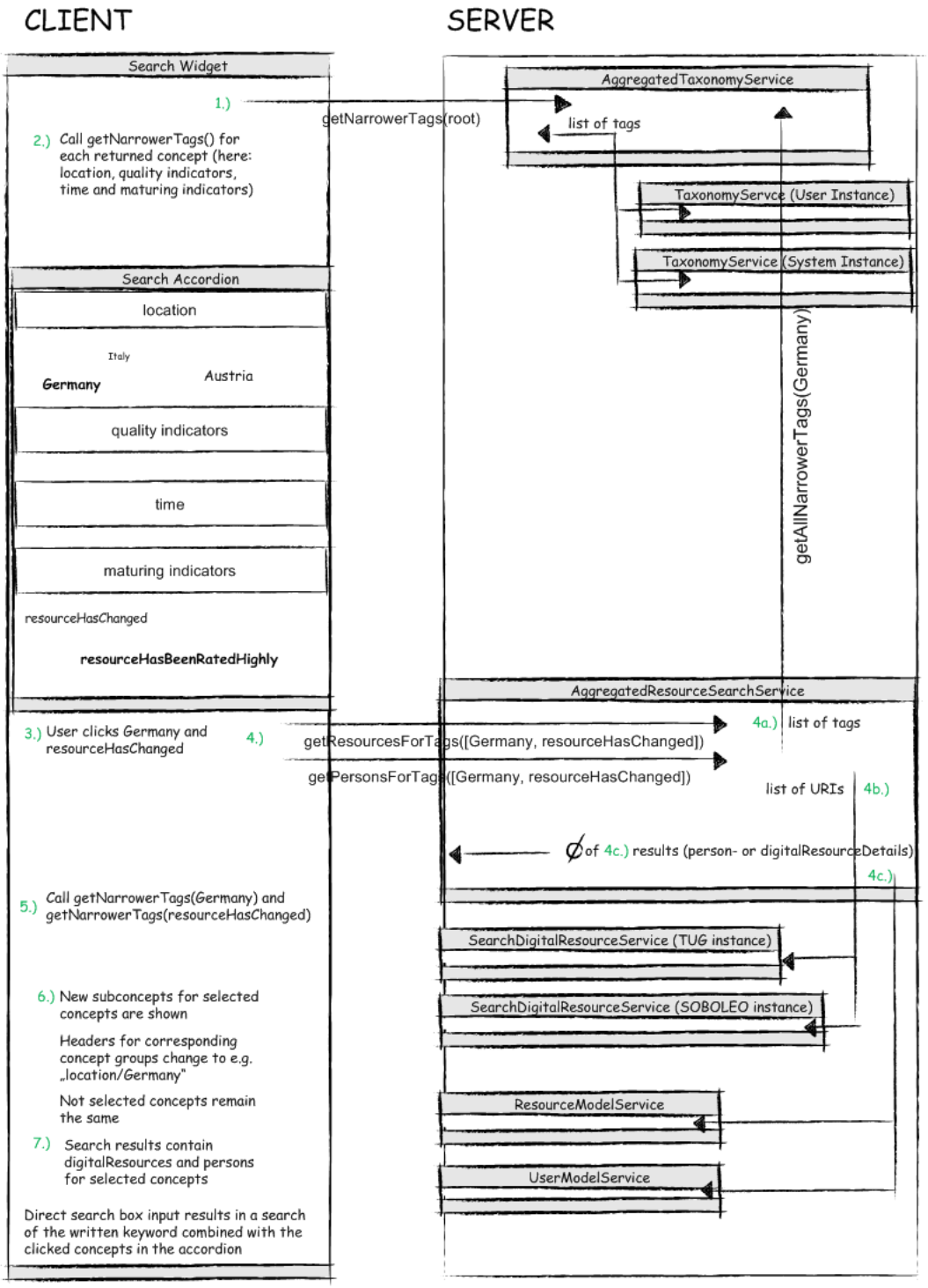

Figure 20: Extended Search Sequence Mockup

Basically, it was sought to deal with new search requirements with the help of WS. For reasons described in this work, they had to be removed as described. As searching shouldn't just rely on tags but on a more sophisticated hierarchical tag structure generated by users of DS1, a separate already existing system dealing with those structures had to be staked to DS1 with the help of WS. Figure 20 shows both the structure of services dealing with searching

and the sequence of client server communication necessary to use tag structures and calculated II for searching.

The ARSS which got implemented within this work in the same environment as used for other DS1 functionalities got responsible for handling actual user search requests, retrieving possible sub-tags of a given search keyword and to search for resources by both user assigned tags and calculated II. To enable the extended search process, the actual calling sequence got implemented as shown in figure 21. Details of how the search process actually searches for resources with given tags in TS got excluded from the sequence diagram as this was not part of this work.

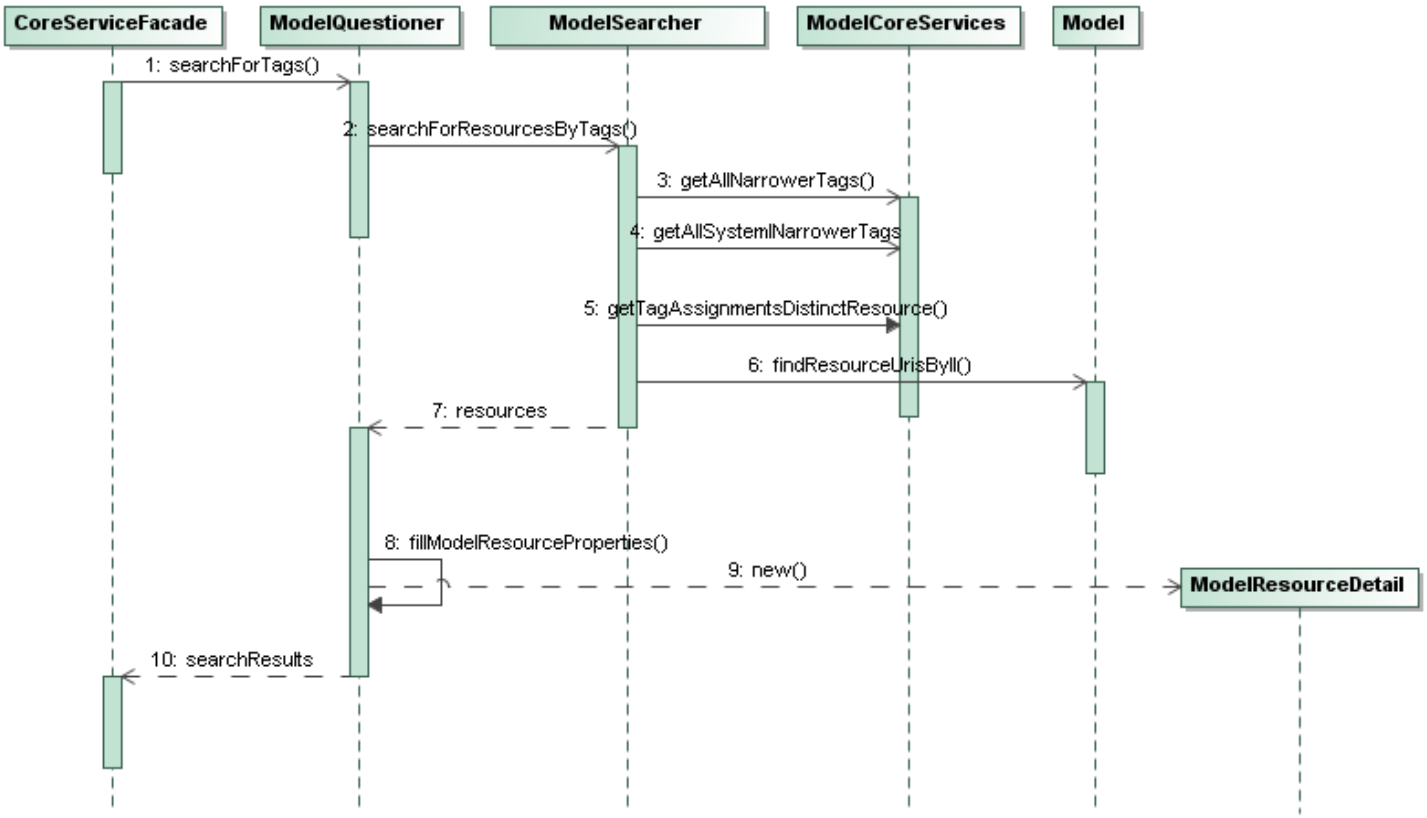

Figure 21: Extended Search Sequence

After class ‘SocketInterface’ retrieves message ‘searchForTags’ forwarding it to class ‘CoreServiceFacade’, class ‘ModelQuestioner’ is mainly responsible for handling the overall search request with the help of method ‘searchForTags’.

ModelQuestioner

Basically, class ‘ModelQuestioner’ is responsible for delegating all required TS queries, WS operations and RM accesses to further classes in order to retrieve search results for tags and II. The following code fragment reflects, first, the delegation of the actual search request to

class ‘ModelSearcher’ instantiated in property ‘resourceSearcher’ and second, retrieving additionally available properties for found resources from RM and available UM. Following code fragment was taken from method ‘searchForTags’ in class ‘ModelQuestioner’.

```
ArrayList<ModelResourceDetail> result =
      new ArrayList<ModelResourceDetail>();
ModelResourceDetail resourceDetail = null;

for(
      ModelSearcherResult foundResource:
            resourceSearcher.searchForResources(
                  user,
                  tags,
                  searchOperation)){
      […]
      result.add(fillModelResourceProperties(foundResource));
}
return (ModelResourceDetail[])
      result.toArray(new ModelResourceDetail[result.size()]);
```

The actual search results get represented by instances of class ‘ModelResourceDetail’ filled by method ‘fillModelResourceDetails’. In order to provide complete search results for each found resource, the actual URI of the resource, a property whether it’s found in a shared or a private space, the name and type of the resource and the overall rating for the resources get retrieved directly from TS. For additional properties of a resource contained in the RM, class ‘Model’ gets queried in order to retrieve assigned II, parent resources, i.e. collections or discussions, the author / editors and related persons as shown in next code fragment of method ‘fillModelResourceDetails’.

```
return new ModelResourceDetail(
      foundResource.resourceUri,
      foundResource.isSharedResource,
      ModelGC.socket.getNodeLabel(foundResource.resourceUri),
      ModelGC.socket.getResourceType(foundResource.resourceUri),
      ModelGC.socket.getOverallRating(foundResource.resourceUri),
      Model.getRelatedPersons(foundResource.resourceUri),
      tagFrequencies,
      Model.getAuthor(foundResource.resourceUri),
      Model.getParentResourceList(foundResource.resourceUri),
```

```
      Model.getIIList(foundResource.resourceUri),
      Model.getEditorList(foundResource.resourceUri),
      ModelGC.socket.getUserRating(user,foundResource.resourceUri),
      Model.getRecentArtifact(foundResource.resourceUri),
      Model.getRecentTopic(foundResource.resourceUri),
      Model.getOwnedSharedCollections(foundResource.resourceUri),
      Model.getOwnedFollowedCollections(foundResource.resourceUri),
      Model.getContributedDiscussions(foundResource.resourceUri),
      Model.getContributedResources(foundResource.resourceUri),
      Model.getTopicScores(foundResource.resourceUri));
```

From this, it becomes apparent that the RM as such is responsible for providing all those properties of resources which can be directly derived from UE. Additionally, a usage-based UM got tied to the RM. Implementing the UM got based upon the same approach and code generated by this work so that the UM got capable of reflecting user-related properties within the system. From that, it is possible to query the model for user's recent resources and topics she dealt with, shared collections she owns, discussions she contributed to and an assessed list of topics she dealt with.

ModelSearcher

As method 'searchForTags' in class 'ModelQuestioner' suggests, method 'searchForResources' in class 'ModelSearcher' got responsible for both querying the TS for resources with certain tags attached and for involving the RM in order to retrieve resources with given II from within the search request. In the following, the first part of method 'searchForResources' can be shown:

```
ArrayList<String> allFoundTagLabels = new ArrayList<String>();
allFoundTagLabels.addAll(tags);

for(String tag : tags){
      allFoundTagLabels.addAll(
            ModelGC.socket.getAllNarrowerTags(tag));
      allFoundTagLabels.addAll(
            ModelGC.socket.getAllSystemNarrowerTags(tag));
}
```

As tag hierarchies as well as possibly to introduce II hierarchies were sought to be maintained by another application called SOBOLEO [Braun, Schmidt and Zacharias, 2007], actually two

WS calls (‘getAllNarrowerTags’ and ‘getAllSystemNarrowerTags’) got necessary in order to retrieve all sub-concepts of a given tag or II. These methods got implemented outside the realm of this work. After receiving all the tags which could be possibly used as search input, the second part of method ‘searchForResources’ actually searches for resources with found and given tags attached.

```
Hashtable<String, ArrayList<ModelSearcherResult>> foundResourcesPerTag =
      new Hashtable<String, ArrayList<ModelSearcherResult>>();
ArrayList<ModelSearcherResult> foundResourcesForTag =
      new ArrayList<ModelSearcherResult>();

for(String foundTagLabel : allFoundTagLabels){
      foundResourcesForTag = new ArrayList<ModelSearcherResult>();

      for(
            String foundResourceUri :
                  ModelGC.socket.getTagAssignmentsDistinctResource(
                        ModelGC.STRING_EMPTY,
                        ModelGC.STRING_EMPTY,
                        foundTagLabel,
                        ModelGC.TYPE_SHARED)){

            foundResourcesForTag.add(
                  new ModelSearcherResult(
                        true,
                        foundResourceUri));
      }
      […]
      for(
            String foundResourceUri :
                  Model.findResourceUrisByII(foundTagLabel)){

            foundResourcesForTag.add(
                  new ModelSearcherResult(
                        true,
                        foundResourceUri));
      }

      foundResourcesPerTag.put(
            foundTagLabel,
            foundResourcesForTag);
```

```
}
return ModelGC.typeConverter.getDistinctRMResourceSearcherResultsFromArray(
      selectFoundResourceUrisAccordingToSearchOperation(
            foundResourcesPerTag,
            searchOperation));
```

First, method ‘getTagAssignmentsDistinctResource’ implemented by the server framework for DS1 searches for resources by a given tag directly in TS. Afterwards, found resources get stored in new instances of class ‘ModelSearcherResult’. As the whole RM got kept in working memory, searching for resources with regard to II got possible directly within the RM. Method ‘findResourceUrisByII’ implemented in class ‘Model’ returns all resources for a given II. Finally, method ‘searchForResources’ packs all instances of ‘ModelSearcherResult’ and returns search results with regard to given search operations.

Model

As using the method ‘findResourceUrisByII’ got shown above, the body of this method looks as follows:

```
ArrayList<String> result = new ArrayList<String>();
if(ModelIIype.contains(possibleII) == false){
      return result;
}
for(ModelResource resource : resources.values()){
      if(getIIList(resource.resourceUri).contains(possibleII) == true){
            result.add(resource.resourceUri);
      }
}
return result;
```

First, respective method checks whether given parameter actually is a valid II. Second, it requests all resources having given II attached from within the RM with the help of method ‘getIIList’:

```
if(resources.get(resourceUri) != null){
      return resources.get(resourceUri).iIs;
}else{
      return new ArrayList<String>();
}
```

From this chapter it became apparent that the ARSS came to life by implementing, first, the possibility to use results of the RMKMS for querying the RM for resources having certain II stored in their RP and second, the possibility to use certain (sub-) concepts of tags and II to search for resources.

Using the ARSS from within client side got made possible by extending the MC library introduced by this work. Respective library implements requesting the server with message ‘searchForTags’ as well as distributing the results to DS1 client parts.

As shown in figure 20, within the RMKMS available II had to be provided in advance to client side of DS1 in order to enable users to select respective II for triggering II-based searches. From that, ‘getAvailableIIAndRepresentations’ implemented within class ‘Model’ and as such in the ARSS provides the mappings for II types and their respective textual representations for the client. From that, the user will be able to select meaningful II whereas the MC library is able to send defined instances of II to server.

As by this work not only search mechanisms within DS1 got extended but retrieving detailed information upon resources got enabled, the ARSS is able to provide respective information described in the following chapter.

5.2 Retrieving Additional Resource Information

Besides enabling improved search mechanisms by searching with the help of calculated II, the results of the RMKMS get used for additional information in resource summaries in DS1 as well. The RM is able to provide various properties of resources based on usage data including calculated II. From that, contents shown in the ‘Resource Summary Widget’ mockup in figure 22 got based on properties available within RP of certain resources.

Actually, the current implementation of the ARSS is able to provide these kinds of additional information on resources. Information returned when searching for resources already contain additional information on resources and come together with the search results themselves as described above. Respective information upon resources including calculated II get displayed to the user when opening a search result in provided ‘Resource Summary Widget’.

Figure 22: Resource Summary Widget Mockup

Previous chapters focused on the calculation process of KMI for resources and their application in the search process of DS1. The next chapter shall describe evaluations done in the context of this work yielding results of the application of followed approach in real world settings. Afterwards, the overall outcome of this work will be shown.

6 Evaluation

In order to test the feasibility of chosen approach for calculating KMI for digital resources, two evaluations were done subsequently. To elaborate on whether specific II are more or less interesting for the users of DS1 and whether certain II are able to indicate maturity of resources, a questionnaire got handed out to users at CK. This evaluation yielded whether it is useful to present users with resource properties considering the maturity of resources. The results of this evaluation subsequently got employed within the application of DS1 at SA providing the setting for the second evaluation done within this work. The evaluation at SA yielded an analysis of II calculated during the use of DS1. Respective evaluation gave insight into whether II reach sufficient discriminating power for allowing users to decide whether certain resources should be preferred when choosing material for their work. In turn, this evaluation bore whether it is reasonable to follow presented approach for the calculation of KMI.

First, the evaluation at CK will be described before the results of the evaluation at SA will be presented.

6.1 Evaluation at Connextions Kent

The evaluation done at CK was intended to derive whether users of DS1 think certain textual representations of II are capable of indicating certain maturity of resources. From that, it could be derived whether certain II should get used for the application within DS1. Moreover, it provided insight into which II should be considered when representing resources with regard to maturity. The questionnaire in figure 23 got handed out to a subset of seven key-users of DS1 at CK. It asked the users to indicate whether they think chosen II indicate certain maturity of resources. Additionally, the questionnaire asked – independently of whether respective II indicate maturity or not – about possible interests of users in coming to know resources having certain II. The II stated in the questionnaire are equal to calculated II in the sense that they represent textual instances of respective II. As the questionnaire's intention was to find out which kinds of II are both expressing maturity and / or interests of users, only a representative subset of calculable II got presented in the questionnaire.

Please indicate whether you think following **indicators** can give information about the **maturity of a digital resource.** Please also state whether it **would be interesting** for you to have the following **information for** digital resources.	Indicates Maturity	Does not indicate Maturity	I am not sure	Would be Interesting	Would not be too interesting
The resource appears in a search result	3	3	1	5	1
The resource has been viewed a lot by other users	6	1		6	
A resource has been rated frequently by other users	7			6	
A resource is in a collection	1	1	3	5	1
A resource has not been tagged yet (negative indicator)		4	3	2	2
A is part of a collaboratively used collection	3		4	4	1
A resource has just been created (negative indicator)		5	2	4	2
A resource is associated with a lot of persons (through tags etc.)	6	1		5	1
A resource has been rated by a person	2	4	1	7	
A resource has been recommended by the software		2	5	3	3
A resource has been tagged	2	3	1	6	
A resource has been the subject of a discussion among users	6	1		6	
A resource has changed a lot		4	3	3	2
A resource is not part of any collection (negative indicator)		4	3	1	4
A resource has not been rated yet by anyone (negative indicator)		5	2	6	

A resource has not been changed yet by anyone (neg. indicator)	1	5	1		5
A resource has been viewed by many persons	4	2	1	7	
A resource has been used widely by others (e.g. collected, etc)	6			7	
A resource has often been rated high	7			7	
A resource has not been viewed by any person (negative indicator)		4	3	3	3

Figure 23: Intermediate Indicator Questionnaire at Connextions Kent

Six to seven interviewees stated that the maturity of a resource could be expressed when either a resource has been viewed a lot by other users, has been rated frequently by other users, is associated with a lot of persons, has been the subject of a discussion among users, has been used widely by others or has often been rated high. Four users stated that a resource can be considered mature when the resource has been viewed by many persons, three users can derive information on the maturity of a resource when the resource appears in a search result or is part of a collaboratively used collection. Two users each stated that maturity can be derived from the fact that a resource has been rated by a person or has been tagged.

Four to five interviewees considered following information on resources not to express the maturity of resources: the resource either has not been tagged yet, has just been created, has been rated by a person, has changed a lot, is not part of any collection, has not been rated yet by anyone, has not been changed yet by anyone and has not been viewed by any person. Three persons each do not think following facts indicate maturity: a resource appears in a search result and a resource has been tagged. One up to two persons each cannot see an expression of maturity by the fact that the resource has been viewed a lot by other users, is in a collection, is associated with a lot of persons, has been recommended by the software, has been the subject of a discussion among users or has been viewed by many persons.

With regard to the interests of getting to know II-information for resources, six to seven persons stated that it would be interesting to know that a resource has been viewed a lot by other users, has been rated frequently by other users, has been rated by a person, has been

tagged, has been the subject of a discussion among users, has not been rated yet by anyone, has been viewed by many persons, has been used widely by others or has often been rated high. Four to five interviewees stated that if a resource appears in a search result, is in a collection, is part of a collaboratively used collection, has just been created or is associated with a lot of persons they would be keen to get updated. A maximum of three persons declared that it would be interesting to know whether the resource has not been tagged yet, has been recommended by the software, has changed a lot, is not part of any collection or has not been viewed by any person.

It becomes apparent that most users at CK think that II indicate maturity when II are able to convey that either a resource got an assessment (rating) or a resource was used by many (other) persons (viewed, associated, subject of a discussion, used widely and etc.). In turn, users do not consider resources to be indicated by certain maturity when negative II got applied. It was very interesting to see that people would like to have presented information on resources even when thinking that in their opinion certain II do not indicate maturity. Moreover, it can be recognized that in most cases where II are considered to not indicate maturity at least some persons were not sure whether maturity actually is indicated or not. Concluding, these results show that users at CK, first, want to learn and know more about resources and their maturity in DS1, second, consider certain II to indicate maturity and third, are keen to get additional information for resources even if they think certain II do not indicate maturity.

Following the results of the questionnaire, it can be stated that, first, it makes sense for users to get additional information for resources possibly indicating resources' maturity, second, it is reasonable to at least present discriminating II to users and third, it is necessary to foster explaining the concept of maturity to users as well as to better support the 'Knowledge Maturing Process' by tool support indented by this work.

Subsequently, results of this questionnaire got considered for the application of this work in the SA case so that chosen II (either indicating maturity or being considered as interesting) got presented to users, considering participants at SA likely to be akin to users at CK in terms of using DS1 similarly.

6.2 Evaluation at Structuralia

The evaluation period for DS1 at SA lasted for around two and a half month. During this period DS1 got used 79 days according to the UE streams recorded. Updating of the resources' RP was done once per night, hence didn't consider changes during the daytime when users most likely used the system. As the evaluation at CK yielded certain II to be either more interesting or indicating maturity as such, a subset of calculated II got presented in the 'Search Widget'.

The pre-analysis of UE streams from DS1 applied at SA revealed that participants didn't use II presented in the 'Search Widget' in order to search for resources. Various reasons have been identified; tough they couldn't be directly traced back to respective usage data. First, course instructors didn't make participants aware of the II search functionality; hence participants neither felt capable of using respective kind of search nor felt supposed to do so. Second, participants rather got advised to use collections in order to retrieve important learning material which got put there by advisors. From that, again participants didn't feel supposed to use the extended search. Third, it is supposed that update times for the calculation of II didn't reflect the actual need of timely accurate II in order to represent actual states of resources.

From that, it was reasonable for this work to manually analyze the results of II calculations as possibly applied questionnaires or interviews asking for the searching behavior of participants would not have been feasible for answering whether II within the extended search process help in defined KMA. Hence, the calculation process of II got extended so that during the evaluation period calculated II could be recalculated in a day wise manner. Past UE streams got used to calculate II for each day back in the evaluation period. This allowed for analyzing resources with regard to maturity represented by II in order to gain insight into the distribution of II over time as well as into which resources reached certain II.

In following chapters different aspects of the evaluation shall be described so that it becomes apparent whether the approach of this work is able to support users in finding mature resources as well as possible interesting ones and probably several other KMA.

Though the whole set of resources was considered during evaluation, resources helping to describe calculation results got chosen randomly from respective set of resources collected

and created during the evaluation phase of DS1 at SA. As could be derived from results of the evaluation at CK, discriminating II are considered to be indicating resources' maturity as well as are considered to be interesting for users. In consequence, mainly those II will be used for presenting results here. Moreover, showing results for discriminating II is reasonable from the fact that the calculation of discriminating II considers other resources' maturity states too.

6.2.1 Basic Resource Analysis

From the analysis of UE streams and calculated II, following interesting information can be derived in advance: Altogether 339 resources were brought into or created in the system by users and course leaders. 50 resources had an attached discussion at the end of the evaluation period (which got subsumed under the total number of resources). 31 resources got filed in collections. 83 resources received an assessment through one or the other way sometime during the evaluation period, 78 resources got a rating, 60 resources got rated high at least once as well as 84 resources got viewed at least one time.

6.2.2 Maturity of Resources

It was preliminary interesting whether calculated II are able to tell anything about the maturity of certain resources at hand. As certain II suggest, e.g. that a resource was used or changed more often than others, II were supposed to be able telling whether certain resources could be more mature than others. As the maturity of a resource in this work mainly gets expressed with regard to the resource's UE frequencies in certain UE groups in relation to other resources, assumptions about the maturity of a resource are possible by inspecting resources' reached discriminating II. In the following, resources can be shown with regard to reached discriminating II over time eliciting that some resources' II remained stable over time, changed a lot or decreased or increased in their amount. Only a subset of resources reached specific II as well as reached different amounts of discriminating II over time. This yields, that resources can be very well described with regard to their maturity in relation to other resources in the system. Calculated II were able to suggest increasing or decreasing maturity as well as suggest changing maturity of resources. In the following, some resources reaching chosen discriminating II get presented. In following figures, the x-axis depicts the actual number of days during the evaluation period whereas the number of reached II at certain points in time will be shown on the y-axis respectively.

6.2.2.1 Resources with Stable Numbers of Intermediate Indicators

In this case, some resources could be filtered where the daily amount of discriminating II remained very stable over time. As figure 24 shows, respective resource reached eight II at a certain point in time, could increase the amount to ten after three days and hold those characteristics until the end of the evaluation period. From the maturity expressed by reached II, like became standard, associated with many persons or was prepared for audience many times, it can be derived that the resource reached a very high level of maturity within the breadth of resources.

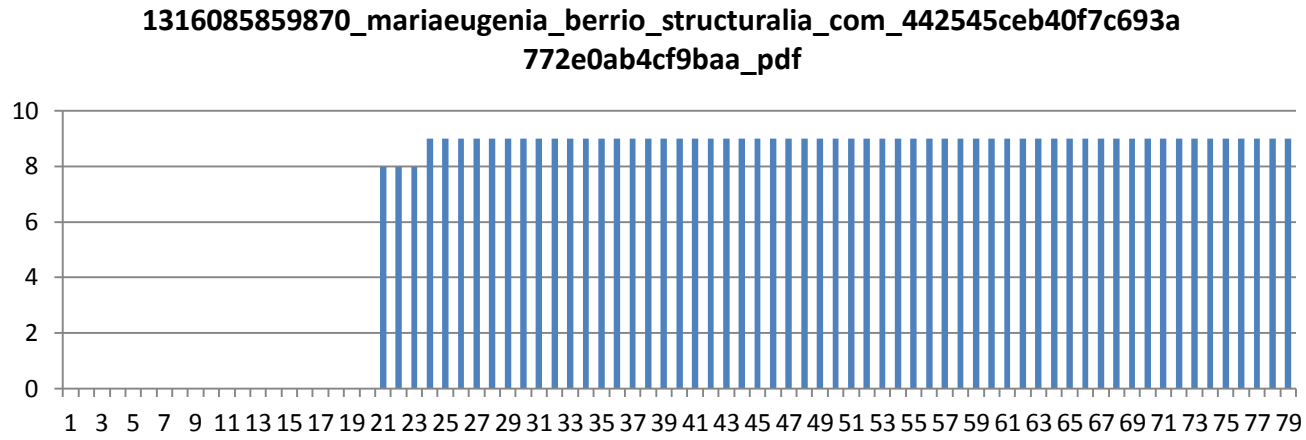

Figure 24: Resource with Stable Number of Intermediate Indicators

From figure 25 it becomes apparent that some resources where able to reach a number of discriminating II at certain times but had to be downgraded by the calculation process during their use in the system. This obviously happened during days where a lot of other resources got used and edited by users of the system, hence other resources in the focus of the system got improved in maturity and pushed back those resources with currently high maturity. As figure 25 suggests, respective resource could catch up and reach its initial II again. In relation to the previous resource, this one wasn't able to reach as many indicators as the previous one, e.g. only reached used widely, viewed a lot and reached high awareness in the system at its best times.

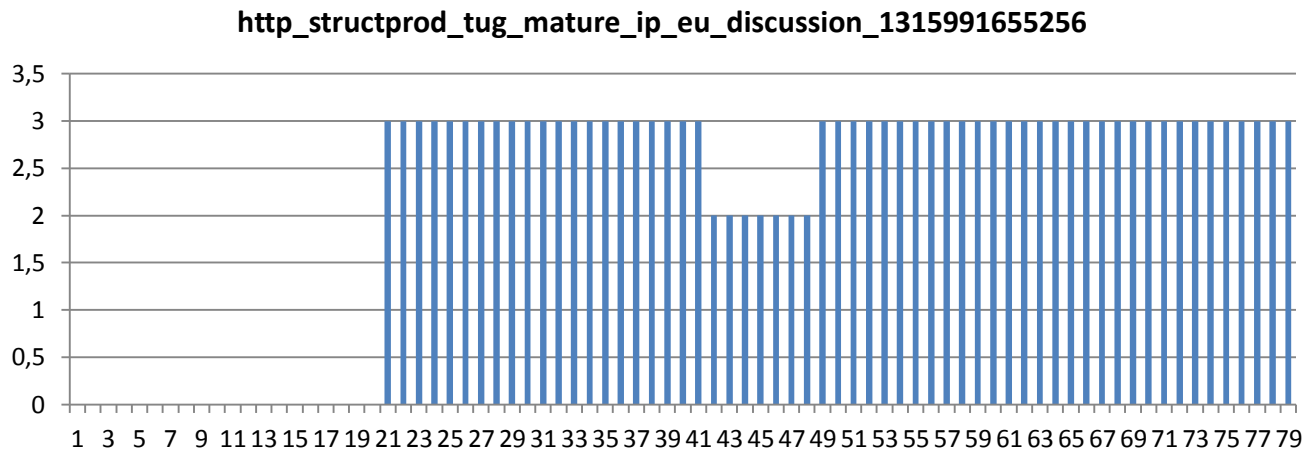

Figure 25: Resource with Stable Number of Intermediate Indicators

6.2.2.2 Resources with Decreasing Numbers of Intermediate Indicators

Resources shown in next figures were able to reach a certain level of maturity with regard to attached discriminating II but lost this status during their use in the system. They had to dispense reached II to other resources getting more mature with regard to their use.

The resource shown in figure 26 was able to reach changed a lot and used widely during the initial days of the evaluation period. Subsequently, respective resource had to emit used widely first and lost changed a lot after ten days, indicating that the resource wasn't edited that often anymore. As the resource didn't reach any other II the next days, the resource apparently wasn't mature enough to be used in the online training course on a frequent basis further on.

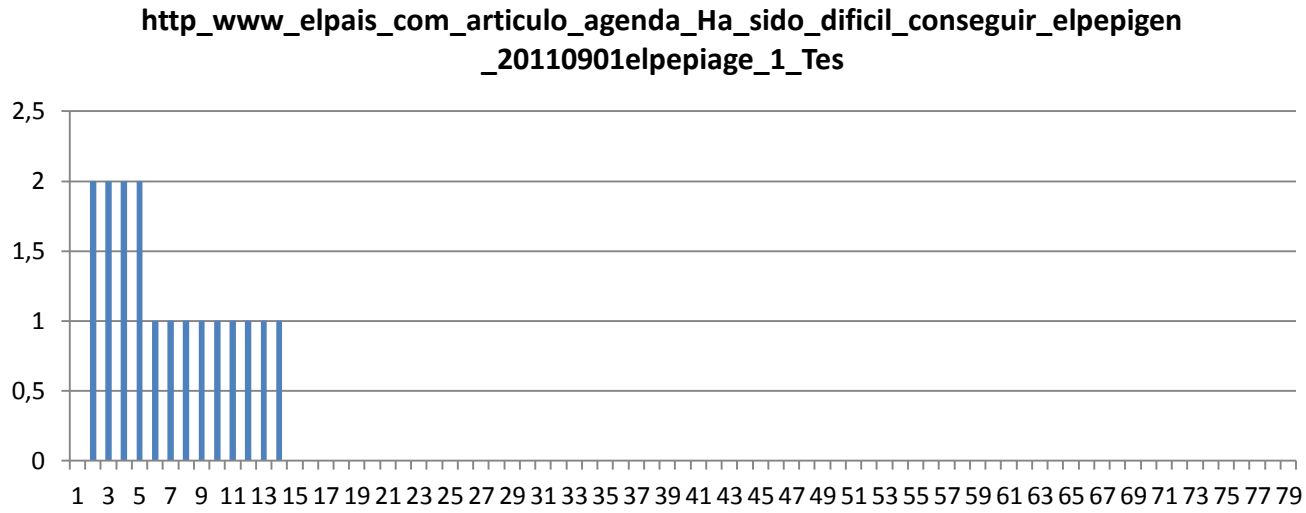

Figure 26: Resource with Decreasing Number of Intermediate Indicators

Figure 27 depicts a resource which first reached some discriminating II and was even able to add good II during its use peaking during the fourteenth day. At the day when the resource reached most of available discriminating II, it was often recommended and used widely and in relation to other resources had many entries. Moreover, the resource got changed by many adding or deleting steps indicating that the resource got improved in maturity a lot until this day. Apparently, the resource lost most of its II during the next days but could keep the used widely indicator until the end of the systems use. This suggests that the resource reached a certain level of maturity, supposing that the resource stayed very popular in the system; although in relation to other resources it couldn't catch up with resources getting even better.

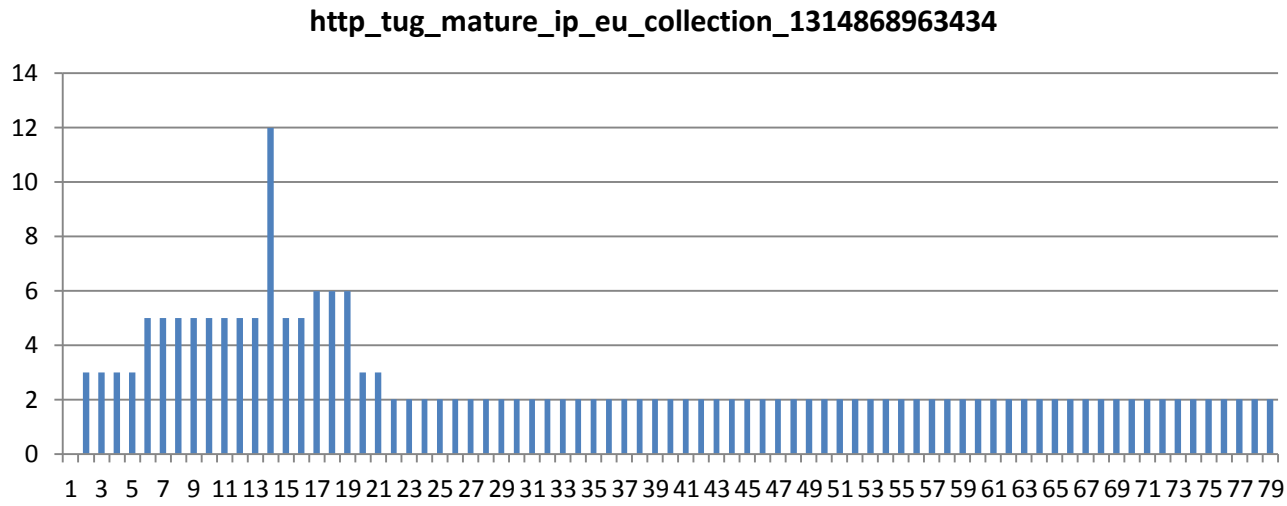

Figure 27: Resource with Decreasing Number of Intermediate Indicators

Similarly to the previous resource, resource in figure 28 could reach a considerable amount of discriminating II but lost some over time. In opposition to the previous one, this resource was able to reach and hold more II over time, e.g. often recommended, viewed by different persons and became standard. This resource just lost II indentifying increasing the maturity of a resource like often rated high and shared a lot within community but could keep most of the other II. From that, although the resource decreased its number of discriminating II, it becomes apparent that the resource didn't completely lose its maturity in relation to others.

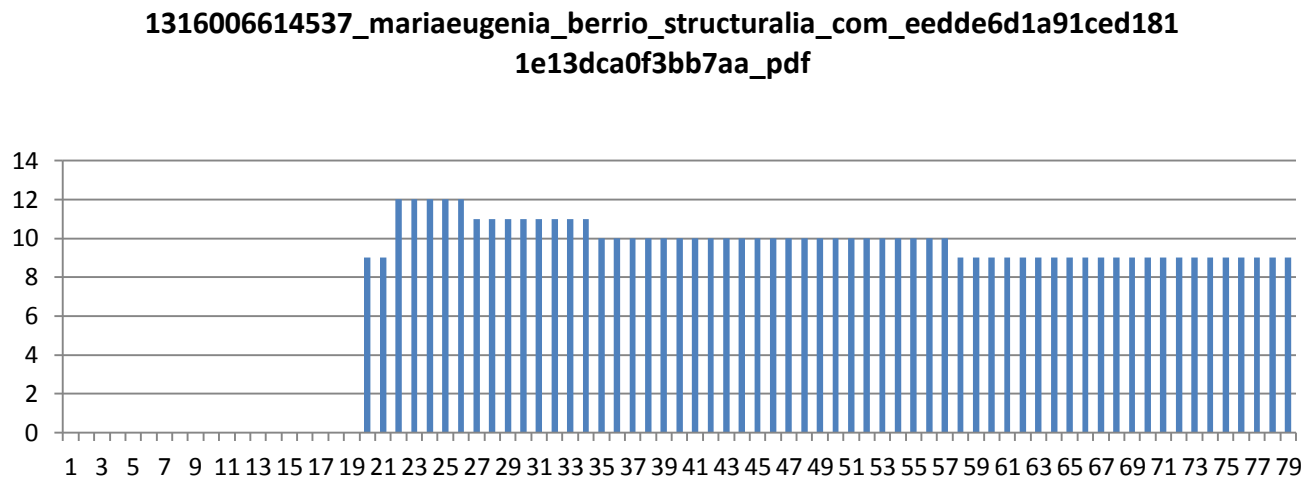

Figure 28: Resource with Decreasing Number of Intermediate Indicators

6.2.2.3 Resources with Increasing Numbers of Intermediate Indicators

Resources shown in next figures step by step increased their number of discriminating II. This might suggest that those resources continually developed their maturity within the system and were able to increase their visibility in the system until the end of the evaluation period.

The resource shown in figure 29 reached the II became standard during the end of the system's use, moreover was able to attain viewed a lot, often rated and assessed many times during its use. The frequent change of discriminating II during the beginning of the resource's visible maturity development suggests that the resource had to undergo various changes and has had some competition with other resources in the system concerning the attainment of certain II.

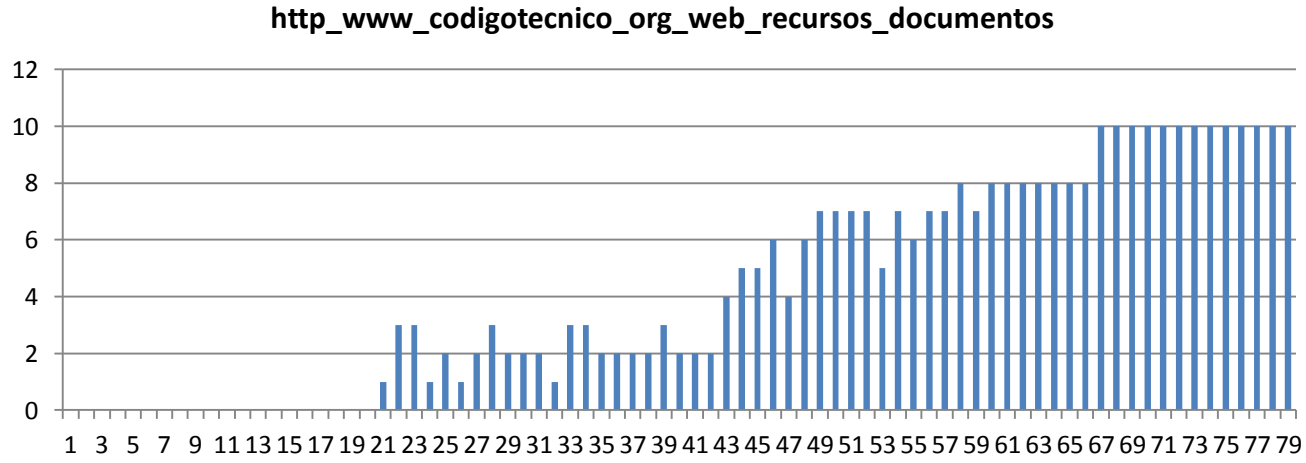

Figure 29: Resource with Increasing Number of Intermediate Indicators

The resource shown in figure 30 first reached the changed a lot II and got associated with many persons during evaluation. The changed a lot II as well as the early reached a lot collection collaboration II could be held until the end of the evaluation suggesting that the resource got steadily improved in maturity over time. Other II like often organized in collections couldn't be hold, though the number of discriminating II remained stable until the end. This resource as well as some other resources was able to reach a number of discriminating II, denominating them of being part of a selection of valuable resources within the system with regard to their maturity.

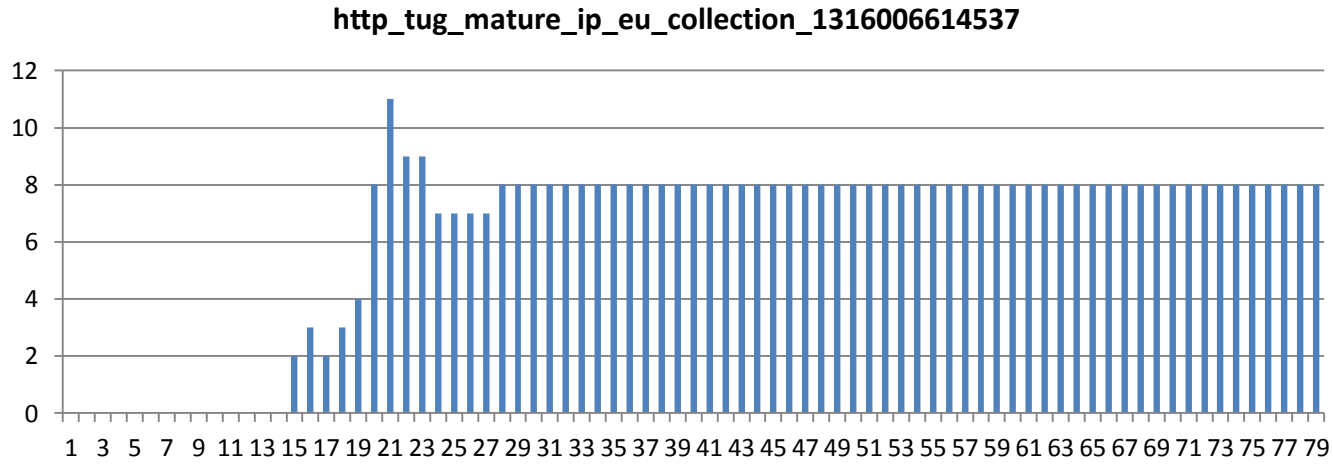

Figure 30: Resource with Increasing Number of Intermediate Indicators

6.2.2.4 Resources with Changing Numbers of Intermediate Indicators

Resources shown in next figures steadily changed with regard to the number of attained II. Due to the evolutionary evolvement of the system's content, some resources received ongoing change, hence were not always able to reach certain maturity.

The resource in figure 31 reached the became standard II from time to time and lost it as often due possible system's focus on other resources at certain times. Despite that, the resource always was able to keep the often selected from others and used widely II which depicts that the popularity of the resource was very high.

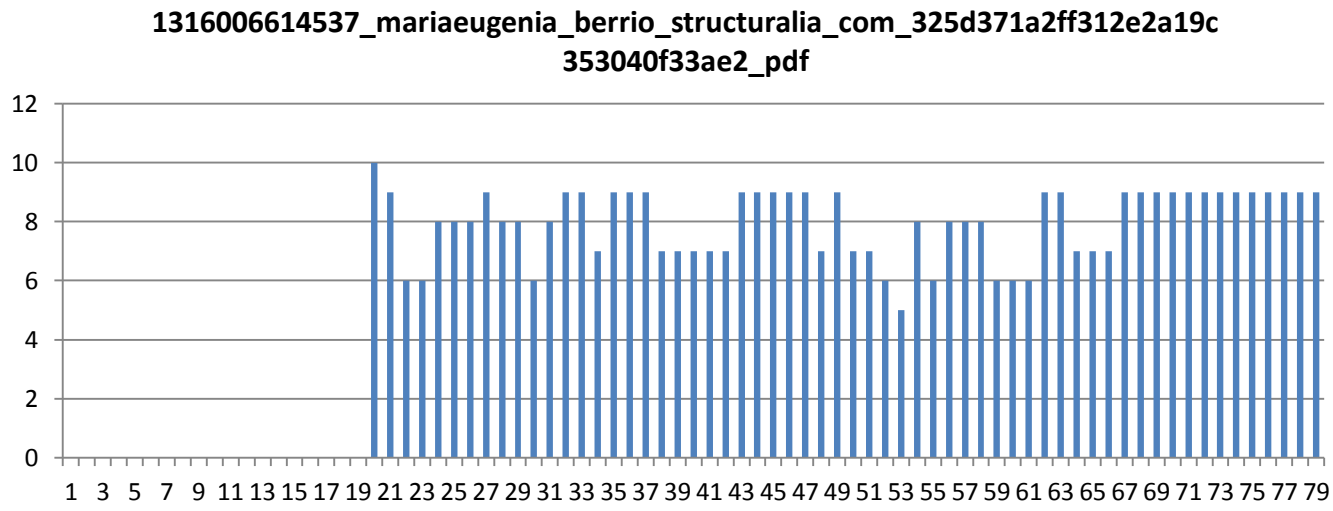

Figure 31: Resource with Changing Number of Intermediate Indicators

Shown in figure 32, the next example of a resource with changing number of II shows that although a resource hasn't been able to hold attained discriminating II over a longer period of time, it was possible for resources to catch up at certain times and reach the one or the other II (again). In this example, the resource reached changed by many adding or deleting steps in its early use and could attain used widely further on in time for a small number of days.

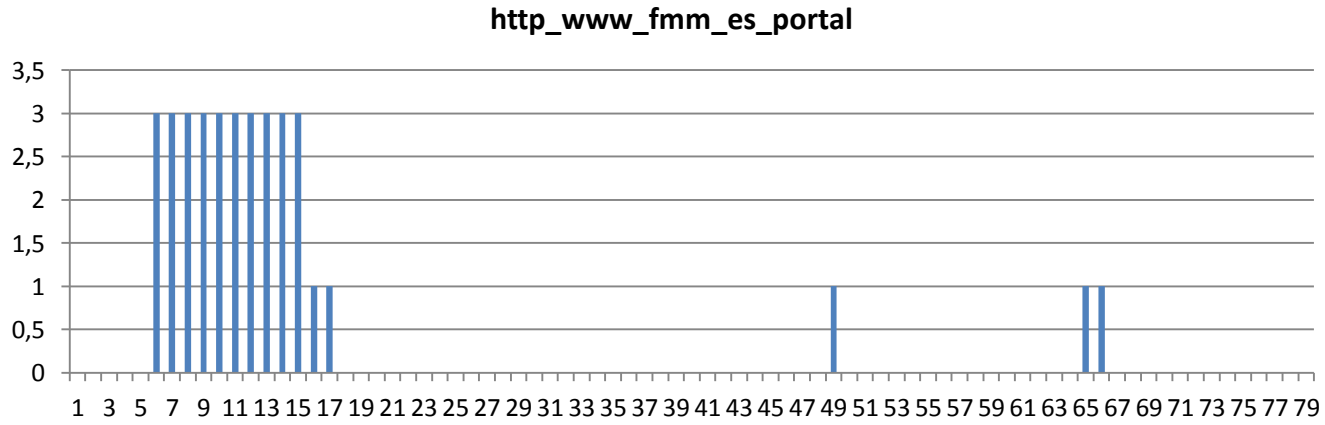

Figure 32: Resource with Changing Number of Intermediate Indicators

As well as in the previous example, resource in figure 33 reached the used widely II from time to time though was not able to hold that II over time. As this resource was not denominated by any other discriminating II, it is possible that the resource was for example put into the system some time, was possible tagged with a very general keyword, hence was found in many search results but didn't receive any viewing or changing II. From that, the resource obviously reached a discriminating II from time to time but couldn't compete with other more mature resources in the system because of its lower maturity derived from its use.

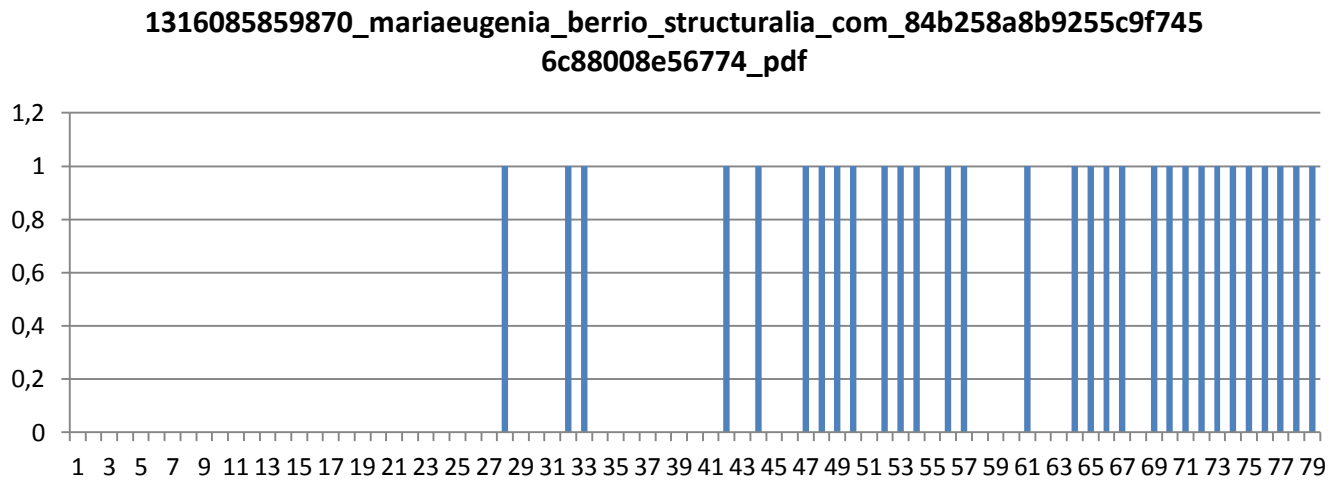

Figure 33: Resource with Changing Number of Intermediate Indicators

After showing evaluation results focusing on respective resources, next chapter will show the distribution of discriminating II with the help of the number of resources reaching respective indicators over time.

6.2.3 Distribution of Indicators Over Time

It was also interesting to exploit how many resources reached a certain discriminating II during the evaluation period. Hence, calculated II were analyzed with regard to the number of resources reaching respective II per day. In the following results for a subset of available II will be shown.

6.2.3.1 Became Standard

This II was able to exploit resources which either became very popular in the system or reached a high level of maturity within the resources in the system. From figure 34 it becomes apparent, that only view resources were able to reach this II per day.

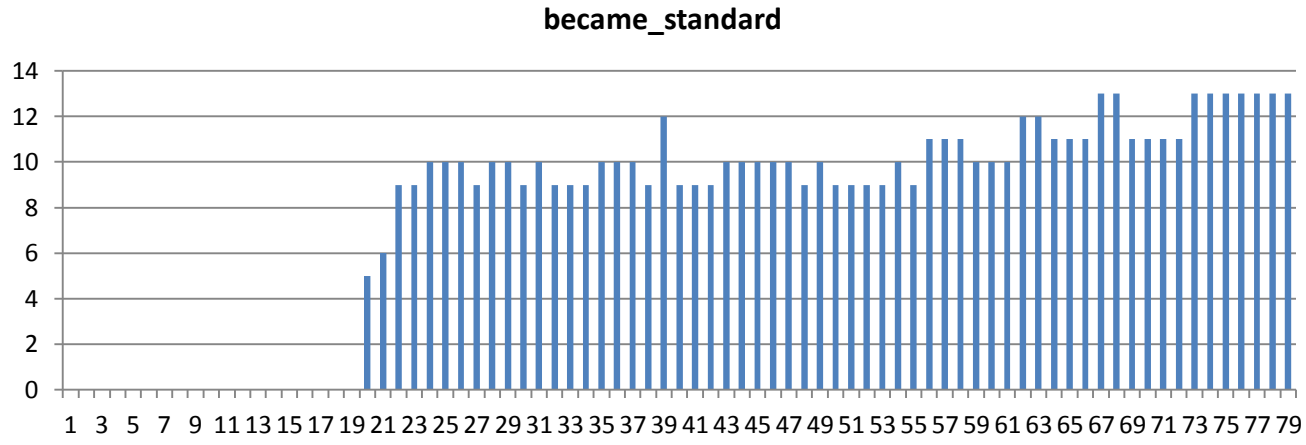

Figure 34: Intermediate Indicator Distribution for Became Standard

6.2.3.2 Reached High Awareness

The number of resources reaching this discriminating II increased during the evaluation period starting at the time the use of the system continually evolved. Figure 35 depicts that an average of ~30 resources got this II attached. Supposing, that only a limited number of resources continually got used very frequently throughout the online training course, shown numbers for resources attaining this II are reasonable.

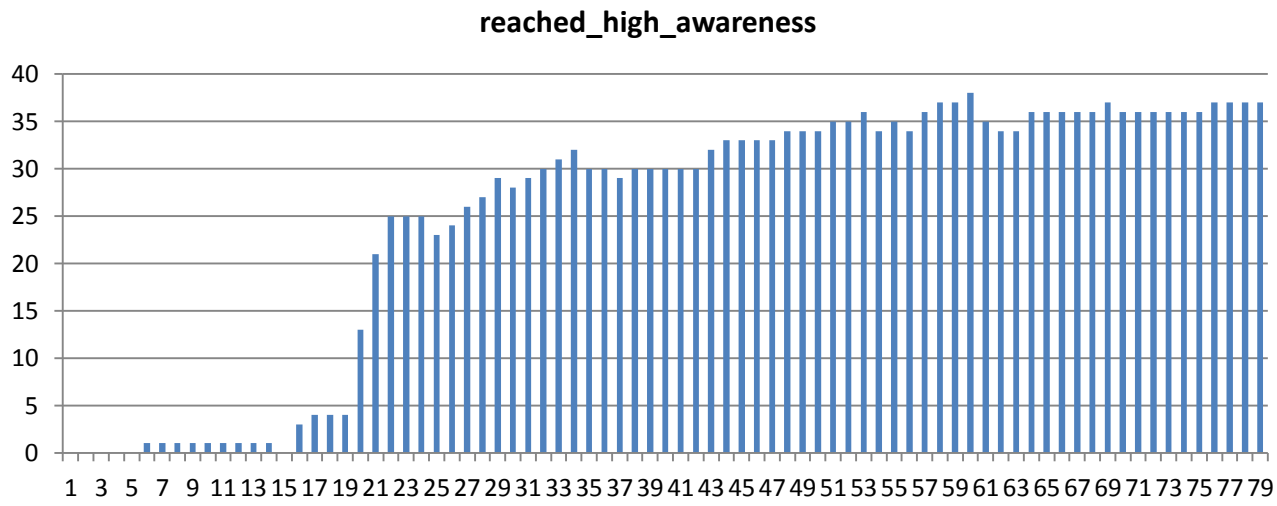

Figure 35: Intermediate Indicator Distribution for Reached High Awareness

6.2.3.3 Changed by Many Adding or Deleting Steps

Due to the course arrangement, only a limited number of resources got improved in maturity from within the whole set of resources brought into the system. II shown in figure 36 makes visible that even from within the limited subset of resources only certain resources could reach respective II. It also gets apparent that at certain points in time the number of resources with this II increased with regard to the increasing total number of resources brought into the system.

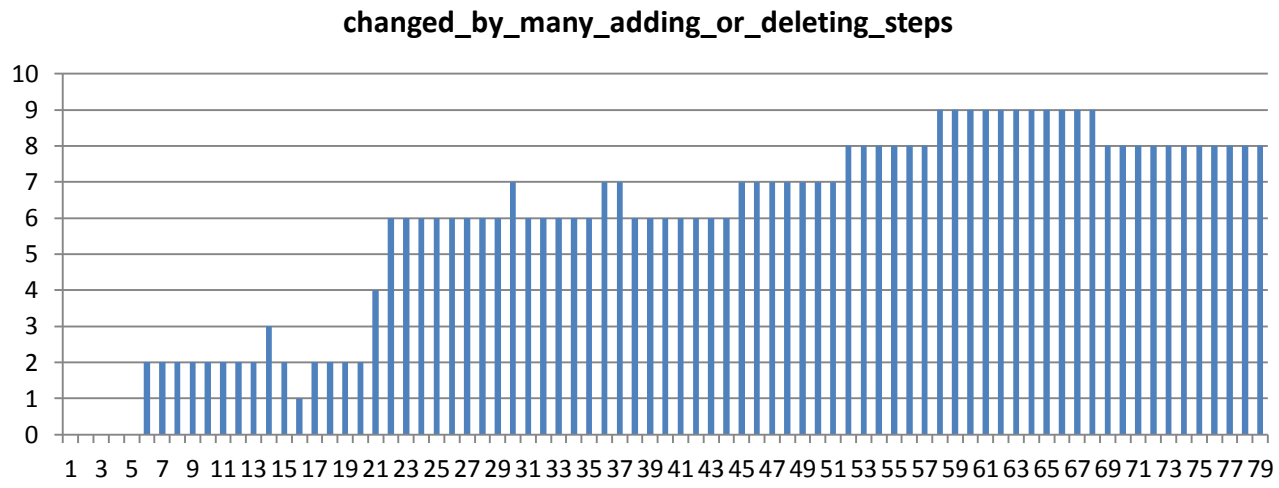

Figure 36: Intermediate Indicator Distribution for Changed by Many Adding or Deleting Steps

6.2.3.4 Shared within Community

This II does not relate to other resources' properties in the system, hence got defined as positive II (see above). Showing the distribution of this II in figure 37 elicits that altogether 180 resources from within the system got shared with the community. Taking into account that its discriminating counterpart – discriminating II shared a lot within community – at its best just reached a maximum of seven resources per day, this arrangement of II depicts that the approach taken by this work for the calculation of II – taking usage characteristics of resources into account – is able to discriminate mature resources from within the whole set of resources.

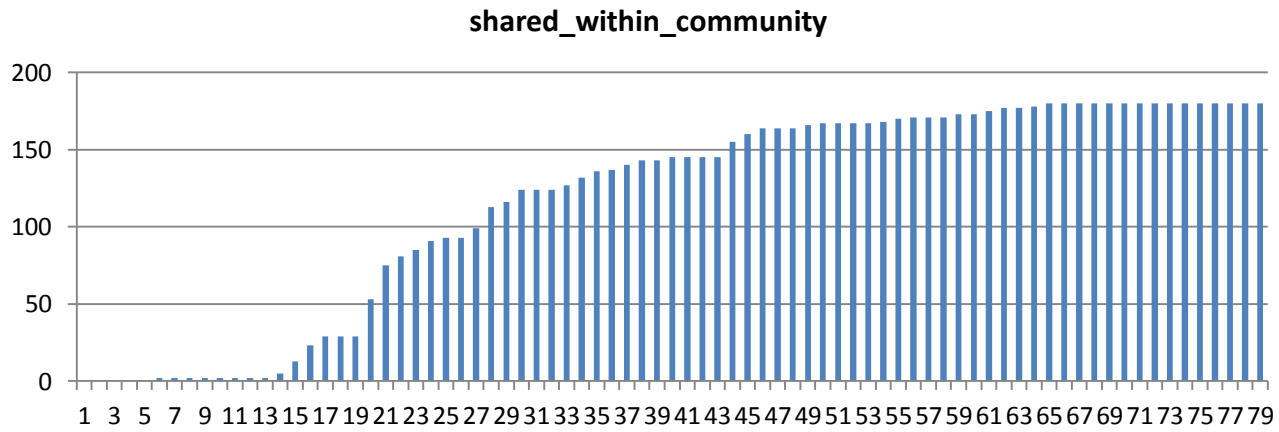

Figure 37: Intermediate Indicator Distribution for Shared with Community

6.3 Selection of Mature Resources

From the set of resources reaching a higher maturity than others expressed by reached discriminating II, three resources got picked in order to present characteristics of mature

resources. These resources at least reached discriminating II once during the evaluation period.

The resource ‘1316085859870_mariaeugenia.berrio@structuralia.com_442545ceb40f7c693a772e0ab4cf9baa.pdf’ reached became standard, often organized in collections, high awareness, viewed a lot, associated with many person, often recommended, used widely and was prepared for audience many times.

The resource ‘http://faq.cype.es/C3D’ reached became standard, often organized in collections, high awareness, viewed a lot, used widely, often recommended, assessed many times, changed by many adding or deleting steps, often rated high, shared a lot within community, tagged a lot, appears in many search results and associated with many persons.

The resource ‘http://tug.mature-ip.eu/collection/1316085859870’ reached changed by many editing or deleting steps, high awareness, associated with many persons, used widely, viewed a lot, often organized in collections, a lot collection collaboration and changed a lot.

As the usage-based calculation process of KMI for resources and its respective integration directly in DS1 yielded several insights and implementations, next chapters will focus on this work’s results before an outlook to possible future work will be given.

7 Results

The aim of this work was to elaborate on the feasibility of a usage-based approach for calculating KMI for resources so that results directly could be used by users of DS1 where developments done got applied and integrated to support KMA. Together, this yielded a framework (MC library) enabling the client side of DS1 collecting necessary TI / UE, a new client server communication approach within DS1 providing the bases for using implemented services as well as making DS1 useable in real world settings, a KMS (ULKMS) accepting and storing retrieved usage data, a KMS (RMKMS) calculating II and KMI for resources relying on traced usage data and a service (ARSS) providing extended search possibilities within DS1 based on calculation results.

The results of this work show that the calculation of KMI for resources with the help of usage data within a POLME indeed is feasible when taking into account following considerations. First, not all chosen resource-related KMI can be calculated directly but with introducing an intermediate basis upon which a mapping to defined KMI gets possible. This roots in the fact that often UE available within a specific POLME possibly cannot reflect given semantics of chosen KMI so that a single UE group or even a single UE could be directly mapped to a single KMI. Second, introduced II are more or less more representative with regard to what the outcome of the calculation process done in this work reflects, i.e. meaningful characteristics of resources used in DS1 which in turn get able to be used by users of respective POLME. As KMI got defined independently of concrete instances of POLME, using II instead of KMI for the actual application in users' domains becomes reasonable even though the mapping of II to KMI seems to be appropriate in the context of this work. Third, using frequency-based thresholds for assigning concrete discriminating II to resources does not consider the actual weight of certain TI / UE within DS1. Although, this lays out the basis for further application of various imaginable weighting schemes with regard to valuing different KMA represented by TI / UE differently. Fourth, this work does not take into account any demands with regard to whether certain TI consider either in- or decreasing the maturity of underlying resources but considers the sum of undergone TI / UE to be changing respective resources' maturity equally. Hence, possible increasing, decreasing, changing or stable maturity of resources gets expressed by the total number of concrete instances of KMA at certain points in time in relation to other resources.

From the results of the evaluation done at CK, it is helpful to provide users of a POLME more information on used resources with regard to maturity. Information provided and represented by calculated II had been perceived providing advice on the maturity of respective resources as well as had been wanted to be used to gain additional knowledge about resources.

Applying these results in another evaluation done at SA, yielded that calculated II enable users to pick resources from within DS1 reaching certain maturity states at certain points in time. Examining calculated II for each day during respective evaluation period, revealed that there indeed exist resources being developed over time so that they get more mature but also resources exist which decrease, remain stable or change a lot in maturity over time with regard to calculated II. II as frequency-based maturity measures upon resources relying on the traces of KMA represent the link of respective KMA to defined subset of KMI; hence make 'Knowledge Maturing' visible so that results can be used by users.

Together, results of this work support users in KMA 'find relevant digital resources', 'keep up-to-date with organization-related knowledge', 'familiarize oneself with new information' and 'assess, verify and rate information' as using calculated II in search processes and for additional resource information enables knowledge workers, first, to more specifically and more sophisticatedly search for resources on higher maturity levels, second, to recognize changes in the maturity of resources, third, to become aware of resources recently developed in maturity and fourth, to make relevance judgments upon resources with regard to maturity for using them in certain work processes.

Beside supporting several KMA, both calculated II and KMI mapped accordingly provide input for KMGA 'make aware', 'recommend, suggest & advice', 'irritate & challenge', 'evaluate & assess results' and 'monitor activities & progress' as, first, provided II help to inform about resource developments as well as to recommend resources reaching certain states of maturity and second, on II based KMI help to evaluate maturity developments from a resource perspective as well as to gain insight in KMA within the POLME with the help of automatically provided aggregation and compaction of respective activities into dedicated maturity measures.

Considering the environment created by this work, i.e. a client side library in combination with two KMS (plus an extended search service) and a RM in the background, this work

provides additional value with regard to possibly further developments in the realm of usage-based ‘Resource Modeling’ as well as further attempts to improve the support for knowledge workers in dealing with work-related content for learning or daily routine tasks.

8 Outlook

As this work followed a usage-based approach for calculating KMI for resources wherein certain TI / UI groupings had to be done enabling the calculation of dedicated measures (II) mapped to KMI, future work has to elaborate on whether respective mappings hold in different contexts, i.e. different POLME, as available traces of KMA and the inherent meaning of KMI could possibly change or interpreted differently. Further, it is necessary to evaluate on whether established UE groups accurately reflect the meaning of certain KMI being mapped to resulting II. This will be reasonable as in parallel to this work within the MATURE project already certain efforts have been undertaken in order to map KMI to possible TI of concrete instantiations of various POLME.

This work tried to express the maturity of resources with regard to the frequency of instances of KMA applied to resources. It remains open examining whether the frequency of undertaken KMA alone can be used as vital parameter for calculating evidence of changes in maturity, although this work showed the feasibility of used approach.

Having in mind that calculated II only depict resources which either have no UE, one UE or more UE than other resources in respective UE groups – hence resources with UE frequency numbers in between currently are not selectable, e.g. in the ‘Search Widget’ –, it would be interesting to see whether users would like to be presented with resources actually not reaching certain II but coming close to the maturity of currently (with the help of II) retrievable resources. For that, introducing expiry dates for reached II possibly would make sense so that resources from time to time dispensing discriminating II to other resources nevertheless will have a chance to be selected for an extended period of time. In combination with displaying the history of reached indicators, this would allow for a more sophisticated selection of resources upon calculated maturity measures.

This work builds on triggered UE in order to calculate KMI but does not take into account any wider time frames (than the actual point in time when a UE occurs) wherein possible UE could happen. Using not only the independent sequence of UE but respective sequences together with (possibly overlapping) time frames could yield more insight into the actual development of underlying resources as well as allow for the possibility to calculate an extended set of KMI.

As this work only considers digital resources for the calculation of KMI, it would be possible to calculate KMI for other knowledge manifestations too. For cognifacts, e.g. persons, this could be done with the help of already implemented UM based on this work. Moreover, the RMKMS together with the RM could be used for more sophisticated support in dealing with resources in a POLME when taking into account the RM's possibility to describe resources in the system with regard to various (usage-based) properties. Using presented usage-based approach could help to, e.g. give users live-insight into developments of resources while working on certain tasks. When it comes to supporting the user with detailed information on resources she works with, several possible filters upon UE characteristics applied to resources could be thought considering, e.g. possibly defined user tasks and actual work contexts. As there exist KMI not necessarily relying on usage data, i.e. quality-based KMI, calculation of such indicators could be tied to this work too.

The calculation of KMI has to be checked whether it is possible to automatically detect the actual phase within the 'Knowledge Maturing Process' wherein the resource under focus possibly resides. This would allow for presenting resources in the system with regard to the actual 'Knowledge Maturing Phase' both for users and domain experts evaluating an organization with regard to its maturity status at least from the perspective of explicit knowledge contained in digital artifacts.

Bibliography

A. Agostini, S. Albolino, G. De Michelis, F. De Paoli, and R. Dondi. Stimulating knowledge discovery and sharing. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, 2003.

G. Attwell. The personal learning environments - the future of elearning? *eLearning Papers*, 2, 2007.

G. Attwell, J. Bimrose, A. Brown, and S. A. Barnes. Maturing learning: Mashup personal learning environments. *Journal of Knowledge Management*, 388:78-86, 2008.

S. A. Barnes, J. Bimrose, C. Bradley, A. Brown, D. Feldkamp, P. Franzolini, A. Kaschig, M. Kohlegger, C. Kunzmann, J. Magenheim, R. Maier, T. Nelkner, S. Nikles, U. Riss, A. Sandow, A. Schmidt, M. Shuttleworth, and S. Thalmann. D1.1 results of the ethnographic study and conceptual knowledge maturing model. Deliverable, 2009a.

S. A. Barnes, J. Bimrose, C. Bradley, A. Brown, D. Feldkamp, P. Franzolini, A. Kaschig, M. Kohlegger, C. Kunzmann, J. Magenheim, R. Maier, T. Nelkner, S. Nikles, U. Riss, A. Sandow, A. Schmidt, M. Shuttleworth, and S. Thalmann. D2.1 pedagogical and usability foundations and concept for a PLME. Deliverable, 2009b.

S. A. Barnes, C. Bradley, A. Brown, J. Cook, A. Kaschig, C. Kunzmann, J. Magenheim, R. Maier, A. Mazarakis, A. Ravenscroft, U. Riss, A. Sandow, and A. Schmidt. D1.2 results of the representative study and refined conceptual knowledge maturing model. Deliverable, 2010.

S. A. Barnes, J. Bimrose, A. Brown, A. Kaschig, C. Kunzmann, T. Ley, R. Maier, J. Magenheim, A. Mazarakis, A. Sandow, A. Schmidt, and P. Seitlinger. D1.3 results of in-depth case studies, recommendations and final knowledge maturing model. Deliverable, 2011.

C. Bereiter and M. Scardamalia. Cognitive coping strategies and the problem of 'inert knowledge'. In *Thinking and Learning Skills*. LEA, 1985.

F. Blackler. Knowledge, knowledge work and organizations: An overview and interpretation. *Organization Studies*, 16:1021-1046, 1995.

V. Blaževic, S. Braun, R. Brun, H. Eichner, A. Martin, and R. Woitsch. D5.3 infrastructure and integrated 1st mature system prototypes. Deliverable, 2010.

C. Bradley, S. Braun, R. Brun, J. Cook, K. Hinkelmann, B. Hu, C. Kunzmann, T. Ley, A. Martin, A. Mazarakis, T. Nelkner, A. Ravenscroft, U. Riss, A. Schmidt, K. Schoefegger, B. Thoenssen, N. Weber, and H. F. Witschel. D2.2/d3.2 design and delivery of demonstrators of PLME / OLME and tool wrapper infrastructure. Deliverable, 2010.

S. Braun and A. Schmidt. Wikis as a technology fostering knowledge maturing: What we can learn from wikipedia. In *7th International Conference on Knowledge Management (IKNOW '07), Special Track on Integrating Working and Learning in Business (IWL)*, 2007.

S. Braun, A. Schmidt, and V. Zacharias. Soboleo: vom kollaborativen Tagging zur leichtgewichtigen Ontologie. In *Mensch & Computer - 7. Fachuebergreifende Konferenz - M&C*, 2007.

V. Braun, D. Czech, B. Fletschinger, S. Kohler, and V. Lueber. Motivation und Anreize in informellen Lernprozessen beim Thema Wissensmanagement. Project thesis, 2009.

J. S. Brown and R. P. Adler. Minds on fire: Open education, the long tail, and learning 2.0. *Educause Review*, 43:16-32, 2008.

P. Brusilovsky and E. Millán. User models for adaptive hypermedia and adaptive educational systems. In *The Adaptive Web*. Springer, 2007.

G. Buchanan. Frbr: enriching and integrating digital libraries. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2006.

R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331-370, 2002.

P. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. In *The Semantic Web: Research and Applications*. Springer, 2005.

T. H. Davenport, S. L. Jarvenpaa, and M. C. Beers. Improving knowledge work processes. *Sloan Management Review*, 37:53-65, 1996.

C. Doctorow. Metacrap putting the torch to seven strawmen of the meta-utopia. Online 2012-04-05, 2001. URL http://www.well.com/ doctorow/metacrap.htm.

S. Downes. Resource profiles. *Journal of Interactive Media in Education*, 5, 2004.

D. Feldkamp, K. Hinkelmann, and B. Thoenssen. Kiss knowledge-intensive service support: An approach for agile process management. In *Advances in Rule Interchange and Applications*. Springer, 2007.

G. Fischer, J. Grudin, R. McCall, J. Ostwald, D. Redmiles, B. Reeves, and F. Shipman. Seeding, evolutionary growth and reseeding: The incremental development of collaborative design environments. In *Coordination Theory and Collaboration Technology*. 1996.

N. Henze and W. Nejdl. Logically characterizing adaptive educational hypermedia systems. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, 2003.

D. Hung. Forging links between communities of practice and schools through online learning communities: Implications for appropriating and negotiating knowledge. *International Journal on E-Learning*, 1:23-33, 2002.

A. Jameson. Adaptive interfaces and agents. In *The Human-Computer Interaction Handbook. L. Erlbaum Associates Inc.*, 2003.

T. D. Jong, M. Specht, and R. Koper. Contextualised media for learning. *Journal of Educational Technology and Society*, 11:41-53, 2008.

A. Kaschig, R. Maier, A. Sandow, M. Lazoi, S. A. Barnes, J. Bimrose, C. Bradley, A. Brown, C. Kunzmann, A. Mazarakis, and A. Schmidt. Knowledge maturing activities and practices fostering organisational learning: Results of an empirical study. In *Sustaining TEL: From Innovation to Learning and Practice 5th European Conference on Technology Enhanced Learning, EC-TEL 2010*, 2010.

E. K. Kelloway and J. Barling. Knowledge work as organizational behavior. *International Journal of Management Reviews*, 2:287-304, 2000.

D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37:18-28, 2003.

M. Kohlegger, R. Maier, and S. Thalmann. Understanding maturity models. results of a structured content analysis. In *9th International Conference on Knowledge Management (I-KNOW '09)*, 2009.

B. Kump, T. Ley, K. Schoefegger, N. Weber, D. Theiler, A. Schmidt, S. Braun, M. Ramezani, J. Hagelauer, S. Brandner, H. F. Witschel, B. Hu, and T. Nelkner. D4.3 maturing services prototype v2. Deliverable, 2011.

T. Ley, S. N. Lindstaedt, K. Schoefegger, P. Seitlinger, N. Weber, B. Hu, U. Riss, R. Brun, K. Hinkelmann, B. Thoenssen, R. Maier, and A. Schmidt. D4.1 maturing services definition. Deliverable, 2009.

T. Ley, A. Ulbrich, P. Scheir, S. N. Lindstaedt, B. Kump, and D. Albert. Modeling competencies for supporting work-integrated learning in knowledge work. *Journal of Knowledge Management*, 12:31-47, 2008.

A. Lih. Wikipedia as participatory journalism: reliable sources? metrics for evaluating collaborative media as a news resource. In *Proceedings of the 5th International Symposium on Online Journalism*, 2004.

S. N. Lindstaedt, G. Beham, B. Kump, and T. Ley. Getting to know your user - unobtrusive user model maintenance within work-integrated learning environments. In *Proceedings of the*

4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines, 2009.

S. N. Lindstaedt and J. Farmer. Kooperatives Lernen in Organisationen. In *CSCL-Kompendium. Lehr- und Handbuch zum Computerunterstuetzten Kooperativen Lernen*. Oldenbourg, 2004.

S. N. Lindstaedt, T. Ley, and H. Mayer. Integrating working and learning in APOSDLE. In *Proceedings of the 11th Business Meeting of the Forum Neue Medien 2005*, 2005.

S. N. Lindstaedt, P. Scheir, R. Lokaiczyk, B. Kump, G. Beham, and V. Pammer. Knowledge services for work-integrated learning. In *Proceedings of the 3rd European Conference on Technology Enhanced Learning Times of Convergence Technologies Across Learning Contexts*, 2008.

S. N. Lindstaedt and A. Ulbrich. Integration von Arbeiten und Lernen - Kompetenzentwicklung in Arbeitsprozessen. In *Semantic Web*. Springer, 2006.

R. Maier and A. Schmidt. Characterizing knowledge maturing: A conceptual process model for integrating e-learning and knowledge management. In *4th Conference Professional Knowledge Management - Experiences and Visions (WM 07)*, 2007.

A. Mazarakis, C. Kunzmann, A. Schmidt, and S. Braun. Culture awareness for supporting knowledge maturing in organizations. In *Motivation und kulturelle Barrieren bei der Wissensteilung im Enterprise 2.0, Workshop auf der Mensch & Computer 2011*, 2011.

M. Memmel and A. Dengel. Sharing contextualized attention metadata to support personalized information retrieval. In *Int'l ACM/IEEE Workshop on Contextualized Attention Metadata (CEUR'07): Personalized Access to Digital Resources*, 2007.

M. J. Muller and A. Druin. Participatory design: The third space in HCI. In *The Human-Computer Interaction Handbook*. L. Erlbaum Associates Inc., 2003.

J. Najjar, E. Duval, and M. Wolpers. Attention metadata: collection and management. In *Proceedings WWW2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, 2006.

T. Nelkner, J. Magenheim, and W. Reinhardt. PLME as a cognitive tool for knowledge achievement and informal learning. In *Education and Technology for a Better World. Springer*, 2009.

T. Nelkner, B. Hu, A. Marting, S. Brander, S. Baun, U. Riss, G. Attwell, K. Hinkelmann, and M. Berrio de Diego. D2.3 / d3.3 design and delivery of prototype version v2 of PLME / OLME. Deliverable, 2011.

F. Neven and E. Duval. Reusable learning objects: a survey of LOM-based repositories. In Proceedings of the tenth ACM International Conference on Multimedia, 2002.

I. Nonaka and V. Peltokorpi. Objectivity and subjectivity in knowledge management: A review of 20 top articles. *Knowledge and Process Management*, 13:73-82, 2006.

X. Ochoa and E. Duval. Use of contextualized attention metadata for ranking and recommending learning objects. In *Proceedings of the 1st International Workshop on Contextualized Attention Metadata: Collecting, Managing and Exploiting of Rich Usage Information*, 2006.

A. Rath, M. Kroell, S. N. Lindstaedt, N. Weber, M. Granitzer, and O. Dietzel. Context-aware knowledge services. In *Proceedings of Computer Human Interaction (CHI 2008), Workshop on Personal Information Management: PIM 2008*, 2008.

A. Ravenscroft. Social software, web 2.0 and learning: Status and implications of an evolving paradigm. *Journal of Computer Assisted Learning*, 21: 1-5, 2009.

A. Ravenscroft and T. Boyle. Deep learning design for technology enhanced learning. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2010*, 2010.

A. Ravenscroft, A. Schmidt, and J. Cook. Designing complex systems for informal learning and knowledge maturing in the web 2.0 workplace. In *Proceedings of International Conference on Educational Media (Ed-Media) 2010*, 2010.

J. Rehaeuser and H. Krcmar. *Managementforschung*, chapter Wissensmanagement im Unternehmen, pages 1-40. De Gruyter, 1996.

U. V. Riss, H. F. Witschel, R. Brun, and B. Thoenssen. What is organizational knowledge maturing and how can it be assessed? *Journal of Applied Sciences*, pages 28-38, 2009.

C. Roda and J. Thomas. Attention aware systems: Theories, applications, and research agenda. Computers in Human Behavior, 22:557-587, 2006.

A. Schmidt. Knowledge maturing and the continuity of context as a unifying concept for knowledge management and e-learning. In *Proceedings of I-KNOW 05, Special Track on Integrating Working and Learning (IWL)*, 2005a.

A. Schmidt. Bridging the gap between knowledge management and e-learning with context-aware corporate learning. In *Proceedings of the Third Biennial Conference on Professional Knowledge Management*, 2005b.

A. Schmidt. Potentials and challenges of context awareness for learning solutions. In *LWA 2005: Lernen-Wissensentdeckung-Adaptivitaet, 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems (ABIS 2005)*, 2005c.

A. Schmidt. Microlearning and the knowledge maturing process: Towards conceptual foundations for work-integrated microlearning support. In *Micromedia and Corporate Learning. Proceedings of the 3rd International Microlearning 2007*, 2007.

A. Schmidt. Mature: Den Wissensreifungsprozess in Unternehmen verbessern. In *Verfuegbarkeit von Informationen - 30. Online-Tagung der DGI / 60. Jahrestagung der DGI*, 2008.

A. Schmidt, K. Hinkelmann, T. Ley, S. N. Lindstaedt, R. Maier, and U. Riss. Conceptual foundations for a service-oriented knowledge and learning architecture: Supporting content, process and ontology maturing. In *Networked Knowledge - Networked Media: Integrating Knowledge Management, New Media Technologies and Semantic Systems*. Springer, 2009.

K. Schoefegger, P. Seitlinger, and T. Ley. Temporal patterns in collaborative tagging: Analyzing maturing of semantic knowledge structures. In *It's About Time: Exploring temporality in Group Learning. Alpine Rendez-Vous 2009*, 2009a.

K. Schoefegger, N.Weber, S. N. Lindstaedt, and T. Ley. Knowledge maturing services: Supporting knowledge maturing in organisational environments. In *Proceedings of the 3rd International Conference on Knowledge Science, Engineering and Management*, 2009b.

K. Schoefegger, P. Seitlinger, and T. Ley. Towards a user model for personalized recommendations in work-integrated learning: A report on an experimental study with a collaborative tagging system. In *Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)*, 2010.

C. Scholz. *Personalmanagement: Informationsorientierte und Verhaltenstheoretische Grundlagen*. Vahlen, 2000.

U. Schultze. A confessional account of an ethnography about knowledge work. MIS Quarterly, 24:3-41, 2000.

M. Stefaner, E. D. Vecchia, M. Condotta, M. Wolpers, M. Specht, S. Apelt, and E. Duval. MACE - enriching architectural learning objects for experience multiplication. In *EC-TEL 2007 - Second European Conference on Technology Enhanced Learning*, 2007.

J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.

E. Tomadaki, P. J. Scott, and K. Quick. Attention metadata visualizations: Plotting attendance and reuse. In *CAMA - CEUR Workshop Proceedings*, 2007.

N. Weber, K. Schoefegger, J. Bimrose, T. Ley, S. N. Lindstaedt, A. Brown, and S. A. Barnes. Knowledge maturing in the semantic mediawiki: A design study in career guidance. In *Learning in the Synergy of Multiple Disciplines*. Springer, 2009.

N. Weber, T. Nelkner, K. Schoefegger, and S. N. Lindstaedt. Simple – a social interactive mashup ple. In *Proceedings of the Third International Workshop on Mashup Personal Learning Environments (MUPPLE09), in conjunction with the 5th European Conference on Technology Enhanced Learning (EC-TEL2010)*, 2010.

D. M. Wegner. Transactive memory: A contemporary analysis of the group mind. *Theories of Group Behavior*, pages 185-208, 1987.

D. A. Wiley. Getting axiomatic about learning objects. Online 2012-04-05, 2000. URL http://reusability.org/axiomatic.pdf.

H. F. Witschel, B. Hu, U. V. Riss, B. Thoenssen, R. Brun, A. Martin, and K. Hinkelmann. A collaborative approach to maturing process-related knowledge. In *BPM'10: Proceedings of the 8th International Conference on Business Process Management*, 2010.

E. N. Wolff. The growth of information workers in the U.S. economy. *Communications of the ACM*, 48:37-42, 2005.

M. Wolpers, G. Martin, J. Najjar, and E. Duval. Attention metadata in knowledge and learning management. In *Proceedings of I-KNOW '06, 6th International Conference on Knowledge Management*, 2006.

M. Wolpers, J. Najjar, K. Verbert, and E. Duval. Tracking actual usage: the attention metadata approach. *Journal of Educational Technology & Society*, 10:106-121, 2007.

Appendix

Complete List of Knowledge Maturing Indicators

ID	Level 1	Level 2	Level 3	Indicator
I.	Artefacts			
I.1	Artefacts	Artefact characteristics		
I.1.1	Artefacts	Artefact characteristics		
I.1.1.1	Artefacts	Artefact characteristics	Artefact quality characteristics	An artefact has changed its degree (score) of readability
I.1.1.2	Artefacts	Artefact characteristics	Artefact quality characteristics	An artefact has changed its degree (score) of structuredness
I.1.1.3	Artefacts	Artefact characteristics	Artefact quality characteristics	An artefact has changed its degree (score) of formality
I.1.2	Artefacts	Artefact characteristics	Artefact metadata characteristics	An artefact's meta-data has changed its quality characteristics
I.2	Artefacts	Creation context and editing		
I.2.1	Artefacts	Creation context and editing	creator	
I.2.1.1	Artefacts	Creation context and editing	creator	An artefact has been changed after an individual had learned something
I.2.1.2	Artefacts	Creation context and editing	creator	An artefact has been edited by a highly reputable individual
I.2.1.3	Artefacts	Creation context and editing	creator	An artefact has been created/edited/co-developed by a diverse group
I.2.2	Artefacts	Creation context and editing	purpose	
I.2.2.1	Artefacts	Creation context and editing	purpose	An artefact has been changed as the result of a process
I.2.2.2	Artefacts	Creation context and editing	purpose	An artefact was prepared for a meeting
I.2.2.3	Artefacts	Creation context and editing	purpose	An artefact describing a process has been changed
I.2.3	Artefacts	Creation context and editing	creation process	
I.2.3.1	Artefacts	Creation context and editing	creation process	An artefact was created/refined in a meeting
I.2.3.2	Artefacts	Creation context and editing	creation process	An artefact was created by integrating parts of other artefacts
I.2.3.3	Artefacts	Creation context and editing	creation process	An artefact has been the subject of many discussions
I.2.3.4	Artefacts	Creation context and editing	creation process	An artefact has not been changed for a long period after intensive editing
I.2.3.5	Artefacts	Creation context and editing	creation process	An artefact is edited after a guidance activity
I.2.3.6	Artefacts	Creation context and editing	creation process	An artefact is edited intensively within a short period of time
I.2.3.7	Artefacts	Creation context and editing	creation process	An artefact has been changed to a lesser extent than previous version(s)
I.2.3.8	Artefacts	Creation context and editing	creation process	An artefact was changed in type
I.3	Artefacts	Usage		
I.3.1	Artefacts	Usage		An artefact has achieved a high degree of awareness among others
I.3.2	Artefacts	Usage		An artefact is used widely
I.3.3	Artefacts	Usage		An artefact was selected from a range of artefacts
I.3.4	Artefacts	Usage		An artefact became part of a collection of similar artefacts
I.3.5	Artefacts	Usage		An artefact was made accessible to a different group of individuals
I.3.6	Artefacts	Usage		An artefact is referred to by another artefact
I.3.7	Artefacts	Usage		An artefact was presented to an influential group of individuals
I.3.8	Artefacts	Usage		An artefact has been accessed by a different group of individuals
I.3.9	Artefacts	Usage		An artefact has been used by an individual
I.3.10	Artefacts	Usage		An artefact was changed
I.4	Artefacts	Rating & legitimation		
I.4.1	Artefacts	Rating & legitimation		An artefact has been accepted into a restricted domain
I.4.2	Artefacts	Rating & legitimation		An artefact has been recommended or approved by management
I.4.3	Artefacts	Rating & legitimation		An artefact has become part of a guideline or has become standard
I.4.4	Artefacts	Rating & legitimation		An artefact has been rated high
I.4.5	Artefacts	Rating & legitimation		An artefact has been certified according to an external standard
I.4.6	Artefacts	Rating & legitimation		An artefact has been assessed by an individual
II	Individual capabilities			
II.1	Individual capabilities	Individual activities		
II.1.1	Individual capabilities	Individual activities		An individual has acquired a qualification or attended a training course
II.1.2	Individual capabilities	Individual activities		An individual has contributed to a project
II.1.3	Individual capabilities	Individual activities		An individual has contributed to a discussion
II.1.4	Individual capabilities	Individual activities		An individual is approached by others for help and advice
II.1.5	Individual capabilities	Individual activities		An individual has significant professional experience
II.1.6	Individual capabilities	Individual activities		An individual is an author of many documents
II.2	Individual capabilities	Individual - organization		
II.2.1	Individual capabilities	Individual - organization		An individual changed its role or responsibility
II.2.2	Individual capabilities	Individual - organization		An individual has been a member of the organisation for a significant period of time
II.2.3	Individual capabilities	Individual - organization		An individual has been involved in a process a number of times
II.2.4	Individual capabilities	Individual - organization		An individual has been involved in a process for a significant period of time
II.2.5	Individual capabilities	Individual - organization		An individual has been the owner of a process for a significant period of time
II.3	Individual capabilities	Individual - group		
II.3.1	Individual capabilities	Individual - group		An individual has a central role within a social network
II.3.2	Individual capabilities	Individual - group		An individual changed its degree of cross-topic participation
II.4	Individual capabilities	Rating, assessment		
II.4.1	Individual capabilities	Rating, assessment		An individual has been rated with respect to expertise
III	Topic			
III.1	Topic	Activities		
III.1.1	Topic	Activities		Topic has been searched for
III.1.2	Topic	Activities		Topic has been associated with an artefact
III.1.3	Topic	Activities		Topic has been associated with an individual
III.1.4	Topic	Activities		Topic has been described/documented (or the documentation has improved) in an artefact
IV	Sociofacts			
IV.1	Sociofacts	Process/task (knowledge)		
IV.1.2	Sociofacts	Process/task (knowledge)		A process has been successfully undertaken a number of times
IV.1.3	Sociofacts	Process/task (knowledge)		A process was certified or standardised according to external standards
IV.1.4	Sociofacts	Process/task (knowledge)		A process was internally agreed or standardised
IV.1.5	Sociofacts	Process/task (knowledge)		A process was changed by adding or deleting steps
IV.1.6	Sociofacts	Process/task (knowledge)		A process was documented
IV.1.7	Sociofacts	Process/task (knowledge)		A process was changed according to the number of cycles (loops)
IV.1.8	Sociofacts	Process/task (knowledge)		A process was changed according to the number of decisions
IV.1.9	Sociofacts	Process/task (knowledge)		A process was changed according to the number of participants
IV.2	Sociofacts	Quality of social network		
IV.2.1	Sociofacts	Quality of social network		An individual changed its degree of networkedness
IV.2.2	Sociofacts	Quality of social network		An individual changed its degree of participation
IV.2.4	Sociofacts	Quality of social network		An individual changed its intensity of social action
IV.2.5	Sociofacts	Quality of social network		A group of individuals changed their degree of external involvement
IV.2.6	Sociofacts	Quality of social network		A group of individuals changed their degree of heterogeneity
IV.3	Sociofacts	Agreement		
IV.3.1	Sociofacts	Agreement		A group of individuals created a consensus artefact

IV.4	Sociofacts	Collective capability	
IV.4.1	Sociofacts	Collective capability	A group of individuals has established a reflective practice
IV.4.2	Sociofacts	Collective capability	A group of individuals changed their (systematic) approach to organizational development
IV.4.3	Sociofacts	Collective capability	A group of individuals meets certain quality criteria for collaboration
V	Impact/performance		
V.1	Impact/performance	Performance	
V.1.1	Impact/performance	Performance	The performance of a process has improved
V.1.2	Impact/performance	Performance	The performance of a group of individuals has improved
V.1.3	Impact/performance	Performance	A process was improved with respect to time, cost or quality
V.2	Impact/performance	Quality	
V.2.1	Impact/performance	Quality	The output of a process (product/service) has improved with respect to quality
V.3	Impact/performance	Impact	
V.3.1	Impact/performance	Impact	The customer satisfaction has improved

Figure 38: Complete List of Knowledge Maturing Indicators

User Events from Transition Indicators

Digital Resource	
exportCollectionItem	A collection item got exported to a PDF.
addPrivateCollectionItem	An item got added to a private collection.
addSharedCollectionItem	An item got added to a shared collection.
rateEntity	A resource got rated.
appearsInSearchResult	A resource appeared in a search result.
startDiscussion	A discussion about a resource got started.
viewEntity	A resource got viewed / opened.
addPrivateTag	A private tag got added to a resource.
addSharedTag	A shared tag got added to a resource.
removePrivateTag	A private tag got removed from a resource.
removeSharedTag	A shared tag got removed from a resource.
removePrivateCollectionItem	An item got removed from a private collection.
removeSharedCollectionItem	An item got removed from a shared collection.
renamePrivateCollectionItem	An item of a private collection got renamed.
renameSharedCollectionItem	An item of a shared collection got renamed.
Collection	
shareCollection	A private collection was made public.
subscribeCollection	A collection got subscribed.
unSubscribeCollection	A collection got unsubscribed.
createPrivateCollection	A private collection got created.
createSharedCollection	A shared collection got created.
removePrivateCollection	A private collection got removed.
removeSharedCollection	A shared collection got removed.
renamePrivateCollection	A private collection got renamed.

renameSharedCollection	A shared collection got renamed.
structurePrivateCollection	A private collection got structured.
structureSharedCollection	A shared collection got structured.
Discussion	
renameDiscussion	A discussion got renamed.
addDiscussionComment	A discussion comment got added to a discussion.

Table 7: User Event Types from Transition Indicators

Additional User Events

Digital Resource	
shareCollectionItem-ShareCollection	An item in a private collection got shared by sharing the collection.
structurePrivateCollectionItem-StructurePrivateCollection	An item got structured by structuring its private collection.
structureSharedCollectionItem-StructureSharedCollection	An item got structured by structuring its shared collection.
renameSharedCollectionItem-RenameSharedCollection	An item got renamed by renaming its shared collection.
unSubscribeCollectionItem-UnSubscribeCollection	An item got unsubscribed by unsubscribing its collection.
subscribeCollectionItem-SubscribeCollection	An item got subscribed by subscribing its collection.
viewEntityFromSearchResults	An item got viewed from search results.
addEntityToCollection-FromSearchResults	A resource got added to collection from search results.
rateEntityFromSearchResults	A resource got rated from search results.
Collection	
changeCollection-AddPrivateCollectionItem	A private collection got changed by adding an item.
changeCollection-AddSharedCollectionItem	A shared collection got changed by adding an item.
changeCollection-	A private collection got changed by

RemovePrivateCollectionItem	removing an item.
changeCollection-RemoveSharedCollectionItem	A shared collection got changed by removing an item.
changeCollection-RenamePrivateCollectionItem	A private collection got changed by renaming an item.
changeCollection-RenameSharedCollectionItem	A shared collection got changed by renaming an item.
Discussion	
createDiscussion	A discussion got created.
renameDiscussionTarget-RenameDiscussion	A discussion's target resource got renamed by renaming the discussion.
addDiscussionTargetComment-AddDiscussionComment	A resource for a discussion got a comment by commenting on the discussion.

Table 8: Additional User Event Types