



OIR
34,4

540

Received 16 November 2009
Accepted 12 April 2010

Living Requirements Space

An open access tool for enterprise resource planning systems requirements gathering

Femi Adisa and Petra Schubert

*Centre of Applied Information and Communication Technology,
Copenhagen Business School, Frederiksberg, Denmark*

Frantisek Sudzina

*Institute of Economic Research, Slovak Academy of Science, Bratislava,
Slovak Republic, and*

Björn Johansson

*Department of Informatics, School of Economics and Management,
Lund University, Lund, Sweden*

Abstract

Purpose – This paper aims to discuss a new tool for requirements gathering in the Web 2.0 era. It seeks to investigate the features that this kind of tool should have in order to be as widely applicable and useful as possible. Further, it aims to explore the extent to which business requirements for enterprise resource planning (ERP) systems can be collected and discussed collaboratively in a worldwide community of business process experts.

Design/methodology/approach – The paper is a combination of empirical research, hermeneutics and design research.

Findings – The proposed Living Requirements Space (LRS) platform has the potential of becoming an international forum for collecting and discussing business requirements for ERP systems.

Practical implications – The LRS platform will allow ERP developers, ERP systems implementers, and academics to better understand the evolution of business requirements for ERP systems. It will create a knowledge base of ERP business requirements, that is, a repository that guarantees open and unrestricted access to content. It will thus allow for more international ERP systems and far more comprehensive education on and understanding of business processes and ERP systems.

Originality/value – LRS is an open access tool that allows for the gathering of ERP systems requirements in a vendor- and project-independent approach that is unbiased towards any geographic region.

Keywords Manufacturing resource planning, Resource management, Open systems

Paper type Research paper

Introduction and background

In recent years there has been a discussion as to whether companies should be required to adapt to the inherent (standard) processes of software or whether the software should provide enough flexibility for highly customised processes. In 1998 Davenport described the situation as follows:

Most companies installing enterprise systems will need to adapt or even completely rework their processes to fit the requirements of the system (Davenport, 1998, p. 125).



The term “enterprise system” is a meta-concept for all kinds of software systems that support the activities of a company. Enterprise systems comprise hardware, software, databases, and the necessary networks to link different software systems. Enterprise resource planning (ERP) systems are a specific kind of enterprise system whose emphasis is on the management of a company’s resources (information, financial and human). In this paper we focus specifically on the further development of standard business software, which is generally available through specialised ERP software vendors such as SAP, Microsoft, Oracle, and Sage.

Future generations of ERP systems are expected to be significantly more flexible and agile, and to be more easily adaptable to the needs of an individual company – much like tailor-made software – while retaining the advantages of a standard ERP solution. The biggest challenge to achieving this goal, however, is the proper identification and documentation of requirements at the source: namely, from the users who are dependent on the software to support their specific and often unique tasks.

The process of identifying, gathering, and specifying requirements is the most difficult aspect of information systems development (Alvarez, 2002). Identifying requirements for ERP systems is considered to be especially difficult due to the fact that these systems address all the functions of an organisation (Worley *et al.*, 2005). A basic problem is that the developers of such software struggle to gather, document, and manage the requirements for their products (Wiegiers, 2003).

Jackson (1995) sees the challenge in the requirements collection process between the method for problem structure on the one hand, and the description per se on the other. This is substantiated by Power (2002) who speaks about “requirements as needs” and “requirements as text”. Both authors emphasise the distinction between these two different types of requirements specifications. They describe the challenge of transforming requirements from text into a formal description, which can be easily transformed into program code and software system features. ERP systems operate in the business applications domain, an environment that is complex, uncertain, and volatile; constantly evolving business requirements require that information systems take into account not only current, but also future business requirements.

In this paper we have coined the term “living requirements” to reflect challenges posed by the constantly evolving characteristic of requirements. The concept of living requirements is a reference to the volatile nature of business requirements, to which we propose the “Living Requirements Space” (LRS). The LRS is an online platform that gives access to collaboratively developed online information for the community of stakeholders in the realm of ERP requirements engineering.

Duncan-Howell (2010, p. 324) states that “[o]nline communities are being increasingly used by teachers for professional support, guidance and inspiration”. But we aim to go even further. The LRS is an online platform that will support a community of stakeholders in the ERP domain. Members of this web-based community – requirements engineers, domain experts, business analysts, software users, ERP providers and other stakeholders – will collaborate on the discovery and development of business requirements for future generation ERP systems. The platform aims to support the information flow among the different ERP system stakeholder groups, from users to business analysts through to developers and also researchers, ultimately creating a truly global and authoritative repository on business requirements. This paper describes the concept of the LRS platform and provides a

framework that aims to foster the collaborative collection and development of business requirements for ERP systems.

Basing the platform on open access allows the content to be of benefit not only to ERP vendors and professionals, but also to academics and students in the framework of courses such as ERP systems, requirements engineering, business processes, and modelling methodologies, and to researchers looking into the evolution and development of business requirements.

According to Piedra *et al.* (2009, p. 497:

Open Educational Practices and Resources are a direct response to privatisation of knowledge; they promote their exchange across the world with the aim of increasing human intellectual capacity.

Although many people are freely opening and sharing their diaries, social networks and source codes, we do not experience the same degree of openness in scientific knowledge, due to reasons discussed by Mann *et al.* (2009). There are, however, some journal publishers who allow authors to retain their rights (Seadle, 2005).

According to Chantavaridou (2009, p. 167) the European Union encourages “citizens and institutions to support free and open access to European research”. Storing publications in open repositories is already a condition for project funding in some cases. The research question that is addressed in this paper is thus: to what extent can business requirements for ERP systems be collected and discussed collaboratively in a worldwide community of business process experts?

The remainder of the paper is structured as follows. The next section describes the data and methodology, after which the literature review on requirements and an explanation of the basic terms and definitions used in this paper are provided. The process of collecting requirements is then elaborated upon. The penultimate section describes the proposed solution with its technical components. The last section concludes the findings and gives an outlook for future work in this area.

Data and methodology

In this paper we use a combination of empirical research, hermeneutics and design research. In the first phase (empirical research) we conducted interviews with a large ERP software vendor and learned about the internal process of requests for change. This vendor depends on its network of independent implementation partners to deliver its offerings to user organisations. Consequently the ERP vendor cannot “learn” directly from implementation projects and is totally dependent on the feedback from business analysts in the partner companies. We performed an analysis of pain points (Johansson and Sudzina, 2007) and learned that the indirect sales channel poses a great challenge, and that it is currently difficult to request and receive user requirements in a form that can be easily interpreted by a programmer. Additionally we performed a survey of current tools for requirements engineering, primarily to analyse how requirements are represented, described, stored and displayed, and also to further establish if and how current tools provide support for project-independent requirements gathering in a vendor-independent, distributed and heterogeneous environment.

In the second phase (hermeneutics) we conducted a series of workshops with researchers and representatives of the ERP vendor, and brainstormed new ways of

requirements gathering that could overcome the challenges of the current (inefficient) processes. The end result was an idea of a collaborative web platform based on principles from social software and Web 2.0 technologies that can be employed in the collection and development of business requirements – the LRS platform. The last phase (design research) describes ongoing work that involves the development of the first prototype, based on and guided by the framework and principles presented in this paper, which will later be tested (validated) with the ERP vendor's implementation partners (business analysts) in the field.

Literature review and definition of terms

Requirements

A requirement is a specification of what should be implemented. It is a description of how the system should behave, including information on application domain, constraints on the system's operation, or specification of a system property or attributes (Kotonya and Sommerville, 1998). According to Thayer (1997) a system requirement is a system capability needed by the user to solve a problem or achieve an objective, and/or a system capability that must be met or possessed by a system or system component in order to satisfy a contractual, standard specification or other formally imposed document. Thayer's definition underscores the importance of the user focus when dealing with requirements. Understanding user requirements is crucial to the planning and success of a project and the subsequent acceptance of the new system. According to Hull *et al.* (2005) requirements provide the basis for project planning, risk management, acceptance testing, trade-off, and change control.

A requirement describes what is needed in order to achieve something. Robertson and Robertson (1999) state that a requirement is something that the product must do or a quality that the product must have. Further refining this definition, it can be suggested that requirements can be defined as statements of needs. In this paper we focus on two different kinds of requirements: business and system requirements. We use the term "business requirements" to refer to the capabilities a user requires from a system, and "system requirement" when we refer to a system feature that implements a user requirement or what is needed to generate a software code (software program).

Domain

Hull *et al.* (2005) emphasise the importance of making a clear distinction between the "problem domain" and the "solution domain" from an engineering and management perspective when dealing with requirements. They state that requirements in the problem domain are stakeholder requirements that describe the capabilities that users expect from the new system, while requirements in the solution domain represent system features which engineers implement to solve stakeholder requirements. Power (2002) reaffirms this view on requirements, further adding that the type of requirements people consider in any given situation depends on the context; requirements in the context of system development may be defined as: a function, capability, or property required of a proposed system, and/or the statement of such a function, capability or property.

Requirements engineering (RE) is often described as the process of closing the gap between a specific problem and the solution for that problem. The RE process consists of a set of activities which cover the development and management of the set of

requirements for a computer-based system. Requirements development activities include the elicitation, analysis and negotiation, and validation processes, all running in parallel with the requirements management process which not only documents the requirements but also tracks and control changes to the “requirements document” (Kotonya and Sommerville, 1998).

Requirements management (RM) is described as the requirements engineering activities concerned with finding, organising, documenting and tracking requirements for software systems (Finkelstein and Emmerich, 2000). Cant and McCarthy (2006) posit that the requirements management process should deal with requirements as they evolve through the system lifecycle. Thus the primary focus of RM is maintaining traceability, which is the ability to traverse the lifecycle of a requirement in both a forward and backward direction – that is, from its origin, through its development and specification, to its subsequent deployment and use, and through all periods of ongoing refinement and iteration in any of these phases (Gotel and Finkelstein, 1994). Requirements are specified in a systems requirements specification document or requirements document (see Figure 1). This is a document consisting of a textual description of the requirement in natural language supplemented by diagrams, process diagrams, system models, etc.

Studies have shown that there is a link between the quality of requirements collected and the tools utilised. RE tools are said to provide better support for the RE process than general office and modelling tools, resulting in higher quality requirements documents (Matulevicius, 2004). However according to Finkelstein and Emmerich (2000) a dominant issue with current RM tools is distribution; the authors claim that most of the existing RM tools are localised – referring to the fact that they are meant to be used for gathering requirements on a project by project basis – and are therefore not meant for the elicitation of business requirements that are independent of any one particular project in a project-independent and highly distributed heterogeneous collaborative manner. The ability to collaborate on the development of business requirements is highly desirable, especially because expert knowledge on the business requirements that shape ERP systems is scarce and typically thinly distributed. This is not surprising when we consider the fact that ERP systems are global information systems that are implemented across the globe. Therefore there is a need for tools that can foster widespread collaboration in the identification and

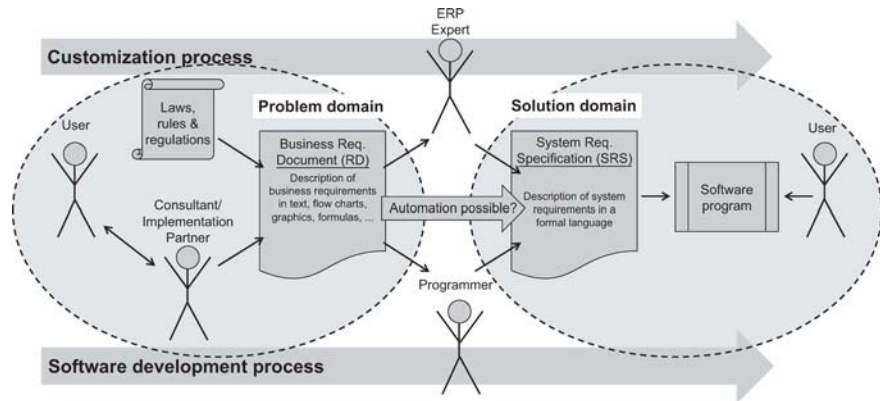


Figure 1.
The process from requirements gathering to programming the software

development of business requirements on a global scale. This is also supported by Damian and Zowghi (2002, p. 9) who show that “distance has a significant impact on the collaboration between geographically distributed functional groups involved in the negotiation of requirements from a diverse customer market”. The concept of mass collaboration is covered in later sections.

Stakeholder requirements, also known as user requirements or business requirements, are requirements written from the point of view of system stakeholders (Sommerville and Sawyer, 1997). They are an initial statement of capability that defines the problem and standard practices within the particular industry in which an organisation operates (Krebs, 2005). They belong to the stages of development associated with the highest levels of system description, and together with the statement of need and usage modelling should be firmly rooted in the problem domain (Hull *et al.*, 2005). Stakeholder requirements are not usually expressed in great detail and should state no more than is necessary to define the problem at a level of abstraction that avoids referencing any particular solution.

Systems requirements are derived by modelling business requirements. They should be aligned with business requirements and focus on possible solutions to help the business get its job done more effectively (Krebs, 2005). Thayer (1997) defines a system requirement as a system capability needed by the user to solve a problem or achieve an objective. Systems requirements are more detailed specifications of requirements that should be expressed as an abstract model of the system (Sommerville and Sawyer, 1997). According to Hull *et al.* (2005) systems engineers should refrain from inappropriate bias for a certain software product and avoid encroaching into the solution domain space of designers by making sure the elements of solution introduced through functional modelling remain at a high level, allowing designers the freedom to design an abstract solution. Systems requirements are used to create system models of the proposed systems that are abstracted from the final solution. They serve as a basis for discussion and understanding what characteristics the system must have, irrespective of the final detailed design (Hull *et al.*, 2005). Figure 1 shows an overview of the requirements process.

Requirements in the problem domain (user perspective)

The problem domain represents the business environment where a system will be used (see Figure 1). It is the starting point of any systems engineering project. It is here that the capabilities required from the new system are established. The end product of the RE process in the problem domain is a requirements document that is a structured set of stakeholder requirements.

Requirements engineering in the problem domain is primarily about eliciting capabilities. In order to be able to do this, it is important to establish what people want to be able to do with the system. This leads to an important question and the first aspect of RE in the problem domain according to Hull *et al.* (2005): “Who should be asked?”

Requirements in the solution domain (programmer perspective)

The solution domain is the domain of engineers (programmers). This is where problems outlined in the problem domain are solved. The refined and structured set of business requirements that are the output of the problem domain provide the input and

serve as the basis for the development efforts in the solution domain. Once the stakeholder requirements are well understood and documented, it is time to think of potential solutions in terms of the characteristics that the new system must have, irrespective of the final detailed design. This process is known as establishing the systems requirements (Hull *et al.*, 2005). However there is said to be a gap of knowledge between users and developers of information systems, mainly due to the fact that the people who identify user needs are not the eventual builders of the system (Johansson and Sudzina, 2007).

An abstract model of the proposed system is developed to provide a basis for discussion and establishing a common understanding of the proposed solution among the development team. The model provides a structure for documenting the systems requirements and may also be used to explain the solution concept to stakeholders. The produced document serves as an aid to reviewing the complete requirements set from a consistency and completeness perspective.

The next phase is to develop architectural designs based on these sets of systems requirements. The design architecture is expressed as a set of interacting components that collectively exhibit the desired properties. It defines what each system component must do and how the system components interact with each other to produce the overall effects specified in the system requirements (Hull *et al.*, 2005).

The concept of living requirements

Requirements can and will inevitably change during the system development lifecycle, and do not necessarily point to a flaw in the RE process. They occur while requirements are being elicited, analysed, and even after the system has gone into service. Such changes may occur as a result of stakeholders developing a better understanding of the application domain or due to external circumstances, e.g. legal, social, etc. (Kotonya and Sommerville, 1998). In the ERP application domain, requirements management is not about keeping such changes under control; RM is about creating an environment where the evolution of business requirements can be recorded, where ERP providers can keep abreast with changing business requirements and hopefully be able to gain an insight into how requirements evolve and develop over time, so that they can be in a position to better predict requirements. The concept of living requirements stems from the volatility and constantly evolving nature of business requirements in the ERP domain. The LRS is our proposed solution to constantly changing characteristics of business requirements. It is a platform where domain experts, business analysts and other ERP stakeholders can collaborate in every stage of the requirements lifecycle, from the identification, to the analysis, and through to the management of business requirements for ERP systems.

In a classic software development cycle, the requirements development phase of RE ends as soon as the documented requirements are passed on to developers. RM then tries to keep changes under control during the rest of the cycle (Kotonya and Sommerville, 1998). ERP providers have no such luxury, as they need to be aware of changes in the business domain: new requirements emerge, and current requirements change in any local market where their software is used, and thus they must continue to improve their offerings in tune with those changes if their products are to continue to offer “best practice” solutions. Hence we can infer that requirement development and

management in the ERP domain are parallel, intertwined, overlapping, continuous, and iterative activities.

It therefore follows that in a volatile business environment which characterises the ERP application domain, an ideal situation would be one in which ERP developers are able to involve the various cross-sections of ERP stakeholders from across multiple regions in the engineering of business requirements for future ERP releases. Kotonya and Sommerville (1998) emphasise the need for requirements elicitation to be a cooperative process involving requirements engineers and system stakeholders and that effective elicitation requires effective cooperation. Such collaboration could lead to a better understanding of how business requirements are evolving in various regions across the globe, and result in the ability to predict future requirements and implement them as system features prior to their next release. However in reality a good working relationship amongst all parties is often difficult to achieve. Several obstacles to properly understanding system requirements have been identified: application domain knowledge not being collected neatly in one place, (in)accessibility of people who understand the problem to be solved, organisational issues and other external factors, and stakeholders not knowing what they want at any given time. Our aim is to create an environment through the LRS platform where such limitations can be overcome.

The requirements template

The end product of the requirements documenting activity is the requirements document. This is a formal document used to communicate the systems requirements to customers, end-users, software developers and managers of the software engineering process (Sommerville and Sawyer, 1997). Different organisations have different names for it, including: Requirements Document (RD) (Kovitz, 1998), “functional specification”, “the requirement definition”, and “the software requirements specification” (IEEE, 1998); there is no universal standard template. Depending on the target audience, a requirements document can be either customer or developer oriented. The RD plays an important role in the requirements specification process. It serves as the basis for software development contracting (Robertson and Robertson, 1999), providing a means for external reviews (Boehm, 2002), for reuse in other projects (Czarnecki, 2002), as well as a standard medium to communicate requirements to customers (Sommerville and Sawyer, 1997), users, managers and developers.

An important characteristic of requirement documentation is that it should not have a definite end. The information about a requirement is typically illustrated with various process and system model diagrams, and can also be supplemented with other detailed documentation files. This multi-document nature of the requirements document requires a template that makes it easy to link all these documents together and maintain traceability, and since an information system needs to change and often does, the requirements document should be a supporting tool in this process.

Mass collaboration and virtual teams

Sound requirements processes emphasise a collaborative approach to product development that involves multiple stakeholders in a partnership throughout the project (Wiegers, 2003). Collaboration is very powerful for solving problems, building consensus, and helping in the decision making process (Straus and Layton, 2002). Tapscott and Williams (2006, p. 2) in *Wikinomics: How Mass Collaboration Changes*

Everything base their concept of mass collaboration on four principles – openness, peering, sharing, and acting globally – thus leading to what they call an economic democracy where stakeholders and even competitors “co-create value in the absence of direct managerial control”. This decentralised model characterised by the four principles is the source of success and uniqueness over the traditional collaboration paradigm, where central control is applied (Braffman and Beckstrom, 2006). The internet through Web 2.0 technologies and services has allowed for a shift in users’ traditional role from being information receivers to becoming knowledge and content generators (Archer, 2009). Fasoldt (2004), Orlowski (2005), and Lipczynska (2005) have raised the question of Wikipedia’s reliability, i.e. to what degree one should trust the information posted there. Korfiatis *et al.* (2006) have proposed an approach for evaluating the reliability of Wikipedia entries that can also be used for LRS.

Mass collaboration relies on free individual agents to come together and cooperate to improve a given operation or solve a problem. However this type of coming together neither requires participants to be physically present in one location, nor for them to participate at the same time. Such collaboration is usually web-based and employs the use of social software and computer-supported collaboration tools, such as wiki technologies (Cadiz *et al.*, 2000), and differs from traditional forms of collaboration in that the collaborative process is mediated by the content being created and not by direct social interaction.

Although the term mass collaboration is relatively new, the concept has been around for some time. According to Berners-Lee and Fischetti (1999) one of the initial design aims of the web was not only to facilitate views of the resources requested, but also to allow editing and annotation of these resources. The Open Source community enjoys the benefits of mass collaboration. Linux owes its success to the collaborative efforts of open source enthusiasts that are scattered around the globe. Wikipedia, a popular online resource, owes its contents to the efforts of about “10 million volunteers who collaborate over the web to create a worldwide encyclopaedia” (Panchal and Fathianathan, 2008, p. 1).

Not all members of the online community actively contribute, and the term “lurkers” has been coined to describe those who do not. Nonnecke *et al.* (2006) have studied lurkers and posters, and found that some of the former may eventually become the latter. Since lurkers are less satisfied with their online community experience and gain fewer benefits from membership, their unhappiness may lead them to become active posters. The higher satisfaction of posters should lead them to keep doing what they are doing, i.e. to actively contribute to the online community.

There is a growing trend of actively engaging users outside their walls, as a new way through which organisations are improving the quality and capabilities of their offerings. In alignment with this trend, our concept of living requirements aims to exploit the openness, peering, sharing and acting with global principles that characterise mass collaboration to collectively create content and build a knowledge base on business requirements for future ERP systems; the LRS platform will provide such an enabling environment.

Relevant Web 2.0 technologies for the proposed LRS platform include wikis, content sharing, folksonomy, mashups, forums, and social networking. Wikis allow for collaborative editing supported by a revision system, which monitors changes and contributions to the edited entries (Dearstyne, 2007; Lee and Lan, 2007). Content

sharing allows sharing of different types of data files with a possibility of setting various access rights (Hart *et al.*, 2007). Folksonomy – social tagging – allows members to collaboratively annotate web content using tags (Soloman and Schrum, 2007). Mashups are web applications generated by combining applications, content, presentations, and services from possibly more than one source; the idea is to provide new applications that provide better support to users (Kulathuramaiyer, 2007). Forums – discussion services – make it possible for members to interact, thus they can share their opinions on different topics (Guzdial and Turns, 2000). Social networks are web applications which focus on creating online user communities; members often have profiles with information on their activities and interests, including pictures, videos, and music. These social networks offer a wide range of possibilities for users to interact (Boyd and Ellison, 2007; Boyd and Heer, 2006; Farnham *et al.*, 2004).

The LRS community

The LRS community will be made up of stakeholders in the ERP system domain, specifically:

- *ERP experts* – domain experts, business analysts, consultants, implementation professionals, and partners – i.e. people with expert knowledge on ERP implementation.
- *ERP users* – members of organisations that have implemented, are implementing, or are pondering implementation of ERP systems – i.e. the source and owners of business requirements.
- *ERP providers*, i.e. the manufacturers and developers of standard ERP systems.
- Academics and researchers and students of ERP systems in particular, and business software in general.

Members of the community will be engaged in active collaboration in the identification, analysis, negotiation, validation, conflict resolution and prioritisation of business requirements for future ERP systems. They will also be reporting business trends and academic breakthroughs that affect business requirements. The LRS platform will support the requirements engineering activities of its user community by facilitating the free flow of information amongst the identified groups through the use of Web 2.0 technologies identified in the previous section. The homogeneity of stakeholders does not allow the application of all the findings of Ruth and Houghton (2009, p. 149) such as the importance of members (students in their case) being online at the same time, but the LRS platform is expected to “foster a deeper style of learning that is more collaborative, reflecting and cooperative than traditional ‘competitive’ assessment”.

Requirements tools survey

Prior to developing a new tool for the distributed and collaborative gathering of requirements, we performed an analysis of existing tools for RE. The base for our investigation was the INCOSE requirements management tools survey (INCOSE, 2009). We selected a sample of 27 companies from the INCOSE list and sent them a request (by e-mail or through their web site) to fill in our web-based questionnaire. We received 14 responses, giving a response rate of 51.9 per cent. The questionnaire contained 16 questions which addressed issues such as general use of RE tools with

focus on mass collaboration, interconnectivity of requirements, and level of detail of requirements. The collected information about RE tools is presented in Table I.

One conclusion that can be drawn from the investigation is that software management tools work under the assumption that new, as well as old, requirements must be connected to a project. This is a problem in the ERP system context since the development of an ERP system is not a single project. An implementation of an ERP system can be seen as a project – in fact it is probably the biggest and most problematic IT project that most organisations can conduct; however since the development of ERP systems represents the development of a standard software package, a specific requirement cannot be connected to a single instance in the form of a project. The toolmakers limit themselves to selling the licences to companies, that is, they do not build up an open access database on ERP requirements. Another limitation of the existing software management tools is that they do not support a distributed, collaborative collection and analysis of requirements, and thus they are not able to support different stakeholders in a distributed environment.

A crucial functionality of ERP systems requirements management software tools is to support efficient communication of, as well as fast responses to, changes in requirements. In that way it can be claimed that the tool needs to support agile development methods by delivering the software in small and frequent increments. However from our investigation of software requirements management tools currently on the market (a study that was conducted from the perspective of a distributed requirements management view), the main conclusion is that existing tools do not support requirements management in the ERP system context. The LRS platform is meant to fill this gap and support requirements elicitation from heterogeneous and disparate sources made up of independent stakeholders with no affiliation to any one particular project.

Our aim is not to rival current RE tools, for while they provide the facility to gather requirements towards a particular project implementation, our vision is for a tool that will complement the efforts of ERP developers by developing future ERP systems that are more in tune with both current and future realities.

The proposed solution: the LRS platform

As a platform for the collaborative collection of business requirements the web offers a strategic positioning advantage that is well suited to and essential for distributed and heterogeneous collaboration. This mode of delivery also utilises Web 2.0 technologies to provide an RM tool that requires no installation on the part of participants, and has easy accessibility worldwide. The LRS RM tool integrates key features that are essential for the proper management of requirements, from identification, storage, retrieval, and projection, to implementation. Its architecture is based on participation, and the aim is to harness collective intelligence by providing an avenue for practitioners and stakeholders in the ERP domain with shared or similar interests to generate knowledge about business requirements for ERP systems in a distributed and collaborative manner, thereby creating a knowledge base that will provide a one-stop-shop for business requirements.

We employ elements of Web 2.0 technologies, particularly social networking and wiki technologies, to create an enabling environment that will foster the:

RE tool	SoftREQ (no particular version)	IBM Rational DOORS 9.2	Cognition Cockpit Version 5.4	Rational Requirements Composer	CaseComplete 2009 R2	RequirementOne ReqMan 2.0	Kovair Software	PACE	www.gatherspace.com	Cradle version 6.0	Polarion ALM	In-Step Version 4.4	PTESY Project and Test Engineering System	MKS Integrity 2009
Is the RE tool linked to only one particular software (P) or is it generic (G)?		G	G	G	G	G	G	G	P	G	G	G	P	G
Does the RE tool allow more than one user to work at a particular time? If so, does the RE tool support web-based collaboration?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
If so, does the RE tool provide any negotiation structure among collaborators?	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	Y	Y	NA	Y
Does the RE tool allow requirements to be registered independent of any particular project?	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
What type of repository of requirements does the RE tool use (centralised – C, distributed – D, federated – F)?	N	N	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y
	C	C	D	C	F	C	C	C	C	C	C	C	C	C

(continued)

LRS: an open access tool

Table I. RE tool properties

Table I.

RE tool	SoftREQ (no particular version)	IBM Rational DOORS 9.2	Cognition Cockpit Version 5.4	Rational Requirements Composer	CaseComplete 2009 R2	RequirementOne ReqMan 2.0	Kovair Software	PACE	www.gatherspace.com	Cradle version 6.0	Polarion ALM	In-Step Version 4.4	PTESY Project and Test Engineering System	MKS Integrity 2009
Does the RE tool provide linkability between a requirement and its supplementary attributes?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Does the RE tool maintain a history of changes made to each requirement and its attributes, including the source of such changes?	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Does the RE tool support the baselining of requirements?	N	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
Does the RE tool support the use of unstructured text (language) in the description of requirements?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
What media does the RE tool support in describing requirements	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
- Text	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
- Video	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
- Audio	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
- Diagrams	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
- Process models	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N

(continued)

RE tool	SofiREQ (no particular version)	IBM Rational DOORS 9.2	Cognition Cockpit Version 5.4	Rational Requirements Composer	CaseComplete 2009 R2	RequirementOne ReqMan 2.0	Kovair Software	PACE	www.gatherspace.com	Cradle version 6.0	Polarion ALM	In-Step Version 4.4	Project and Test Engineering System	MKS Integrity 2009
- Case diagrams	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	Y	N	N
- Use scenarios?	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y
What level(s) of detail does the RE tool include														
- Business scenario	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
- Process	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
- Sub-process	N	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
- Process step	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y
- Task	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
- Details of task?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
What level(s) of detail are required by the RE tool														
- Business scenario	Y	N	N	N	N	N	N	N	Y	Y	N	N	Y	N
- Process	Y	N	N	N	N	N	N	N	Y	Y	N	N	Y	N
- Sub-process	N	N	N	N	N	N	N	N	Y	Y	N	N	Y	N
- Process step	Y	N	N	N	N	N	N	N	Y	Y	N	N	N	N
- Task	N	N	N	N	N	Y	N	N	Y	Y	N	N	N	N
- Details of task?	Y	N	N	N	N	Y	N	N	Y	Y	N	N	N	N
How many industry-specific vocabularies does the RE tool support (One - 1, more - M)?	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Does the RE tool generate documentation?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

LRS: an open access tool

Table I.

- heterogeneous and geographically distributed collaborative efforts of the community;
- requirements management activities including the identification, storage, retrieval, projection and update of requirements as well as maintaining traceability and linkability between requirements and the attributes;
- requirements change management activities allowing for multiple, distributed authors to collaboratively append and update information about a requirement; and
- proper management of the knowledge base.

Given different types of stakeholders and these various goals, we refrain from setting up evaluation criteria for now.

The software architecture

The LRS RM tool architecture is based on Semantic Web 2.0 technologies and is illustrated in Figure 2. It leverages the knowledge management ability of wiki technologies which natively supports collaboration and implements several intrinsic wiki features to support the requirements development and management processes. Such key features include:

- *History and versioning* – providing the ability to record the state of a requirement along major review processes.
- *Traceability* – forwards and backwards, allowing the tracing of a requirement from source to implementation and vice versa.

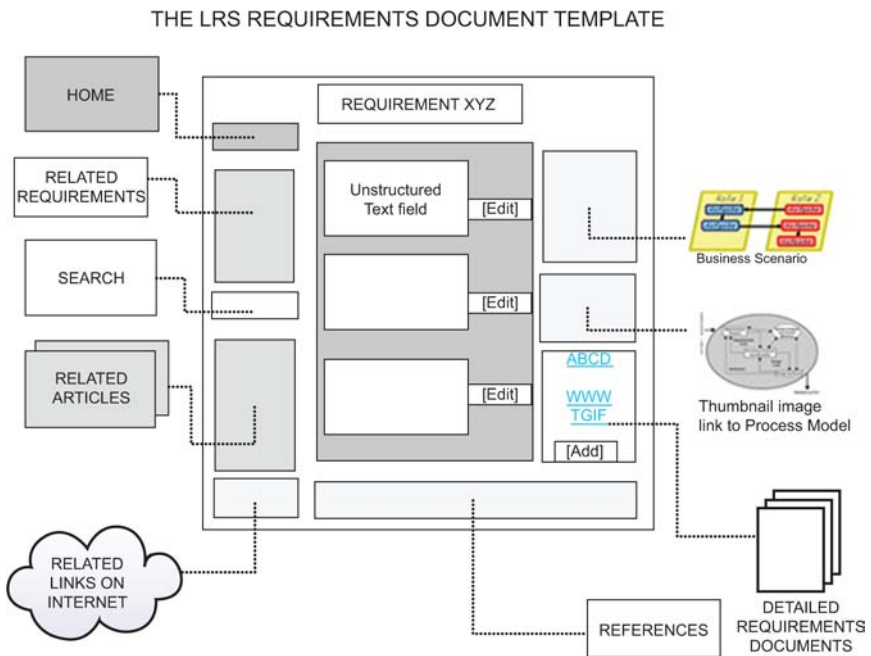


Figure 2.
The requirement template

- *Updatability* – as information about requirements will be added in increments, updatability is a basic feature and wikis facilitate easy updating.
- *Linkability* – providing the ability to link a requirement object with supplementary and complementary attributes.
- *Flexible electronic interface*, which supports the embedding of all media types, which is an essential functionality for RM in general, and is crucial for distributed and heterogeneous collaboration.


The core is an enterprise web application, backed by an object oriented database, knowledge spaces for users/groups, and a freely configurable interface. The wiki software sits on top of the database and provides the interface for interacting with the database objects through a web browser. Wiki pages are set up to represent the various objects in the database; the wiki software not only provides an electronic interface to collect requirements, it also allows for the easy creation of interlinked web pages using a simplified mark-up language or a WYSIWYG text editor.

Links to images, use of case diagrams, use of scenarios and process models are displayed on the base RD template as thumbnail images which, on clicking, open the full image in a new browser window. Links are also placed in the document template to point to current research, references and literature that support the requirement. Sections and subsections can be rolled up to show only headers and to hide details or drill down to display greater levels of detail in the requirement hierarchy. The LRS RD supports the multiple roles that a requirements document must play in the requirements lifecycle. The LRS RD facilitates the registering of new requirements, searching, structuring and viewing of registered requirements and associated attributes and appendices, communication of requirements, easy editing and updating of requirements attributes, and reviewing of requirements, as well as easy management of the complex documents that augment a requirement. Requirements change during their lifecycle, and in the situation where we are trying to promote collaboration, it is crucial that this volatility of requirements is supported. The electronic RD (schematically illustrated in Figure 3) makes it easy to have a holistic overview of requirements and their supplementary attributes, while not clogging up the screen.


The database


The structure of the requirements object as an entity, made up of several entities of different structures with direct links between them, makes it better suited to object-oriented databases, as opposed to relational databases that perform best when records are of the same structure with minimal links (Kotonya and Sommerville, 1998). Object-oriented databases are better suited for RM because they intrinsically handle complex data types such as video, audio, graphs and photos, all of which can be attributes of the requirements object, and which traditional relational database management systems were not natively designed to handle. Furthermore object-oriented database management systems excel when a huge amount of diverse and related data about one item is to be stored, as is the case with requirements. The number of actual requirements that will be stored in an RM database may number in the few thousands, but the number of links to its corresponding attributes such as documents, images, text files, model diagrams and other related requirements will be

REQUIREMENT XYZ

This page describes a typical [requirement](#) of a web ordering business scenario. It describes the processes which are involved in the transactions 

Dependents Requirements LPP Receive orders PKL Requirements ABZ Requirements AMM	Depends On Requirements ABC Requirements HJJ4 Requirements AB×3 Requirements UN4	Sources Adams, W.B Williams, B.M Andersen, N.B. Johnson, F.B
---	---	---

Statement [edit]
This requirement specifies that the software be able to register new customers as well as the ability to retrieve and update existing customers. 

Rationale [edit]
This requirement specifies that the software be able to register new customers as well as the ability to retrieve and update existing customers. 

Constraints [edit]
This requirement specifies that the software be able to register new customers as well as the ability to retrieve and update existing customers.

Performance Requirements [edit]
This requirement specifies that the software be able to register new customers as well as the ability to retrieve and update existing customers.

References [edit]
[Ahhn\(1990\).. Oxford press](#) [Yourdan\(1965\).....](#)
[Hnhnk \(2002\)...](#) [Imogen\(2006\)...](#)

Figure 3.
A sample LRS
requirements document

exponentially greater. Object-oriented database management systems implicitly provide support for the versioning of content and metadata, and linkability between objects.

A requirements object is a complex entity and to comprehensively represent it, we needed to model it. We have mapped a requirements hierarchy, representative of how requirements are derived from the overall business scenario right down to the business task which a requirement helps to fulfil (see Figure 2), thus allowing us to adopt a process-oriented approach to requirements engineering. We have combined different existing methods for the abstract representation of requirements, e.g. the ProZoom and eXperience methods (Schubert and Wölfe, 2007) and Event-Driven Process Chains (Scheer, 2000). The entities in the resulting requirements hierarchy are connected through hyperlinks. This useful feature and design allows us to traverse the

requirements hierarchy and provides traceability, whereby a requirement can be traced upwards to its highest level of abstraction, and where the purpose of a requirement to support upper level business goals is realised. A business use scenario can also be drilled down to its most granular level to view the set of requirements that supports its processes. Figure 3 shows a schematic overview of the RD.

A concrete, real life example of an instance of the requirement hierarchy, top-down would be:

- (1) *business scenario* (e.g. procurement process in the wholesale industry, business partners: manufacturer and customer);
- (2) *process* (e.g. order process of a wholesaler, customer view);
- (3) *sub-process* (e.g. call order process, collection of necessary data);
- (4) *process step* (e.g. generate call order in ERP system);
- (5) *task* (e.g. key in order quantity); and
- (6) *details of task* (e.g. a formula for calculating a special discount).

The requirements hierarchy is illustrated in Figure 4. A business scenario is on the top of the hierarchy because, as Sabol and Delina (2004) suggest, scenarios provide individuals with the opportunity to ask “what if” questions. This then serves as a basis for the analysis process, including checks for incomplete, inconsistent and missing requirements, and allows for further negotiation of requirements, discussion, prioritisation and agreement.

We have also developed a model of our view of the requirement object that is representative of the heterogeneous format of the attributes that make up a requirement, which is central to our database design. We base our design on an original requirements object model by Kotonya and Sommerville (1998) and we have extended it to reflect our requirements hierarchy (see Figures 5-8).

The heterogeneous formats of the constituent attributes of the various object classes are made up of a mix of text, graphics, and models; they also make extensive use of lists. This is needed to support the multi-format nature of materials used to document a requirement and the infinite number of potential supplementary materials.

Requirements development in the LRS

In the LRS, members of the four identified groups contribute to the continuous and iterative engineering of business requirements, with members' contributions based on their area of expertise, and utilising the LRS as a viable way to disseminate and gain valuable knowledge and trade war stories. The primary sources of requirements are the ERP experts and user communities, who will use the platform to: identify and register new business requirements as they arise in their local environment; submit requirements not currently being met by existing systems; and append and update complementary data to registered requirements. The ERP vendors are mainly consumers of the contents of the space. Researchers and academics are both contributors (reporting research) and consumers (taking advantage of the rich information content). Members use folksonomy to create and manage tags or keywords to label and annotate web content.

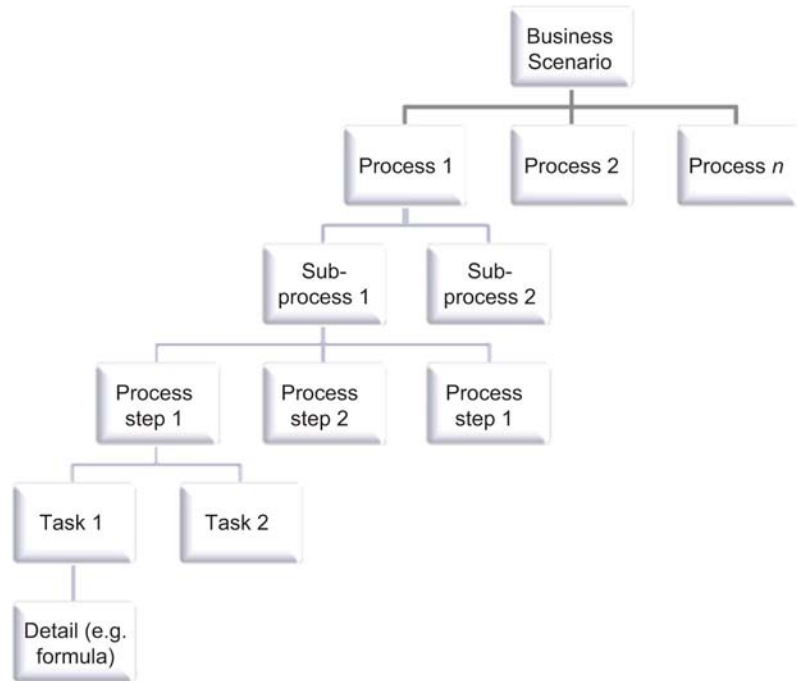


Figure 4.
The decomposition of a requirement (hierarchy)

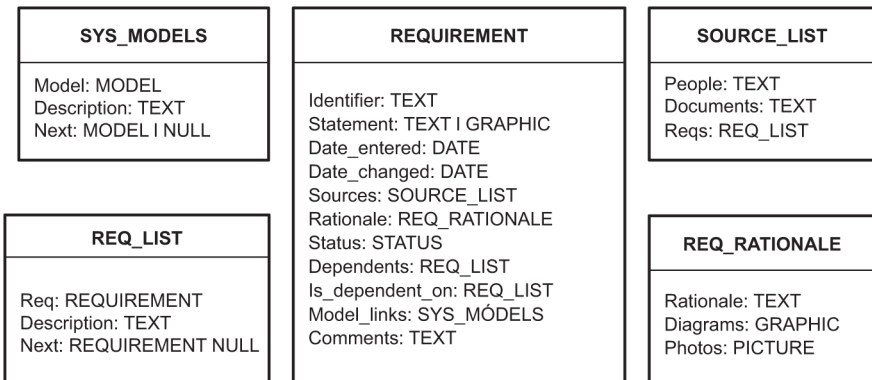


Figure 5.
The requirements object

Sources: Based on Kotonya and Sommerville (1998)

Requirements analysis in the LRS

Collaborative negotiations in the analysis of requirements between the ERP providers and expert community will take the form of web-based online discussions. We try to avoid branding it either as a community of practice or of interest. Jones and Preece (2006) describe advantages and disadvantages of both, but because of the peculiarity of our approach, the LRS community falls between the two extreme positions. Forums and discussion threads will be employed. A discussion thread may be opened on a

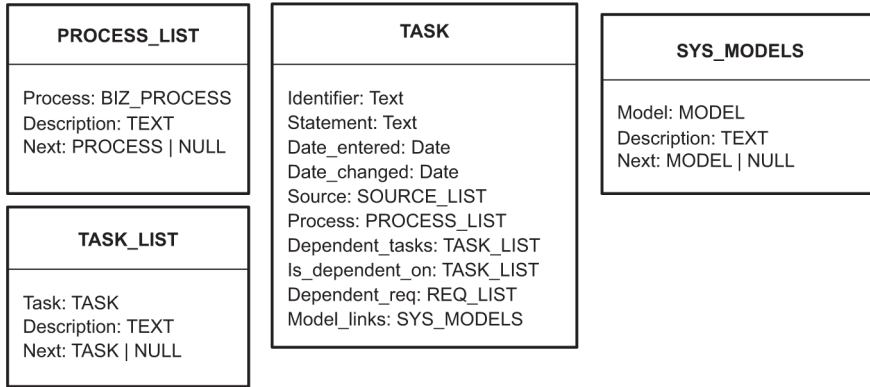


Figure 6.
The LRS task object

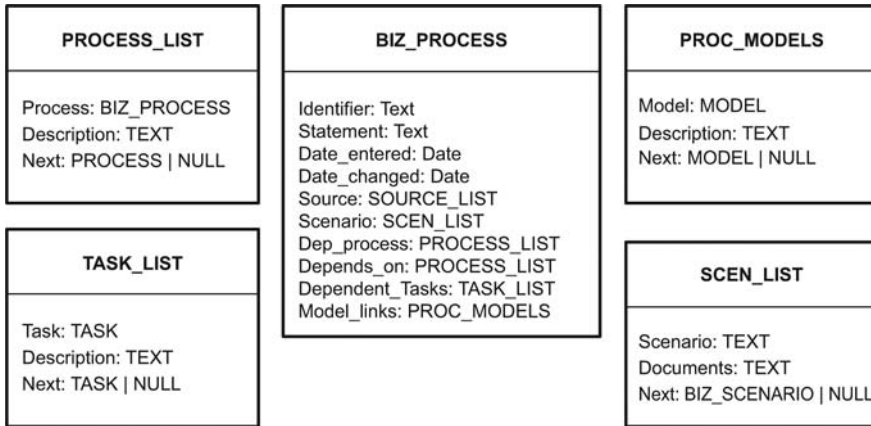


Figure 7.
The LRS task object

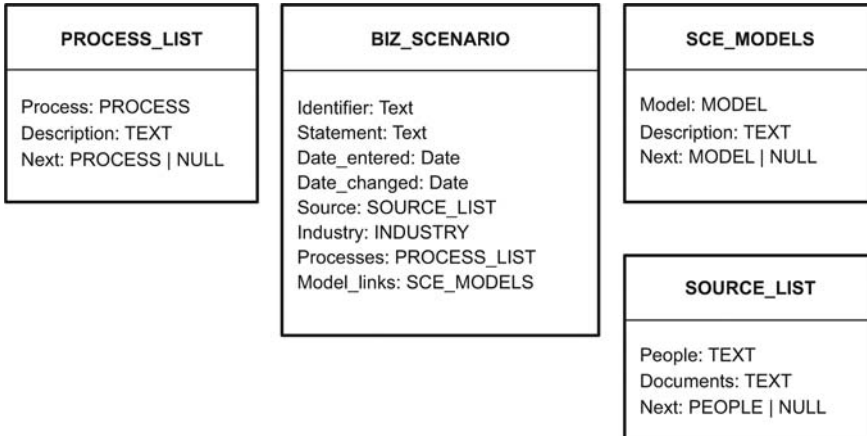


Figure 8.
The LRS scenario object

requirement by a provider who might consider implementing it into a feature, or a member of the expert group questioning the validity of the requirement. Thereafter members of the community can review and make arguments in favour of, or against, the acceptance of the requirement. At certain milestones the requirement is baselined to preserve its status before another round of reviews is carried out. Finally the requirement might be put to a vote, and if it is accepted, providers can then decide when and how to implement such a feature. Consistency checking is usually done to ensure the completeness of a requirement. Here academic research can help point ERP providers in the direction of automated tools and models that can help perform consistency checks to ensure a requirement meets certain consistency criteria.

Prioritisation of a requirement means deciding when to implement it. Requirements that have passed the analysis review stage become candidate requirements, but they still need to be prioritised before they are implemented. The prioritisation process can take the form of forums and discussion threads and a voting system where experts nominate and vote on which requirements they think should be in the next release. Weights can also be attached to candidate requirements. The academic group can suggest improvements in prioritisation techniques.

Requirements management in the LRS

The tool provides the facility to uniquely identify a requirement by attaching a unique identifier to every registered requirement. This unique identifier then makes it possible to add supplementary materials to further illustrate a requirement as it guarantees their association through the identifier. Content sharing features allow content in different formats to be shared. The tool also provides facilities for storage, retrieval, version control, relation to other requirements, and change requests.

Conclusions and future research

In a large-scale research project with interdisciplinary partners from business schools and computer science faculties (3G ERP Project) we designed a prototype for the LRS platform that can be used to collect, store, and display requirements from an international community of independent stakeholders. We extend the traditional concept of requirements engineering by applying collaborative aspects to it. There are several aspects where our platform goes beyond the traditional RE paradigm. The LRS platform:

- follows the Web 2.0 approach of sharing and providing free access to online information;
- addresses a world-wide community of users and business analysts;
- does not belong to a single company but is run by a team of researchers; and
- is independent of any specific ERP system (vendor).

The platform is an online information medium that is ubiquitously accessible to stakeholders with access to the internet. It helps overcome the limitations of paper documentation by granting ambient access to the knowledge field of ERP requirements.

Challenges that need to be addressed are the different cultural backgrounds, languages, and experiences of the users (Damian and Zowghi, 2002). The LRS platform

will harness the potential of its users to collaboratively develop a definition of terms in the online information realm, and thus result in a joint understanding of the underlying vocabulary (similar to the Wikipedia movement). Internet users are increasingly willing to contribute to the global knowledge base. The LRS platform makes use of this global movement by combining the advantages of social software with the fixed structure (hierarchy) provided by the requirements template. Once the platform is operational we will try to find ways to use the collected business requirements to (automatically or semi-automatically) transform them into formal descriptions that can be used to generate code. It will be interesting to see how the different paradigms in ERP software development will evolve in the next ten years. Today ERP software companies are collecting “their own requirements”, and most of them are learning through implementation projects. The ones with indirect sales channels are dependent on their implementation partners. With software getting more and more modularised (thinking of SOA) it is likely that customers will require increasingly individualised solutions. This makes the process of requirements collection even more challenging. The LRS platform as an online medium has the potential of becoming a valuable source not only for ERP developers, but also for academics and students.

References

- Alvarez, R. (2002), “Confessions of an information worker: a critical analysis of information requirements discourse”, *Information and Organization*, Vol. 12 No. 2, pp. 85-107.
- Archer, D. (2009), *Collaborative Leadership: How to Succeed in an Interconnected World*, Butterworth-Heinemann, Oxford.
- Berners-Lee, T. and Fischetti, M. (1999), *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*, Harper, San Francisco, CA.
- Boehm, B. (2002), “Get ready for agile methods with care”, *Computer*, Vol. 35 No. 1, pp. 64-9.
- Boyd, D.M. and Ellison, N.B. (2007), “Social network sites: definition, history, and scholarship”, *Journal of Computer-mediated Communication*, Vol. 13 No. 1.
- Boyd, D. and Heer, J. (2006), “Profiles as conversation: networked identity performance on Friendster”, *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, Kauai*, IEEE Computer Society Press, Colorado Springs, CO.
- Braffman, O. and Beckstrom, R. (2006), *The Starfish and the Spider: The Unstoppable Power of Leaderless Organizations*, Penguin Group, New York, NY.
- Cadiz, J.J., Gupta, A. and Grudin, J. (2000), “Using web annotations for asynchronous collaboration around documents”, *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, ACM, New York, NY, pp. 309-18.
- Cant, T. and McCarthy, J. (2006), *Tools for Requirements Management: A Comparison of Telelogic DOORS and the HiVe*, Department of Defence, Defence Science and Technology Organisation, Edinburgh.
- Chantavaridou, E. (2009), “Contributions of open access to higher education in Europe and vice versa”, *OCLC Systems & Services*, Vol. 25 No. 3, pp. 167-74.
- Czarnecki, K. (2002), “Domain engineering”, *Encyclopedia of Software Engineering*, Vol. 1, John Wiley & Sons, New York, NY, pp. 433-44.
- Damian, D.E. and Zowghi, D. (2002), “The impact of stakeholders’ geographical distribution on managing requirements in a multi-site organization”, *Proceedings of the IEEE Joint*

- International Conference on Requirements Engineering (RE'02)*, IEEE Computer Society, Essen, pp. 1-10.
- Davenport, T. (1998), "Putting the enterprise into the enterprise system", *Harvard Business Review*, Vol. 76 No. 4, pp. 121-31.
- Dearstyne, B. (2007), "Blogs, mashups, and wikis: oh, my!", *Information Management Journal*, Vol. 41 No. 4, pp. 25-33.
- Duncan-Howell, J. (2010), "Teachers making connections: online communities as a source of professional learning", *British Journal of Educational Technology*, Vol. 41 No. 2, pp. 324-40.
- Farnham, S., Kelly, S.U., Portnoy, W. and Schwartz, J.L.K. (2004), "Wallop: designing social software for co-located social networks", *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Kauai*, IEEE Computer Society Press, Colorado Springs, CO.
- Fasoldt, A. (2004), "Librarian: don't use Wikipedia as a source", *Syracuse Post Standard*, 25 August.
- Finkelstein, A. and Emmerich, W. (2000), "The future of requirements management tools", in Quirchmayr, G., Wagner, R. and Wimmer, M. (Eds), *Information Systems in Public Administration and Law*, Oesterreichische Computer Gesellschaft, Wien.
- Gotel, O.C.Z. and Finkelstein, A.C.W. (1994), "An analysis of the requirements traceability problem", *1st International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs, CO, pp. 94-101.
- Guzdial, M. and Turns, J. (2000), "Effective discussion through a computer-mediated anchored forum", *Journal of the Learning Sciences*, Vol. 9 No. 4, pp. 437-69.
- Hart, M., Johnson, R. and Stent, A. (2007), "More content – less control: access control in the Web 2.0", *Proceedings of the IEEE Web 2.0 Privacy and Security Workshop, Oakland, CA*, IEEE Computer Society Press, Colorado Springs, CO.
- Hull, E., Jackson, K. and Dick, J. (2005), *Requirements Engineering*, Springer, London.
- IEEE (1998), *ANSI/IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications – Description*, IEEE, Los Alamitos, CA.
- INCOSE (2009), "INCOSE requirement management survey", available at: www.paper-review.com/tools/rms/vencon.php?vendor=CASE%20Spec%208.0 (accessed 14 September 2009).
- Jackson, M. (1995), *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*, ACM Press, London.
- Johansson, B. and Sudzina, F. (2007), "Finding requirements that support strategic management in organizations", in Kaluza, J. (Ed.), *Strategic Management and Its Support by Information Systems*, University of Technology Ostrava, Celadna, pp. 112-19.
- Jones, A. and Preece, J. (2006), "Online communities for teachers and lifelong learners: a framework for comparing similarities and identifying differences in communities of practice and communities of interest", *International Journal of Learning Technology*, Vol. 2 Nos 2/3, pp. 112-37.
- Korfiatis, N., Poulos, M. and Bokus, M. (2006), "Evaluating authoritative sources using social networks: an insight from Wikipedia", *Online Information Review*, Vol. 30 No. 3, pp. 252-62.
- Kotonya, G. and Sommerville, I. (1998), *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, New York, NY.
- Kovitz, B.M. (1998), *Practical Software Requirements: A Manual of Content and Style*, Manning Publishing Company, Greenwich, CT.
- Krebs, J. (2005), "Dissecting business from software requirements", available at: www.ibm.com/developerworks/rational/library/aug05/krebs/index.html (accessed 14 September 2009).

- Kulathuramaiyer, N. (2007), "Mashups: emerging application development paradigm for a digital journal", *Journal of Universal Computer Science*, Vol. 13 No. 4, pp. 531-42.
- Lee, M. and Lan, Y. (2007), "From Web 2.0 to conversational knowledge management: towards collaborative intelligence", *Journal of Entrepreneurship Research*, Vol. 2 No. 2, pp. 47-62.
- Lipczynska, S. (2005), "Power to the people: the case for Wikipedia", *Reference Reviews*, Vol. 19 No. 2, pp. 6-7.
- Mann, F., von Walter, B., Hess, T. and Wigand, R.T. (2009), "Open access publishing in science", *Communications of the ACM*, Vol. 52 No. 3, pp. 135-9.
- Matulevicius, R. (2004), "How requirements specification quality depends on tools: a case study", *The 16th International Conference on Advanced Information Systems Engineering*, Springer Verlag, Berlin, pp. 353-67.
- Nonnecke, B., Andrews, D. and Preece, J. (2006), "Non-public and public online community participation: needs, attitudes and behavior", *Electronic Commerce Research*, Vol. 6 No. 1, pp. 7-20.
- Orlowski, A. (2005), "Wikipedia science 31% more cronky than Britannica's", *The Register*, 16 December, available at: www.theregister.co.uk/2005/12/16/wikipedia_britannica_science_comparison/ (accessed 15 October 2009).
- Panchal, J. and Fathianathan, M. (2008), "Product realization in the age of mass collaboration", *34th ASME Design Automation Conference*, Paper DETC2008-49865, American Society of Mechanical Engineers, New York, NY.
- Piedra, N., Chicaiza, J., López, J., Tovar, E. and Martínez, O. (2009), "Open educational practices and resources based on social software, UTPL experience", *Proceedings of the 2009 Euro-American Conference on Telematics and Information Systems: New Opportunities to Increase Digital Citizenship*, ACM, New York, NY, pp. 1-8.
- Power, N.M. (2002), "A grounded theory of requirements documentation in the practice of software development", PhD thesis, School Dublin, Dublin City University, Dublin.
- Robertson, S. and Robertson, J. (1999), *Mastering the Requirements Process*, Addison-Wesley, New York, NY.
- Ruth, A. and Houghton, L. (2009), "The wiki way of learning", *Australasian Journal of Educational Technology*, Vol. 25 No. 2, pp. 135-52.
- Sabol, T. and Delina, R. (2004), "Scenario planning", *Ekonomicky Casopis*, Vol. 52 No. 8, pp. 942-56.
- Scheer, A.-W. (2000), *ARIS – Business Process Modeling*, 3rd ed., Springer, Heidelberg.
- Schubert, P. and Wölflé, R. (2007), "The eXperience methodology for writing IS case studies", *Proceedings of the 13th Americas Conference on Information Systems (AMCIS)*, Keystone, CO.
- Seadle, M. (2005), "Copyright in the networked world: authors' rights", *Library Hi Tech*, Vol. 23 No. 1, pp. 130-6.
- Soloman, G. and Schrum, L. (2007), *Web 2.0: New Tools, New Schools*, International Society for Technology in Education, Eugene, OR.
- Sommerville, I. and Sawyer, P. (1997), *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, New York, NY.
- Straus, D. and Layton, T. (2002), *How to Make Collaboration Work: Powerful Ways to Build Consensus, Solve Problems, and Make Decisions*, Berrett-Koehler Publishers, San Francisco, CA.

-
- Tapscott, D. and Williams, A.D. (2006), *Wikinomics: How Mass Collaboration Changes Everything*, Penguin Books, London.
- Thayer, R.H. (1997), *Software Systems Engineering: An Engineering Process*, IEEE Computer Society Press, Los Alamitos, CA.
- Wieggers, K.E. (2003), *Software Requirements*, Microsoft Press, Redmond, WA.
- Worley, J.H., Chatha, K.A., Weston, R.H., Aguirre, O. and Grabot, B. (2005), "Implementation and optimisation of ERP systems: a better integration of processes, roles, knowledge and user competencies", *Computers in Industry*, Vol. 56 No. 6, pp. 620-38.

About the authors

Femi Adisa has a Master's degree in Information Technology from the University of Turku. He is a PhD Fellow at the Centre of Applied ICT at Copenhagen Business School. Femi Adisa is the corresponding author and can be contacted at: fa.caict@cbs.dk

Petra Schubert has a PhD and a Master's degree in Information and Technology Management from the University of St Gallen. In addition she has a Master's degree in Information Management from the Community of European Management Schools. She is a Full Professor at the Centre of Applied ICT at Copenhagen Business School. Previously she worked at the University of Koblenz-Landau.

Frantisek Sudzina has a PhD in Sectoral and Intersectoral Economies and a Master's degree in Economics and Business Management from the University of Economics Bratislava. In addition he has a Master's degree in Mathematics and a Master's degree in Computer Science from Safarik University Kosice. He is a Researcher in the Institute of Economic Research at the Slovak Academy of Science. Previously he worked at Copenhagen Business School.

Björn Johansson has a PhD and a licentiate degree in Information Systems Development from Linköping University. He is an Associate Professor in the Department of Informatics at the School of Economics and Management of Lund University. Previously he worked at Copenhagen Business School.

To purchase reprints of this article please e-mail: reprints@emeraldinsight.com
Or visit our web site for further details: www.emeraldinsight.com/reprints

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.