CRESUS-T: A COLLABORATIVE REQUIREMENTS ELICITATION SUPPORT TOOL

Paul Stynes[1], Owen Conlan[2] and Declan O'Sullivan[3]

[1] School of Computing, National College or Ireland, Dublin, Ireland
[2,3]ADAPT centre, Trinity College Dublin, Dublin 2, Ireland

ABSTRACT

Communicating an organisation's requirements in a semantically consistent and understandable manner and then reflecting the potential impact of those requirements on the IT infrastructure presents a major challenge among stakeholders. Initial research findings indicate a desire among business executives for a tool that allows them to communicate organisational changes using natural language and a model of the IT infrastructure that supports those changes. Building on a detailed analysis and evaluation of these findings, the innovative CRESUS-T support tool was designed and implemented. The purpose of this research was to investigate to what extent CRESUS-T both aids communication in the development of a shared understanding and supports collaborative requirements elicitation to bring about organisational, and associated IT infrastructural, change. In order to determine the extent shared understanding was fostered, the support tool was evaluated in a case study of a business process for the roll out of the IT software image at a third level educational institution. Statistical analysis showed that the CRESUS-T support tool fostered shared understanding in the case study, through increased communication. Shared understanding is also manifested in the creation of two knowledge representation artefacts namely, a requirements model and the IT infrastructure model. The CRESUS-T support tool will be useful to requirements engineers and business analysts that have to gather requirements asynchronously.

KEYWORDS

Collaborative Requirements Elicitation, Shared Understanding, and Semantically enabled Web Services, Ontology.

1. INTRODUCTION

The Software Requirements knowledge area is a discipline of software engineering that is concerned with requirement elicitation, analysis, specification and validation of software requirements [1]. As part of the Software Requirements knowledge area, today's requirements engineer has to decide the best strategy for identifying the software that fulfils the business needs and aligning that software with the organisation's IT infrastructure. The Requirements Elicitation activity involves intense communication between stakeholders [2]. Stakeholders are anyone interested in, or affected by the outcome of the future system. Stakeholders typically include business executives such as sponsors of the project, IT architects that build the system, business users that will operate the future system and requirements engineers that gather the requirements of the future system. Requirements are defined as a condition or capability defined by stakeholder to solve a problem or achieve an objective [3].

The stakeholders often find it difficult to articulate their requirements [4]. Poor communication makes discovery of the requirements a challenge. The use of elicitation techniques such as interviews, group work, prototypes, and models may provide a solution to this challenge [1, 3 , 5]. Sutcliffe [6] suggests the combination of requirement elicitation techniques such as the use of prototype artefacts and scenarios. Another useful elicitation technique is modelling. Modelling

DOI: 10.5121/ijsea.2018.9202

deals with understanding an organisation through the creation of enterprise models of the IT infrastructure.

Requirements Elicitation is characterised by frequent communication [2]. Frequent communication depends on the richness of the communication channel such as face to face or email. Effective communication has been difficult to achieve and is a recurring problem in the elicitation of requirements [7, 8, 2, 9]. Coughlan and Macredie [2] present a communication framework that promotes effective communication for an organization and its stakeholders in attempting to integrate technology. The challenge is to create a richer channel for communication that embeds effective communication techniques for mediating the communication of requirements.

A PowerPoint mock-up of a scenario-based communication tool was created and an initial evaluation was conducted with ten business executives in higher education. The purpose of the evaluation was to examine the desire for a tool that allows business executives communicate about organisational and IT infrastructural changes [10]. The analysed data produce results that indicate a desire for such a tool with open comments such as "The idea seems to be very good, especially the natural language interface, which I particularly like", and "Good to specify objectives through the use of goals". Quantitative results reinforce the comments. They indicate that the majority of this small sample group (6 executives) like the approach of specifying their goals and rules in natural language. In addition, the majority (7 executives) liked the approach where the system automatically identifies a business process that may solve the executive's goal, and applies rules to services in the process. While this was only a limited survey the results indicate promise. The executives intuitively understood the approach being proposed. That is to augment the requirements gathering process with semantics that drives the formulation of organisational changes using controlled natural language. In addition, they also liked the simulation of the IT infrastructure that supports those changes. Moreover, they strongly indicated that they believed such an approach was desirable. Building on the evaluation, CRESUS-T was designed and implemented as a collaborative requirements elicitation tool. The tool allows stakeholders communicate requirements guided by an ontological domain model, then validate the requirements, and finally create a prototype model of the IT infrastructure that supports those requirements.

The remainder of this paper describes the related work in section 2, then an outline of research in section 3, followed by architecture and implementation in section 4. Section 5 describes the evaluation, followed by limitations, and future work.

2. RELATED WORK

Traditional requirements engineering tends to involve the requirements engineer meeting with the end user to identify the requirements, write the requirement specification, and hand them to the development team to develop the software in-house [11]. Today, the majority of software is not developed in house but is available for purchase or free, such as customer resource management (CRM), and open source software [11]. Today's requirements engineer has to decide the best strategy for identifying the software that fulfils the business needs and aligning that software with the organisations IT Infrastructure.

Requirements elicitation is characterised by frequent communication [2]. However effective communication has been notoriously difficult to achieve and is a recurring problem in the elicitation of requirements [7, 8, 2, 9]. Rogers and Kincaid [12] describes communication as a dynamic process of idea and knowledge generation, which occurs over time through interaction with others and which leads to shared understanding and collective action. The work of Lind and

Zmud [13] showed empirically that frequent communication helped create a shared understanding between technology providers and business users regarding the importance of technology in supporting the business activities. They also demonstrated that communication richness helped create a shared understanding and this was determined by the type of communication channel such as face to face, computer mediated or written documents. Johnson and Lederer [14] extended the work of Lind and Zmud [13] to communication between the Chief Executive and Chief Information Officers. Their work showed that frequent communication helped create a shared understanding of the current and future role of IT in the organisation. They also reported more frequent communication using communication channels such as face to face and email that were perceived to be richer. Coughlan and Macredie [2] conducted an analysis of effective communication in requirements elicitation and concluded with four recommendations for effective communication for an organization and its stakeholders in attempting to integrate technology namely, include business users in the design; select an adequate mix of IT architects and business users who then interact on a collaborative basis; the incorporation of communication activities that relate to knowledge acquisition, knowledge negotiation and user acceptance; the use of elicitation techniques for mediating communication for the requirements of a system such as interviews, brainstorming, prototyping and scenarios.

Successfully understanding an organisation's requirements in such a manner that their impact on the IT infrastructure can be analysed and discussed, presents a major challenge [13-15] between the business executive, IT architect and other stakeholders. The most significant problem arises in communicating the requirements desired in a semantically consistent and understandable manner and then reflecting the potential impact of those requirements on the IT infrastructure. According to Van Lamsweerde [16], the goal of domain understanding is to understand the problem, and the application domain of the problem. The IEEE Computer Society [1] indicates that the requirements engineer needs to acquire and structure available knowledge about the application domain of the problem. Recording this application domain knowledge makes communication easier and future understanding more reliable [11]. The degree and manner of specifying application domain knowledge varies and is dependent on the formality used. Glossaries and data dictionaries are informal and reside at one end of the spectrum. Moving to the other end are formal knowledge representations such as ontologies. Ontology is understood as a way of structuring and specifying the meaning of knowledge in an application domain of the problem [17] in [18]. The IEEE Computer Society [1] indicates that it is good practice to follow an ontological approach.

Gruber [19] defines an ontology “as an explicit specification of a conceptualisation”. In the context of artificial intelligence, an ontology is referred to as an engineering artefact that is based on the vocabulary consisting of terms and relationships of an application domain in addition to the rules for combining the terms and relationships between the terms[20, 21]. The motivation for using ontology is that it allows stakeholders and other systems have a shared understanding of the structure of information [19, 22]. In addition, they are machine-interpretable and are amenable to semantic analysis.

Some of the early work on ontologies in requirements engineering centred around knowledge representation languages [23, 24, 25, 26, 27]. The emergent research area of the semantic web has facilitated a renewed interest in the adoption of ontologies for requirements engineering. The semantic web builds structure and meaning to data that is web accessible [28]. Semantic Web technologies such as XML [29], RDF [30] and OWL [31] enable stakeholders to create requirements on the Web, build vocabularies, and write rules for handling the requirements. XML adds arbitrary structure to the stakeholder’s documents but does not describe what they mean. Meaning is expressed by RDF, using sets of triples, where each triple is like the subject, verb and object of an elementary sentence. The benefit of ontologies for requirements engineering, is the ability to explicitly model domain knowledge in a machine interpretable way. Ontologies provide

reduced ambiguity, increased meaning, formality and support for automated reasoning. This allows for requirements traceability, consistency checking, in addition to automatically generating the software specifications [32].

Castañeda, Ballejos, Caliusco, & Galli [18] comprehensively reviewed and presented the use of ontologies in requirements engineering. One such formalised knowledge representation approach involves the use of ontologies to represent application domain knowledge [33, 34, 35]. Guarino [35] distinguished between ontology-aware and ontology-driven systems [36]. An ontology-aware system knows of the existence of an ontology and can query the ontology. In an ontology-driven system, the ontology is placed as a component within the system. The reason for using an ontology-driven system is that it enables communication through messages that contain expressions formulated from the ontology. Guarino [35] refers to this as ontology-driven communication. The advantage of this approach is that machines can automatically reason over and understand the communication. The ontology can impact the components of a system such as the application program, user interface and database. In addition, Guarino [35] first describes the use of the ontology to develop the static part of the program in the form of type or class declarations and procedures. The advantage of this approach is that it ensures that domain knowledge is represented in the application program. Abbott's [37] technique for developing software programs from informal English descriptions to derive data types from common nouns, objects from proper nouns and operators (functions, procedures) from verbs, provides insight in how to generate software programmes from informal but precise English. Abbott's technique demonstrates that in a highly automated process, a formal knowledge representation model which is complete and precise is amenable to automatic generation of the software programmes [38]. Booch extended Abbott's technique to object oriented development [39]. Secondly, in the context of the user interface components, Guarino describes an ontology as the embodiment of semantic information on the constraints imposed on the classes and relationships from the application domain. Using the ontology for the user interface allows for semantic checking of any violations on those constraints. Thirdly, Guarino [35] discusses using ontologies for databases where ontologies that integrate lexical resources like WordNet may support the analysis of natural language requirements. Such ontologies can be mapped to schemas for different types of databases [40] in [35].

The difficulty in understanding knowledge-based techniques is a well-known issue with ontologies. This is only made worse in requirements engineering since requirements are aimed at a range of stakeholders with different backgrounds and knowledge [32]. Natural language is one of the best mediums for communicating and understanding requirements. This allows stakeholders and especially business executives who may be unfamiliar with modelling notations to still understand and validate requirements. However, natural language is ambiguous and can be easily misinterpreted [41]. If the language is controlled, that is, the system prompts the user on the construction of the sentences, then the language becomes machine readable. By incorporating semantics, then other systems will have the same meaning for the sentences that are created. A controlled natural language (CNL) is a precisely defined subset of full English that can be used for communicating the organisation's needs in such a way that it can be automatically reasoned over by machines and thus removes the ambiguity issues of natural language. Of all the controlled natural languages, ACE appears to have been the most studied and provides the most support. ACE is a knowledge representation language used in applications that include ontology authoring [42] and ontology verbalization [43]. A predictive editor [44] guides stakeholders, word-by-word in the construction of a sentence that complies with ACE. The sentence can be converted to an ontological format using the ACE Parsing Engine (APE) [45]. APE ensures that the sentence is ACE compliant and then converts the natural language sentence into an OWL representation. The approach described with ACE embedded in an ontology driven system shows the potential for a natural language interface to utilize semantic inference to communicate the stakeholder's

requirements in an understandable manner. In order to reflect the potential impact of those requirements on the IT infrastructure it is necessary to look at non-functional requirements.
Ameller, Ayala, Cabot, & Franch [46] in a survey of software architects found that a prototype was useful for eliciting non-functional requirements. They observed that the software architects were the main source of non-functional requirements. It is recognised that non-functional requirements guide the shape of the IT Infrastructure [47, 48]. Nuseibeh and Easterbrook [4] outline a roadmap in which there is a need for better understanding of the impact of software architectural choices on the prioritisation and evolution of requirements. While work in software architectures has concentrated on how to express software architectures and reason about their behavioural properties, there is still an open question about how to analyse what impact a particular architecture choice has on the ability to satisfy current and future requirements. The architectural choices may be guided by reference architectures. Reference architectures represent models of domain specific software structures [38]. They provide a template solution for architecture of a particular domain. The influences from the analysis of non-functional requirements indicate a need to include reference architecture as the basis of the prototype to facilitate the elicitation of non-functional requirements and represent a model of the IT infrastructure.

Modelling approaches such as enterprise modelling are used as drivers to prompt further requirements elicitation [4]. Enterprise modelling is often used to capture the purpose of a system, by describing the behaviour of the organisation through the business processes and services that it provides [49]. Chandrasekaran, Silver, Miller, Cardoso, & Sheth, [50] recognised that there are synergies between models and web services. One of the approaches that Chandrasekaran, Silver, Miller, Cardoso, & Sheth [50] propose is the creation of models from web services in order to provide a high fidelity between the model and the IT infrastructure that comprises of the web services. This approach provides an ability to plug real web services into models, thus creating an alignment between the model and the IT infrastructure that utilises as much realistic data as possible.

In summary, the requirements that emerge from the state of the art are as follows :- to structure and represent application domain knowledge and model knowledge through the use of formal knowledge representation techniques to create an ontology driven system that supports ontology-driven communication; to guide the stakeholders through controlled natural language in the creation of formal knowledge representation artefacts such as the requirements model and IT infrastructure model; to create a rich channel of communication that embeds techniques for requirements elicitation such as prototyping and modelling; to generate a prototype of the IT infrastructure from the vocabulary of the ontology through developing the static part of the program in the form of type or class declarations and procedures; to use a prototype that allows the analysis of an architectural choice to identify non-functional requirements; to incorporate Coughlan and Macredie's [2] four recommendations for effective communication in the CRESUS-T support tool.; and to create enterprise models from web services that form the basis of alignment between the business needs and IT infrastructure. Influences from the literature are used in the architectural design of the Collaborative Requirements Elicitation support tool, named CRESUS-T in section 4.

3. OUTLINE OF RESEARCH

3.1. RESEARCH BACKGROUND

This research explores to what extent CRESUS-T both aids communication in the development of a shared understanding and supports collaborative requirements elicitation to bring about organisational, and associated IT infrastructural, change. In addition, the experiment controls for

title, role, level of service, and academic qualifications of employees at the National College of Ireland.

The experiment employed a pre-test-post-test control group design with matching participants in the experimental and control groups to evaluate the frequency of communication, participant perceptions of attaining a shared understanding of the IT infrastructure supporting organisational change, and, control variables that relate to title, role, level of service and academic qualifications.

In this context CRESUS-T is generally defined as a collaborative communication tool that allows stakeholders to communicate requirements from an ontological domain model, validate the requirements, and create a model of the IT infrastructure that supports those requirements. Communication will encompass one dimensions namely, frequency of communication. Shared understanding is generally defined as the stakeholders' perception that the requirements represent the organisational and associated IT infrastructural change.

Organisational and associated IT infrastructural change will be generally defined as a series of IT systems denoted by web services that are semantically enabled and may access ontological data that represents the organisational changes as denoted by the stakeholders' requirements.

3.2. METHOD

The following steps were carried out in developing CRESUS-T namely a literature review, evaluation of a scenario-based communication tool, implementation of CRESUS-T and evaluation of CRESUS-T. The literature review focused on requirements elicitation, communicating requirements using a controlled natural language interface, shared understanding of organisational and associated IT infrastructural changes, and prototype models that comprises of semantically enabled web services representing the evolution of the organisational IT Infrastructure.

4. ARCHITECTURE AND IMPLEMENTATION

CRESUS-T was designed around a communication mechanism and a machine translation component that translates the vocabulary of the application domain and reference architecture into the IT infrastructure Model as described in Figure 1.

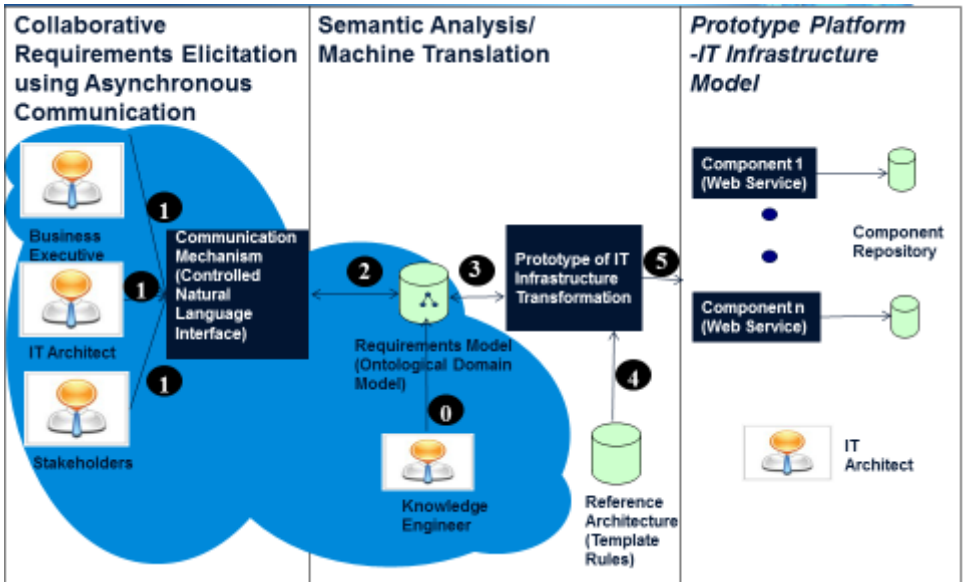

Figure.1 CRESUS-T Architecture

The Communication Mechanism consists of a controlled natural language interface that facilitates ontology-driven natural language communication between the Business Executive, IT Architect and Business Users (see message 1) in Figure 1. The stakeholders can collaboratively communicate and validate requirements that are stored in the Requirements Model (see message 2). The Prototype of IT Infrastructure Transformation component automatically generates a prototype of the IT infrastructure (see message 5) based on the vocabulary of the application domain (see message 3) and a reference architecture (see message 4).

The communication mechanism is responsible for handling the communication between the stakeholders and the underlying formal knowledge representation of the application domain. Using ontology-driven communication, the stakeholders are prompted in the construction of the requirements. The vocabulary for constructing the requirements is taken from the content words of the formal knowledge representation of the underlying application domain. The approach to verbalise the formal knowledge representation into natural language is shown in Table 1[43].

Table 1 Verbalising the Formal Knowledge Representation in Natural Language

Corresponding verb and noun phrases	Knowledge Representation of the Application Domain	Example
Common noun	Named Class	Lecturer
Proper noun	Named Object	JohnSmith
Transitive verb	Named Property	Teaches

The formal knowledge representation concepts of Object, Class and Property are mapped to nouns and verbs. The grammar of the language defines and constrains the form and meaning of the sentences.

The sentences have the following structure: -

Subject + verb + complement [51]

Where subject and complement are represented by nouns. In a course registration system in a third level educational institution for example, this would be similar to John Smith teaches Software Development with the semantic meaning of Lecturer teaches Module. The resultant requirements are validated and form part of the requirements model.

The rich channel of communication integrates Coughlan & Macredie [2] techniques for mediating the communication of requirements as follows: -

- A web based tool that allows the stakeholders to collaborate asynchronously.
- The incorporation of communication activities that centre around the creation of artefacts such as the requirements model, and IT infrastructure model that facilitate knowledge acquisition, knowledge negotiation and user acceptance.
- The use of elicitation techniques for mediating communication for the requirements of a system such as prototyping.

Machine translation automatically generates the IT infrastructure model based on the nouns and verbs identified from the concepts and actions of the underlying formal knowledge representation of the application domain. The resultant nouns and verbs are applied to a choice of reference architectures to create a prototype consisting of architectural components such as software programs and associated data models. Deployment of the architectural components results in the creation of the prototype platform that represents the evolution of the IT Infrastructure model. Stakeholders can retrieve, create, delete and modify data through the software program interfaces. Reference architectures represent models of application domain software structures [38]. They provide a template solution for the IT infrastructure. The code generation process is described in Figure 2.

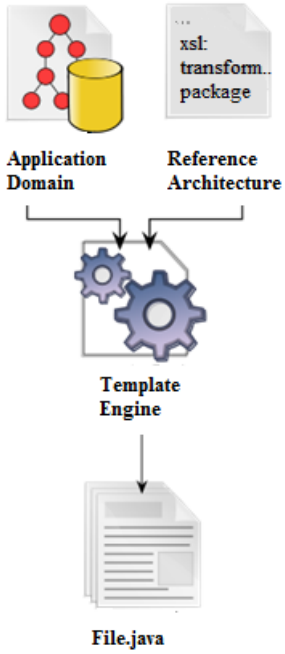

Figure.2 Code Generation Process

The code generation process uses the reference architecture and sets of rules for tailoring the prototype of the IT infrastructure Model based on extending Abbots [37] and Booch's [39] textual analysis technique to the vocabulary from the formal knowledge representation of the application domain [38]. Table 2 describes how to derive classes (or web services) from common nouns, objects (or web parameters) from proper nouns and operations (or web methods) from verbs arising out of the vocabulary from the formal knowledge representation.

Table 2 Identifying Candidate Software Programmes

Vocabulary	Booch (1986) Object Orientation	Service Oriented Architecture
Common noun	Class	Web Service
Proper noun	Object	Web Parameter
Transitive verb	Operation	Web Method

A template engine combines the application domain ontology with a reference architecture in order to automatically generate the source code of the prototype as described in Figure. This prototype represents the IT infrastructure model and when deployed represents a mechanism for the stakeholders to analyse the architectural choice. The stakeholders can test the architectural components, provide information through the interfaces, and view the databases directly.

The knowledge representation artefacts that are created from the CRESUS-T support tool are the Requirements Model and IT Infrastructure Model. The formal knowledge representation artefacts represent a manifestation of the stakeholders shared understanding.

CRESUS-T is a web based application that is implemented in Java using the echo web framework [52]. The web based application was hosted on the Amazon Web Service at http://54.238.49.81:8081. Using ontology-driven communication, the stakeholders communicate

through the ACE editor [44]. Attempto Controlled English Parsing Engine (APE) [53]ensures that the grammar for the requirements is based on a subset of natural language, namely Attempto Controlled English [51]. In addition, the parsing engine converts the sentences created in ACE editor into RDF triples. The APE is deployed on port 8000. The ACE Editor and APE are 3rd party products in the CRESUS-T architecture. The ACE editor is a predictive authoring tool that guides the stakeholders during the construction of the requirements. The ACE editor source code is part of ACEWiki [44]. The application domain ontology is initially created using an external editor, protégé [54]. The ontology is imported into the server side implementation of CRESUS-T and stored as a Jena ontological model [55]. Jena is an open source semantic web framework for building Java applications. Jena provides support to create and read RDF and OWL. The Jena framework uses the Pellet reasoner [56] to reason over the OWL application domain ontology. The reference architecture is a set of template instructions for transforming XML into Java source code. The reference architecture is based on an XSLT stylesheet. XSL Transformations (XSLT) provides the ability to transform an XML document into other formats such as Java source code representing a prototype of the IT Infrastructure. The reference architecture provides the instructions that form the input to the transformation API for XML in order to generate the Java source code. Initially a lexical analysis is performed on each RDF statement in the application domain ontology. Lexemes are created from the semantic meaning of the subject, predicate and object of each RDF statement. A Lexeme is a sequence of characters in the source program that represent an instance of a token [57]. The semantic meaning of the subject and object is derived from the class of the RDF individual of that subject and object. The resultant lexemes represent the Web Service, Web Method and Web Parameter tokens. A token is a key word that the Prototype of the IT Infrastructure Transformation Component processes. The lexemes are represented in the Prototype Data Structure XML document. The Prototype Data Structure XML document is created using JDOM [58]. JDOM provides a Java-based solution for accessing, manipulating, and outputting XML data. The Prototype Data Structure XML document is stored in the eXist database [59] located on port 8680. The lexemes that represent the tokens in the XML document are plugged into template code from the reference architecture in order to generate source code that can be compiled and executed using the transformation process. The Web Services from the prototype of the IT infrastructure can access their instance data from XML documents in the Component Repository stored in the eXist database on port 8680. The XSLT processor in TrAX transforms the prototype Data Structure XML document and XSLT instructions into Java source code. The Java source code is stored in a skeleton NetBeans project [60]. Apache ANT [61] is used to compile, build and deploy the NetBeans project to a glassfish server on port 8081. The deployment of the Java source code in conjunction with the component repository represents the prototype of the IT infrastructure.

5. EVALUATION

5.1. BACKGROUND

The School of Computing at NCI had created a conversion programme in response to Government tenders to provide courses that would up-skill the workforce. The conversion programme required desktop computers with specialised software to run the practical laboratory sessions. The IT Department are responsible for the business process that involves creating the IT software image and rolling it out to the desktop computers. The IT software image contains the specialised software required by the conversion programme. The business process had issues around collaboration and communication resulting in delays with completing the process. The IT Department wanted to develop the IT infrastructure that would automate the business process and engage in a requirement gathering exercise. This environment provided an opportunity to evaluate the CRESUS-T support tool for eliciting requirements of the future system that would align to the business process.

All employees from the School of Computing and the IT Department in NCI were contacted to identify if they would participate in the study. Thirteen employees that consisted of eight employees from the School of Computing and five from the IT department agreed to take part in the study. One employee that participated was the business process owner and only took part in the interview. The remaining twelve employees participated in the experimental study. The academics consisted of four course directors (lecturer grade II), one lecturer grade I, one fulltime lecturer grade II from the school of business, one postdoctoral research fellow and one support tutor. All academics lectured in the school of computing. The IT personnel consisted of one senior IT administrator, one IT support specialist and two IT support personnel.

The senior IT administrator was interviewed in NCI to determine the problem domain description for the roll out of the IT software image. The interview was conducted in a natural work environment in NCI for 30 minutes. The interview comprised of nine questions that assessed the operation of the business process. The problem domain description artefact that represented the business process for the roll out of the IT software image was created from an understanding arising out of the interview. The problem domain description was validated with the senior IT administrator. The results of the interview formed the basis of the scenario that was given to the control and experimental groups.

5.2. OPERATIONALISATION OF VARIABLES

In this context, communication is operationalised by the frequency of communication. Frequency of communication indicates the degree to which messages and responses take place between the business executive, IT architect, and staff.

Shared understanding indicates the degree to which the business executive, staff and IT architect perceive that the IT infrastructure supports the organisational change faithfully.

5.3. METHODOLOGY

Data was captured through online questionnaires, a workshop, logs, observations and a debriefing session. The experimental design for this study used a randomised matched pairs design. A matched pairs design is a special case of a block design. As part of the matched pairs design, twelve of the employees were exposed to the level of service and qualifications questionnaire pre-test.

The aim of the level of service and qualifications questionnaire was to evaluate the blocking factors based on factual information such as the employee's role in NCI, level of service with NCI and academic qualifications based on the National Framework of Qualifications (NFQ) of Ireland [62]. The role allowed the creation of homogenous blocks that determined if the employees were course directors, IT administrators or lecturers. The level of service allowed the creation of homogenous blocks that determined the employees' years of service within NCI ranging from 1-2 years, 3-4 years, 5-6 years, 7-8 years, 9-10 years and greater than 10 years. The academic qualifications allowed the creation of homogenous blocks that determined the employees NFQ level at level 8 (Honours Degree), level 9 (Master's Degree) and level 10 (PhD Degree). The control information that related to employee's role, level of service, and academic qualifications were used in the analysis for matching employees was derived from the pre-test questionnaire's data.

Eight employees from the School of Computing and four employees from the IT Department were assigned to the matched pairs on the basis of their scores in the pre-test questionnaire. The employees were randomly assigned based on one subject of each pair to the experimental group A and C, and the other subject to the control group B and D.

The next step involved defining the ontological domain model in a workshop that involved a brainstorming session with each group where they created a conceptual model of the application domain. The conceptual model was converted into an ontological domain model. The ontological domain model was used to constrain the words in the controlled natural language interface and the naming conventions for the automatically generated web services. The employees were invited to take part in a workshop. The workshop was conducted in a natural work environment in NCI over a 30-minute period. As the ontology for each group is similar, only an analysis of the outcome from group A's workshop is described in section 5.

The next step was to gather the requirements and generate and test a prototype of the IT Infrastructure. The subjects were invited to attend the experiment in the ICELT research laboratory. However due to a logistical reason the participants completed the experimental study in a natural work environment in their own offices at NCI. The experiment took place over a one-hour period. The experimental groups A and C were administered the treatment of the CRESUS-T support tool and email. The control groups B and D used email only. A log of all communication between the subjects was stored in a communal email for each group. In addition, CRESUS-T logged all communication messages. The information that related to the communication feature used in the analysis was derived from the communal emails and CRESUS-T's logged data. CRESUS-T logs and email responses between employees were evaluated to identify the frequency of communication. The goal of capturing this information was to determine if the CRESUS-T support tool leads to increased communication. This metric was important as frequent communication helps create a shared understanding between stakeholders regarding the importance of technology in supporting the business needs.

The experimental groups A and C were then administered the simulation based communication tool post-test questionnaire. The questionnaire evaluated if the prototype supports organisational change faithfully. The goal for capturing this information was to determine if the automatic generation of a prototype that represents the IT infrastructure represents a realistic solution. This metric was important as it demonstrates an ability to plug real IT systems such as web services into the prototype which can lead to richer requirements and is the basis for aligning the IT infrastructure with the business process.

The performance of the experimental and control groups was compared in the post-test(s) using tests of statistical significance in terms of frequency of communication, and if the prototype supports organisational change. IBM's SPSS tool [63] was used to manage quantitative and qualitative data. An analysis of the results of CRESUS-T logs, email and the post-test are described in section 5.

6. RESULTS AND DATA ANALYSIS

Ошибка! Источник ссылки не найден. describes the employee's role, title, number of years of service with the National College of Ireland and their level of educational qualification attained based on the National Qualification Framework. Academic's 2, 3, 5 and 8 with a title of lecturer grade II and where their role indicated that they are programme directors were assigned to the business executive role. The academics 7, 11, 10 and 12 whose titles were lecturer grade I, lecturer grade II, postdoctoral research fellow and support tutor, and where their role indicates that they lecture in the School of Computing were assigned to the business user role. The IT

personnel 1, 4, 6 and 9 with the titles senior IT administrator, IT support specialist and IT support personnel were assigned to the IT architect role. The employees were pair-matched based on their number of years of service with NCI and their level of educational qualification attained based on the national qualification framework. The matched employees were 2 and 3; 7 and 11; 4 and 6; 5 and 8; 10 and 12; and, 1 and 9. Employees were randomly allocated to the experimental and control groups as follows: Employees 2, 7 and 1 to experimental group A; Employees 3, 11 and 9 to control group B; Employees 8, 10 and 4 to experimental group C; and Employees 5, 12 and 6 to control group D.

Table 3 Matching participants based on level of service and academic qualification

Role	Name	Title	Number of years of service with NCI						The level of education attained based on the National Qualification Framework		
			1-2 years	3-4 years	5-6 years	7-8 years	9-10 years	>10 years	Honours Degree	Masters	PhD
Programme Director	2	Lecturer II				X				X	
	3	Lecturer II				X				X	
	5	Lecturer II			X						X
	8	Lecturer II					X			X	
Lecture in School of Computing	7	Lecturer I	X						X		
	10	Computing Support Tutor	X							X	
	11	Post-Doctoral Research Fellow	X						X		
	12	Lecturer II						X		X	
IT personnel	1	Senior IT Administrator				X				X	
	4	IT support		X					X		
	6	IT support		X					X		
	9	IT Support Specialist				X			X		

Legend	
	Experimental group 1
	Control group 2
	Experimental group 3
	Control group 4

The match between experimental group A and control group B is excellent with one difference in educational qualifications of the employees in the role of the IT architect. The match between experimental group C and control group D indicates that the cumulative experience gained in work benefits the control group D, whereas the cumulative experience in educational levels benefit the experimental group C. On balance years of experience may make up the difference with the educational level.

The results from the brainstorming session define the application domain model and the approval process for any requirements that will be identified. The application domain model is converted into an ontology domain model which is used to constrain the grammar in the controlled natural language interface and the naming conventions in the automatically generated web services. A sample of the conceptual model and approval process that one of the group's identified, related to the concepts of Module and Software. An example of a requirement was "Module requires Software" with instance data that represents an organisational change such as "SoftwareDevelopment requires Netbeans". In one group the approval process was for the business executive to approve any requirements that the business user identifies. If approval is

granted, then the rule goes to the IT architect for further approval before becoming part of the ontological domain model.

An analysis of the results in
Figure 1 for frequency of communication indicates that the mean frequency of communication is statistically significantly ($p <= 0.05$) higher for the experimental group when CRESUS-T and email were used ($\mu=25$, $\sigma= 10.218$, $N=6$) compared to the case in the control group when only email is used ($\mu=20.83$, $\sigma=9.042$, $N=6$), paired $t(5) = 2.792$, $p=0.038$. The effect size was large ($\rho = 0.829$). This result demonstrates that CRESUS-T support tool increases communication among employees. This is because the design of the CRESUS-T support tool ensures that employees focus the communication on identifying requirements from the application domain and are supported by automatic machine translation in the generation of the IT infrastructure.

The employees were asked for their opinion on communication. Qualitative feedback from employees that used the support tool, CRESUS-T was useful for communicating especially around the IT infrastructure. Samples of comments provided include: -

- "Provided a new means for communication around IT infrastructure".
- "A simple communication protocol for organisational change".

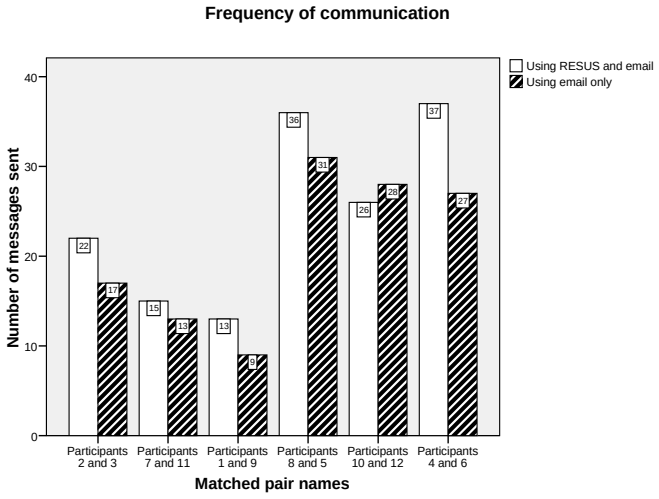

Figure 1 Frequency of communication

Prototype Supports Organisation Change indicates the degree to which employees that use CRESUS-T perceive that the prototype of the IT infrastructure comprising of web services supports the organisational change faithfully from strongly agree to strongly disagree based on simulation based communication tool online questionnaire. 66% of the experimental groups A and C that used CRESUS-T indicated that the prototype of the IT infrastructure comprising of web services supports the organisational change faithfully and 44% were neutral. This result is as expected because the CRESUS-T support tool incorporates a reference architecture that is embedded in the terms from the application domain, and deploys real web services and database on a glassfish server thus simulating an authentic representation of the IT infrastructure. This result was reinforced by comments from the employees such as: -

- "Simulation of the IT infrastructure was realistic and would support the organisational changes".

- "Good for modelling real world scenarios which illustrates to the other users (departments) a more complete workflow"

These comments are in contrast to comments from the control group. Based on observations of control group B and validated by the group, they spent a substantial amount of work in defining the IT infrastructure but very little time on the requirements of the business. They had two suggested solutions around different databases, but had not made a decision on which solution to go with. One of the employees of control group B made the following comment "Maybe the contributed [employees] should be told not to focus on specific technologies too much". The comment highlights an advantage of using CRESUS-T in that it allows the employees to focus on communication around the business and not on defining the IT infrastructure. One of the objectives of the experimental study was to generate the IT infrastructure. One could argue that this is unrealistic and not fair considering that CRESUS-T automatically generates the IT Infrastructure as part of the prototype. Further studies require employee's usage of the asynchronous support tools over a longer period of time such as one week. This would give a more realistic time for the control group to come up with a proper IT Infrastructure.

An observation of control group D and validated by the group was that they identified the software requirements list that was used in one part of the business process for the roll out of the IT software image. This list was used in a previous year. This also included the IT system that the data was stored in namely, Microsoft Excel. With regard to the observations of control group D, they did have a simple IT system however they did have an employee that worked on the business process for the roll out of the IT software image, in his role with the IT department. This presents a challenge with research of this nature that involves a realistic case study. Having an employee from the IT department in each group was an attempt to counter balance this situation. Overall, the results indicate that the CRESUS-T support tool helps create a realistic IT infrastructure that supports organisational change.

6. LIMITATIONS

This was the first experiment with the CRESUS-T support tool and so it was important to get early feedback and direction on further development and experimentation. As such the experiment took place at the National College of Ireland. One of the goals of the experiment was to demonstrate that the architecture has the possibility to scale up to the full expressivity of the controlled natural language and so for this experiment sentences were constrained to simple "noun verb noun".

Maturation was seen as a threat to the matched participants' research design used in this experiment. Maturation is where participants mature or change during the experiment. The experimenter attempted to select participants that would mature at the same rate. In group one and group two, participant 2 and participant 3 are both studying for a PhD. During the experiment, participant 2 completed the viva. Also in group one and group two, participant 7 and participant 11 both currently have an honours degree and are both completing a PhD. During the course of the experiment, participant 11 completed a viva. Survey items that capture potential maturation should be incorporated into a pre-test.

Metrics used for the participants' perception of attaining a shared representation of the IT systems that supports the organisational change were not suitable for a matching participant's research design. As each group is communicating collaboratively to make decisions about organisational changes and creating the IT system that supports those changes. Thus it stands to reason that there will be no significant differences in perception. Objective metrics instead of participants' perceptions should be used in further experimentation.

In group four, participant 6 is involved with the roll out of the IT environment and this extra knowledge would have biased the results in particular when identifying an IT system and identifying organisational change solutions, issues and constraints. Survey items that capture the participants' role in relation to the scenario should be incorporated into a pre-test and controlled for in future experiments.

A weakness in the experimental design was that the controlled experiment was conducted in a one hour setting which would not be representative of actual communication behaviours among the employees. Further experimentation should be conducted over a longer period of time and incorporated into their actual job so that it is representative of their actual communication patterns. The limitations will be addressed in further experimentation.

7. CONCLUSION

The goal of requirements elicitation is to reach a shared understanding between all parties involved in the communication process which often involves an increased amount of communication effort to overcome the gap in communicating the requirements desired in a semantically consistent and understandable manner and then reflecting the potential impact of those requirements on the IT infrastructure.

An initial study conducted among ten business executives in higher education indicates a desire by the majority of this small group for a tool that allows them to communicate organisational changes using natural language where these changes are automatically translated into the IT infrastructure that supports a business process.

Building on this research, CRESUS-T was implemented as a collaborative requirements elicitation support tool that allows stakeholders to communicate requirements from an ontological domain model, validate the requirements, and create a model of the IT infrastructure that supports those requirements. The tool was evaluated at the National College of Ireland.

Results provide confidence that the tool significantly increases the frequency of communication which is a predictor of reaching a shared understanding between all stakeholders during requirements elicitation. Qualitative feedback from participants that use CRESUS-T indicates that the tool facilitates collaboration and communication around the IT Infrastructure with comments such as "good for storing knowledge and facilitating collaboration" and "Provided a new means for communication around the IT Infrastructure".

8. FUTURE WORK

Web services can be orchestrated into an executable business process using the business process execution language. As such future work will involve investigating CRESUS-T's role in collaborative requirements elicitation for the creation of the IT infrastructure that supports a business process.

ACKNOWLEDGEMENTS

The research was partially supported by the SFI ADAPT Research Centre (Grant 13/RC/2106).

REFERENCES

[1] Bourque, P. & Fairley, R. E. eds. (2014). Guide to the Software Engineering Body of Knowledge, Version 3.0. IEEE Computer Society, 2014. September 11, 2014.

[2] Coughlan, J., & Macredie, R. D. (2002). Effective communication in requirements elicitation: A comparison of methodologies. Requirements Engineering, 7(2), 47–60.

[3] International Institute of Business Analysis. (2009). A guide to the business analysis body of knowledge (Babok Guide) Version 2.0. International Institute of Business Analysis.

[4] Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 35–46).

[5] O'Loughlin, E. (2010). An Introduction to Business Systems Analysis: Problem Solving Techniques and Strategies. Dublin, Ireland: The Liffey Press.

[6] Sutcliffe, A., (1997). A technique combination approach to requirements engineering. In Proceedings of the Third IEEE International Symposium on Requirements Engineering, (pp. 65–74).

[7] Saiedian H, Dale R. Requirements engineering: making the connection between the software developer and the customer. Inform Software Tech 2000; 42 (6): 419-428.

[8] Damian, D. E. & Zowghi, D. (2002). The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In IEEE Joint International Conference on Requirements Engineering, 2002. Proceedings (pp. 319–328). doi:10.1109/ICRE.2002.1048545.

[9] Walia, G. S. & Carver, J. C. (2009). A systematic literature review to identify and classify software requirement errors. Information and Software Technology, 51(7), 1087–1109. doi:10.1016/j.infsof.2009.01.004.

[10] Stynes, P., Conlan, O., O'Sullivan, D., (2008). Towards a Simulation-based Communication Tool to Support Semantic Business Process Management. IN: Proceedings of the Fourth International Workshop on Semantic Business Process Management in Proceedings of Workshops held at the Fifth European Semantic Web Conference, (ESWC08), 2nd June, Tenerife, Canary Islands, Spain.

[11] Robertson, S. & Robertson, J. (2012). Mastering the Requirements Process: Getting Requirements Right (3 editions.). Upper Saddle River, NJ: Addison-Wesley Professional.

[12] Rogers, E.M., and Kincaid, D.L. Communication Networks. New York: Free Press, 1981.

[13] Lind, M. R., & Zmud, R. W. (1991). The Influence of a Convergence in Understanding between Technology Providers and Users on Information Technology Innovativeness. Organization Science, 2(2), pp. 195-217.

[14] Johnson, A.M., & Lederer, A.L., (2005). The effect of communication frequency and channel richness on the convergence between chief executive and chief information officers. Journal of Management Information Systems / Fall 2005, Vol. 22, No. 2, pp. 227-252.

[15] Preston, D. S., Karahanna, E., & Rowe, F., (2006). Development of shared understanding between the chief information officer and top management team in U.S. and French Organizations: A cross-cultural comparison. IEEE Transactions on Engineering Management, 53(2), 191-206. doi: 10.1109/TEM.2006.872244.

[16] Van Lamsweerde, A., (2009). Requirements engineering: from system goals to UML models to software specifications (1st edition). Chichester, West Sussex, John Wiley & Sons.

[17] Allemang, D. (2008). Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL (1 edition). Amsterdam ; Boston: Morgan Kaufmann.

[18] Castañeda, V. Ballejos, L. Caliusco, M. L. & Galli, M. R. (2010). The use of ontologies in requirements engineering. Global Journal of Researches in Engineering, 10(6).

[19] Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199–220.

[20] Guarino, N. (1998). Formal Ontology and Information Systems (pp. 3–15). Presented at the Proceedings of FOIS'98, Trento, Italy: Amsterdam, IOS Press.

[21] Neches, R. Fikes, R. E. Finin, T. Gruber, T. Patil, R. Senator, T. & Swartout, W. R. (1991). Enabling technology for knowledge sharing. AI Magazine, 12(3), 36.

[22] Musen, M. A. (1992). Dimensions of knowledge sharing and reuse. Computers and Biomedical Research, 25(5), 435–467.

[23] Greenspan, S., (1984). Requirements modelling: a knowledge representation approach to software requirements definition. Technical Report No. CSRG-155. University of Toronto, Toronto, Canada.

[24] Greenspan, S. Mylopoulos, J. & Borgida, A. (1994). On formal requirements modeling languages: RML revisited. In Proceedings of the 16th international conference on Software engineering (pp. 135–147). IEEE Computer Society Press.

[25] Mylopoulos, J. Borgida, A. Jarke, M. & Koubarakis, M. (1990). Telos: Representing Knowledge About Information Systems. ACM Trans. Inf. Syst. 8(4), 325–362. doi:10.1145/102675.102676.

[26] Dardenne, A. van Lamsweerde, A. & Fickas, S. (1993). Goal-directed requirements acquisition. Science of Computer Programming, 20(1-2), 3–50.

[27] Yu, E., Mylopoulos, J., (1994). Understanding" why" in software process modelling, analysis, and design. In Proceedings of the 16th International Conference on Software Engineering (ICSE-16). (pp 159-168).

[28] Berners-Lee, T. Hendler, J. & Lassila, O. (2001). The Semantic Web. Scientific American, 284, 34–43. doi:10.1038/scientific American 0501-34.

[29] Bray, T. Paoli, J. Sperberg-McQueen, C. M. Maler, E. & Yergeau, F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition).

[30] RDF - Semantic Web Standards. (2014). Available online at http://www.w3.org/RDF/.

[31] OWL 2 Web Ontology Language Document Overview (Second Edition). (2012). Available online at http://www.w3.org/TR/owl2-overview/.

[32] Dobson, G. & Sawyer, P. (2006). Revisiting ontology-based requirements engineering in the age of the semantic web. In Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs.

[33] Li, G. Jin, Z. Xu, Y. & Lu, Y. (2011). An Engineerable Ontology Based Approach for Requirements Elicitation in Process Centered Problem Domain. In H. Xiong & W. B. Lee (Eds.), Knowledge Science, Engineering and Management (pp. 208–220). Springer Berlin Heidelberg.

[34] Fonseca, F., (2007). The double role of ontologies in information science research. Journal of the Association for Information Science and Technology. Wiley Online Library.

[35] Guarino, N. (1998). Formal Ontology and Information Systems (pp. 3–15). Presented at the Proceedings of FOIS'98, Trento, Italy: Amsterdam, IOS Press.

[36] Yildiz, B. & Miksch, S. (2007). Ontology-driven information systems: Challenges and requirements. In International Conference on Semantic Web and Digital Libraries. Indian Statistical Institute Platinum Jubilee Conference Series (2007) 35–44.

[37] Abbott, R. J. (1983). Program Design by Informal English Descriptions. Commun. ACM, 26(11), 882–894. https://doi.org/10.1145/182.358441.

[38] Berzins, V. Martell, C. Luqi, & Adams, P. (2008). Innovations in Natural Language Document Processing for Requirements Engineering. In B. Paech & C. Martell (Eds.), Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs (pp. 125–146). Springer Berlin Heidelberg.

[39] Booch, G. (1986). Object-oriented development. Software Engineering, IEEE Transactions on, (2), 211–221.

[40] Ceri, S., Fraternali, P. (1997). Designing database applications with objects and rules: the IDEA Methodology. Addison-Wesley.

[41] Church, K., & Patil, R. (1982). Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table. Comput. Linguist., 8(3–4), 139–149.

[42] Fuchs, N. E., Kaljurand, K., & Schneider, G. (2006). Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In FLAIRS Conference (Vol. 12, pp. 664–669).

[43] Kaljurand, K. & Fuchs, N. E. (2007). Verbalizing OWL in Attempto Controlled English. In OWLED (Vol. 258).

[44] Kuhn, T. (2008). Acewiki: A natural and expressive semantic wiki. Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges, CEUR Workshop Proceedings.

[45] Fuchs, N. E. Höfler, S. Kaljurand, K. Rinaldi, F. & Schneider, G. (2005). Attempto controlled english: A knowledge representation language readable by humans and machines. In Reasoning Web (pp. 213–250). Springer.

[46] Ameller, D. Ayala, C. Cabot, J. & Franch, X. (2013). Non-functional Requirements in Architectural Decision Making. IEEE Software, 30(2), 61–67.

[47] Kruchten, P. Capilla, R. & Duñeas, J. C. (2009). The Decision View's Role in Software Architecture Practice. IEEE Software, 26(2), 36–42.

[48] Kazman, R. & Bass, L. (1994). Toward deriving software architectures from quality attributes. DTIC Document.

[49] Greenspan, S., & Feblowitz, M. (1993). Requirements engineering using the SOS paradigm. In , Proceedings of IEEE International Symposium on Requirements Engineering, 1993 (pp. 260–263)

.

[50] Chandrasekaran, S. Silver, G. Miller, J. Cardoso, J. & Sheth, A. (2002). Web service technologies and their synergy with simulation. IN: Proceedings of the 34th Winter Simulation Conference: exploring new frontiers (WSC 2002), 8-11, December, San Diego, California, USA. (pp. 606-615).

[51] Fuchs, N. E. Kaljurand, K. and Kuhn, T. (2008). Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Maluszynski, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors, Reasoning Web, Fourth International Summer School 2008, number 5224 in Lecture Notes in Computer Science, pages 104–124. Springer, 2008.

[52] Echo2 Web Framework Available from http://echo.nextapp.com/site/ [last accessed on 11th February, 2010.

[53] Kaljurand, K. (2013). APE (ACE Parser) v6.0. February 26, 2015.

[54] Protégé. Available from http://protege.stanford.edu/ [last accessed on 11th February, 2010].

[55] Jena – A semantic web framework for Java. Available from http://jena.sourceforge.net/ [last accessed on 11th February, 2010].

[56] Pellet: OWL 2 Reasoner for Java. Available from http://clarkparsia.com/pellet [last accessed on 11th February, 2010].

[57] Aho, A. Lam, M. Sethi, R. & Ullman, J. (2006). Compilers: Principles, Techniques, and Tools, 2/E. Prentice Hall.

[58] JDOM v1.1. (2007). Available online at http://www.jdom.org/dist/binary/archive/.

[59] XML Database, eXist. Available from http://exist.sourceforge.net/ [last accessed on 11th February, 2010].

[60] NetBeans IDE v6.8. (n.d.). February 26, 2015, Available online at https://netbeans.org/downloads/6.8/.

[61] Apache Ant v1.8.2. (2010). Available online at http://ant.apache.org/.

[62] Quality and Qualifications Ireland. (2012). National Framework of Qualifications (NFQ). March 9, 2015, Available online at http://www.qqi.ie/Pages/National-Framework-of-Qualifications-(NFQ).aspx.

[63] IBM - SPSS software - Ireland. (2015). Available online at http://www-01.ibm.com/software/ie/analytics/spss/.

AUTHORS

Dr Paul Stynes is Vice Dean of Academic Programmes and Research at the National College of Ireland. He completed a PhD in 2015 at Trinity College Dublin. His research interestsare in the area of Collaborative requirements elicitation and Intelligent Systems specifically semantic web, and ontologies.

Prof. Conlan is an internationally recognised research leader in User Modelling, Adaptation, Personalisation and Visualisation research with over 160 publications in those fields. Owen is a Fellow of Trinity College Dublin and leads the Personalisation research in the ADAPT Centre (www. adaptcentre.ie).

Prof. Declan O'Sullivan is Director of Research and Head of Discipline for Intelligent Systems at Trinity College Dublin. Declan was awarded a B.A. (Mod) in Computer Science from TCD in 1985, an M.Sc. in Computer Science from TCD in 1988, and a Ph.D. in Computer Science from TCD in 2006. Declan has substantial research experience in academia and industry, having worked for IONA Technologies and Broadcom Eireann Research.