Concept Maps: Theory, Methodology, Technology
Proc. of the Fifth Int. Conference on Concept Mapping
Valletta, Malta 2012

AUTOMATED CONCEPT MAP GENERATION FROM SERVICE-ORIENTED ARCHITECTURE ARTIFACTS

John W. Coffey, Thomas Reichherzer, Bernd Owsnick-Klewe, and Norman Wilde,
The University of West Florida, Pensacola, FL. USA
Email: {jcoffey | treichherzer | bowsnickiklewe | nwilde @uwf.edu}

Abstract. This article contains a description of the first phase of a two-phase project in computer program understanding for Service-oriented Architecture (SOA) composite applications. The first phase, which is described here, pertains to development of CARET, a prototype software tool that automates the process of generating concept maps that pertain to the structure of composite applications. CARET produces triples describing the relationships of entities contained in WSDL, BPEL, and XML Schema files. The triples can be imported into CmapTools, where they are automatically rendered into a concept map. The tool is programmatically extensible to capture any relationships of interest in such systems. In the second phase, to be reported in future work, the concept maps created in this fashion will be augmented with knowledge elicited from experts in the software system being modeled.

1 Introduction

Well-structured concept maps provide clear concise representations of conceptual knowledge. Although the vast majority of work on concept map creation pertains to those created by humans, some studies contain descriptions of machine-generated maps. This article describes work on the first part of a two-part process of automatically generating concept maps that identify and visualize relationships in complex artifacts of Service-oriented Architecture (SOA) composite applications and then augmenting these concept maps with domain knowledge of the software elicited from human experts. The end goal of the work is to foster understanding of SOA applications through these constructed knowledge models. This paper reports results on the first phase (automated concept map generation) of this process.

This work is motivated by the rapidly increasing use of Web Services and SOA to tie together heterogeneous networked software systems. The exceptional complexity of the documents that specify and orchestrate SOA applications makes these software systems difficult to understand. Separate XML specifications are used to describe Web service interfaces (WSDL), data types (XML Schema), and composite application processes (BPEL). Understanding how all the pieces fit together is a significant challenge for software maintainers. It is, however, feasible to extract {concept → linking phrase → concept} triples from these documents that explicitly summarize important relationships among them. These triples can be imported into a tool such as CmapTools (Cañas et al, 2004) in order to represent the relationships they convey as concept maps.

The remainder of this article contains a description of a system for the automated generation of concept maps from XML documents (WSDLs, XML Schemas, and BPELs) that describe composite applications executing in a Service-oriented Architecture. Following a review of relevant literature on the automated generation of concept maps and on conceptual models of composite applications residing in Service- oriented Architectures, the paper presents a description of CARET: Composite Application Reverse Engineering Tool. After a basic discussion of the way in which the tool is used, a case study pertaining to a medium-sized composite application is presented. The case study includes a description of a concept map that was automatically generated with CARET and populated with accompanying resources. The paper concludes with an analysis of the benefits and drawbacks of the approach, and a description of future work.

2 Automated Generation of Concept Maps

RDF and OWL are Semantic Web markup languages that are difficult for humans to understand. Eskridge and Hayes (2006) describe a plug-in to CmapTools named COE that allows the automatic generation of concept maps from RDF and OWL ontologies and OWL/RDF ontologies from concept maps. The goal of their work is to foster human understanding of these complex artifacts while retaining the formal properties of the descriptions. For this reason, the set of linking phrases is constrained to those in the OWL and RDF vocabularies. Graudina and Grundspenkis (2008) also described a tool that supported this task. Their contribution was identifying a general algorithm for the transformation of ontologies in any format into concept maps.

Richardson, Goertzel, and Fox (2006) have developed a tool named Relex that produces concept maps from text. The system they produced creates graphs of semantic primitives (Wierzbicka, 2006) extracted from syntactic dependencies. Relex requires domain-specific vocabularies, necessitating a lexicon of relevant terms. The work they performed was in the area of electronic theses and dissertations pertaining to computer science, which involves inherently large documents.

Work has also been done on the automated transformation of concept maps into other formalisms. For example, Gomez, Diaz-Agudo, and Gonzalez-Calero (2004) describe the transformation of concept maps into description logics. Leake, Maguitman, and Reichherzer (2004) describe the extraction of concepts from concept maps and the transformation of those concepts into queries that can be executed by a Web-based search engine. Cabral and Goncalves (2006) describe software that transforms concept maps into SCORM-compliant learning objects, and represented in another XML vocabulary.

Although not directly related to the automatic generation of concept maps, the idea of building understandings with “expert skeleton” concept maps put forth by Novak and Cañas (2004) is also relevant to the current work and briefly discussed here. Novak and Cañas state that expert skeleton maps provide a strong foundation for learning and that they create a context in which additional concepts and propositions may be added. This principle applies regardless of whether the concept maps are generated by humans or by automated means. Ideas in the work on expert skeleton maps are relevant to knowledge modeling for the understanding of SOA, the topic of the next section.

3 Knowledge Modeling for the Reverse Engineering of Software

Canfora, Di Penta, and Cerulo (2011) claim that "Radical innovations are needed to cope with new and emerging software development scenarios and new system architectures (p. 151)." They cite Chikofsky and Cross' (1990) seminal work which defined the problem of reverse engineering software as the creation of representations of the structure and function of software that are useful to humans from available software "artifacts." The word artifact was meant to include internal and external documentation of source and target code, as well as the code itself.

As suggested by Canfora, Di Penta, and Cerulo, a primary goal of reverse engineering is the creation of architectural models of software that support specific maintenance tasks. The prevailing approach to this end relies upon the evaluation of source code and documentation. The literature includes descriptions of a number of tools and methods that serve this goal. What is less clear in the literature is how to perform this task on the complex artifacts comprising SOA applications and how to incorporate less tangible conceptual knowledge that might provide significant insights into the software, including design rationale or undocumented features.

A substantial body of work exists on the idea of knowledge capture, preservation and sharing through the creation of knowledge models based upon the elicited conceptual knowledge of experts (Ford, Cañas, and Coffey, 1993; Cañas et al, 1996; Coffey, et. al., 2003). One approach involves interactive elicitation of concept maps (Novak and Gowin, 1984) which are concise representations of propositional knowledge. Propositions are triples involving two concepts and a linking phrase that describes the relationship between the concepts:

```
<concept> <linkingPhrase> <concept>
```

For instance, *“automobiles are powered by engines”* is a proposition comprised of the concepts “automobiles” and “engines” and the linking phrase “are powered by.” Hierarchically structured groups of concept maps can be populated with other electronic resources to form conceptual or knowledge models (eg: Coffey, Hoffman, and Novak, 2010) pertaining to some body of knowledge.

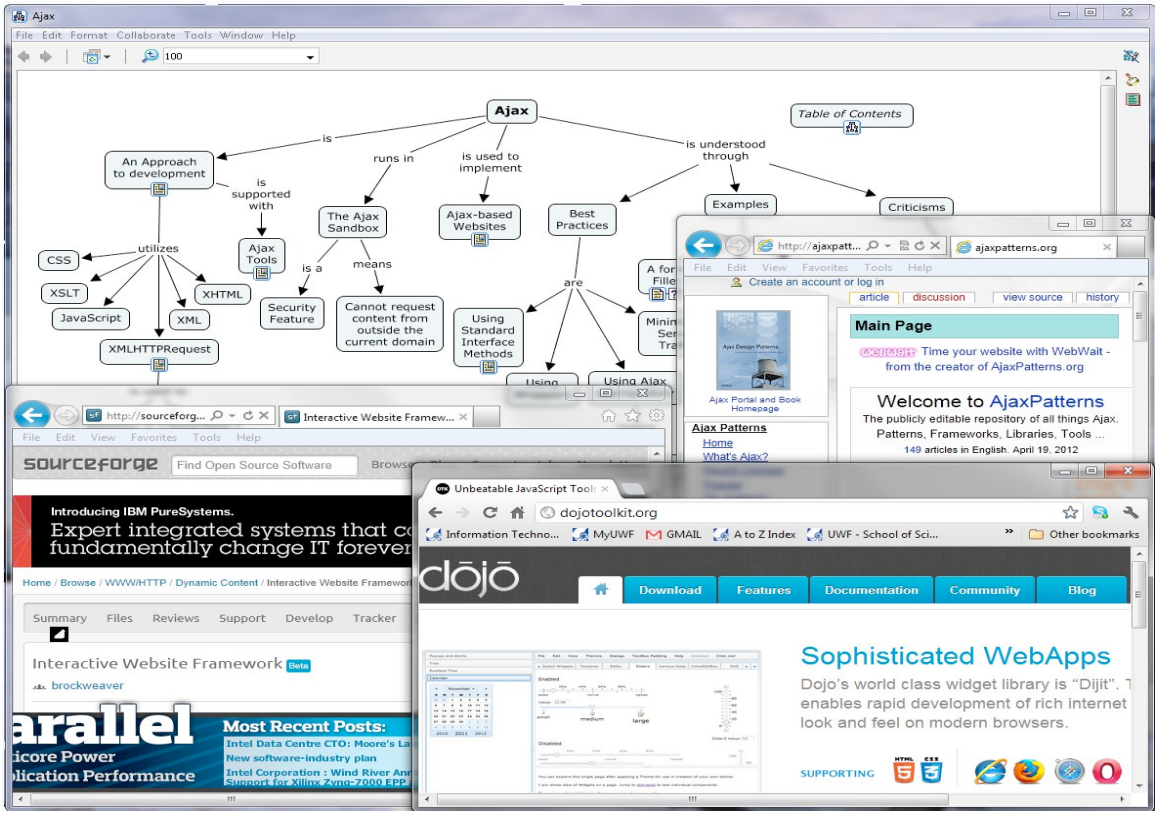

Figure 1. A Knowledge Model with concept map and related resources.

Figure 1 contains an illustration of components of a Web-based knowledge model that pertains to AJAX (Asynchronous JavaScript and XML, a group of technologies used to create interactive websites). The top-most window is a concept map that has been enhanced with links indicated by the icons below the concepts (the rounded rectangles in the maps) in some of the nodes. The icons provide access to pull-down menus that afford navigation to associated electronic resources. Figure 1 might arise from a scenario in which the user first navigated into the AJAX concept map from a more general concept map, perhaps on the subject of the JavaScript programming language. From the AJAX map, the user clicked on a link pertaining to AJAX tools to access the dojo Toolkit website, an "Interactive Website Framework" form Source Forge, and a page that pertains to AJAX design patterns. Knowledge models such as this one have been created in numerous knowledge domains (Coffey, et al. (2003); Coffey, Hoffman, & Novak, (2010)).

A fundamental purpose of CARET is to automate the creation of concept maps that provide a basic structure for the creation of knowledge models pertaining to SOA composite applications. CARET evaluates named entities described in BPEL, WSDL, and XML Schema files for a specific SOA composite application and provides input for a program that creates concept maps from triples. CARET produces all information needed to produce a hierarchically-structured "skeleton knowledge model," a structured representation of several skeleton concept maps. These concept maps may be elaborated with elicited expert knowledge, and augmented with additional electronic resources including the original WSDL, BPEL and XML Schema files that were abstracted to form the maps. The elicited expert knowledge helps to address the issue of identifying design rationale and other important but non-obvious knowledge pertaining to the software.

4 Overview of Service-Oriented Architectures and Maintenance Problems

Service-oriented Architectures offer the prospect of creating multi-platform software out of heterogeneous services executing on the Internet. A composite application is typically comprised of several service components that may be internal or external to the organization constructing it. Each service component has interface descriptions that facilitate interoperability among the set of services in a SOA composite application. The descriptions follow a set of standard languages for building and deploying Web Services including Web Services Description Language (WSDL) for service interfaces, XML Schema Definition (XSD) for service data

types, SOAP for message formatting, HTTP for message exchange, and Business Process Execution Language (BPEL) for service orchestration code. Together, service descriptions, code, and execution hardware form a complex system of systems that is difficult to comprehend.

Any maintenance task on such a composite system requires an in-depth understanding of its service components, the operations they perform, and the data they exchange. This includes knowledge of the service descriptions and the dependencies among the services within the composite. CARETs goal is to provide maintainers with a high-level description of service components, their operations and dependencies for a selected composite application.

5 CARET

CARET, Composite Application Reverse Engineering Tool, is a tool for the semi-automated creation of knowledge models of composite applications. CARET is based upon a SAX parser for XML files that allows it to react to individual elements and attributes that are of interest to the parser, without having to build a DOM tree in RAM. This feature ensures that the processing of large files, typical of those encountered in SOA systems, will be feasible.

When the current version of CARET executes, it processes the contents of a folder that contains files describing composite applications: the BPELs, WSDLs, and XML Schemas. These files will have been gathered into the target folder by other software that is currently in development and that preserves the locations of the original files. The user can select to process an entire composite application comprised of BPELs, WSDLs, and XML Schemas, the BPEL and WSDL components only, or WSDL and XML Schema components only. This feature enables the user to generate a more focused or a comprehensive view depending upon the kinds of information that is currently of interest. Additionally, the user can choose to create a single global concept map of the entire application, or to generate separate descriptions corresponding to individual BPELs, WSDLs, or XSDs that will ultimately become more detailed parts of the conceptual model. Figure 2 contains a graphic of the current interface to CARET.

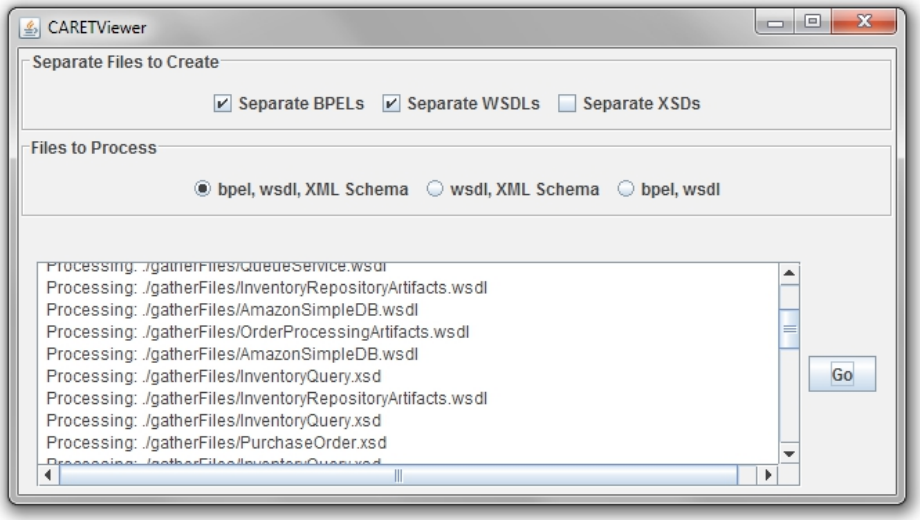

Figure 2. The interface to CARET.

When the program starts it analyzes the available BPEL or WSDL files, depending on the choice made in the “Files to Process” portion of the dialog. If the user has chosen to analyze BPEL files, dependencies based upon receive-reply and invoke operations are noted. These dependencies enable the system to infer the main and subordinate processes, to record the individual operations that are invoked, received, or replied to, and to link together references to them in BPELs and WSDLs. This set is recorded so that operations described in the WSDLs that are not used in the current composite are excluded from the current conceptual model. If the user has chosen to analyze WSDLs and XML Schemas only, the system makes note of the operations in the WSDLs and their links to local or XML Schema-based data definitions. In this case, all operations defined in the WSDLs are documented.

As the BPEL pre-processing proceeds, all referenced WSDL files are dynamically added to a list of files to be processed. As they are processed, following the scanning of the BPELs, their internal and external data declarations are analyzed. Any XML Schema files that are referenced in the WSDLs are added to a dynamically generated list of files to process. CARET is comprised of three SAX parsers, one for BPEL files, one for WSDL files, and one for XML Schemas. This division creates a context that allows for localization of operations specific to one type of file, even in the event that the operation is processing an element name that means different things in different contexts. It also allows for relatively simple programmatic extension of the parsers as other features of interest are identified for inclusion in the models to be generated. The output of CARET is a set of triples typically of the form illustrated in Table 1.

Concept	***Link***	***Concept***
`<filename>`	`<linkingPhrase>`	`<object>`
`<object>`	`<linkingPhrase>`	`<filename>`

Table 1: CARET Triple Representation**.**

In the triple representation, `<object>` might be another file, an operation name, a data element, or other item of interest. The `<linkingPhrase>` includes items such `as “is defined in,” “has invoke operation,” “has receive operation,”` etc. The triples are tab-delimited which allows them to be imported into CmapTools which is used to create the conceptual model. For example the triple:

```
SMSFamilyModeling.wsdl→defines operation→BoundryUpdate
```

is of the first format indicating that the file named `SMSFamilyModeling.wsdl` defines an operation named `BoundaryUpdate`. In this case, the object is a method rather than data or something else.

CARET produces as output one or more files of triples. If no separate file output is specified in the interface, a single file of triples relating features of some combination of the BPELs, WSDLs and XML Schemas (as specified in the “Files to Process” part of the CARET dialog) is created. If the user has additionally selected to output individual WSDL files or other files, separate files of triples are produced that allow for the creation of a hierarchical model with a single concept map at the root of the hierarchy, and additional, more detailed concept maps that can be linked to it.

The single item at the top of the hierarchy is a global view and additional items at lower levels comprise additional details of the WSDLs and XSDs in the composite application. All of the components the user specifies are available for import into CmapTools. The user can pick and choose which to include in the skeleton knowledge model. Once one or more triple files have been loaded into CmapTools, many capabilities are available that enable the user to tailor the model to current needs.

The individual files of triples are manually imported into CmapTools where they can be arranged in any manner that the user wishes. An “auto-layout” feature is available that creates alternative arrangements of the graphical depictions. The model is editable and can be enhanced with other electronic resources or with knowledge of the artifacts that participants in the process might know. A mature body of literature exists regarding the process of eliciting conceptual knowledge from an expert with concept maps. CmapTools provides a simple-to-use mechanism to create links among the graphical depictions of program structure, and the actual files to which the nodes in the map relate. The files that have been analyzed can be associated with their representations in the conceptual model by a simple drag-and-drop process to create a comprehensive knowledge model.

6 A Case Study

A case study in the use of CARET is described in this section. It is based upon a "Web Auto Parts" composite application comprised of two BPEL files, five WSDL files, and two XML Schema files (Wilde, Coffey, Reichherzer and White, 2012). Web Auto Parts is a hypothetical startup online automobile parts store which uses Business Process Execution Language to orchestrate services provided by commercial vendors such as Amazon. In this study, the full BPEL composite application is recovered. The Web Auto Parts application is comprised of the files depicted in Table 2.

`OrderProcessing.bpel`	`OrderProcessingArtifacts.wsdl`
`PurchaseOrder.xsd`	`QueueService.wsdl`
`TaxDataBasic5.wsdl`	`InventoryRepository.bpel`
`InventoryRepositoryArtifacts.wsdl`	`InventoryQuery.xsd`
`AmazonSimpleDB.wsdl`	

Table 2: Resource files of the Web Auto Parts composite application.

These files comprise a composite application that allows the user to request a check on availability of a selection of items with Amazon, to form an order, compute tax and shipping charges for the order, and to receive a reply regarding success or failure to complete the transaction. The composite application is comprised of a combination of both internal and external services. No global view of the composite application was available. The triples for the concept map in Figure 3 were generated automatically by the process described in the previous section.

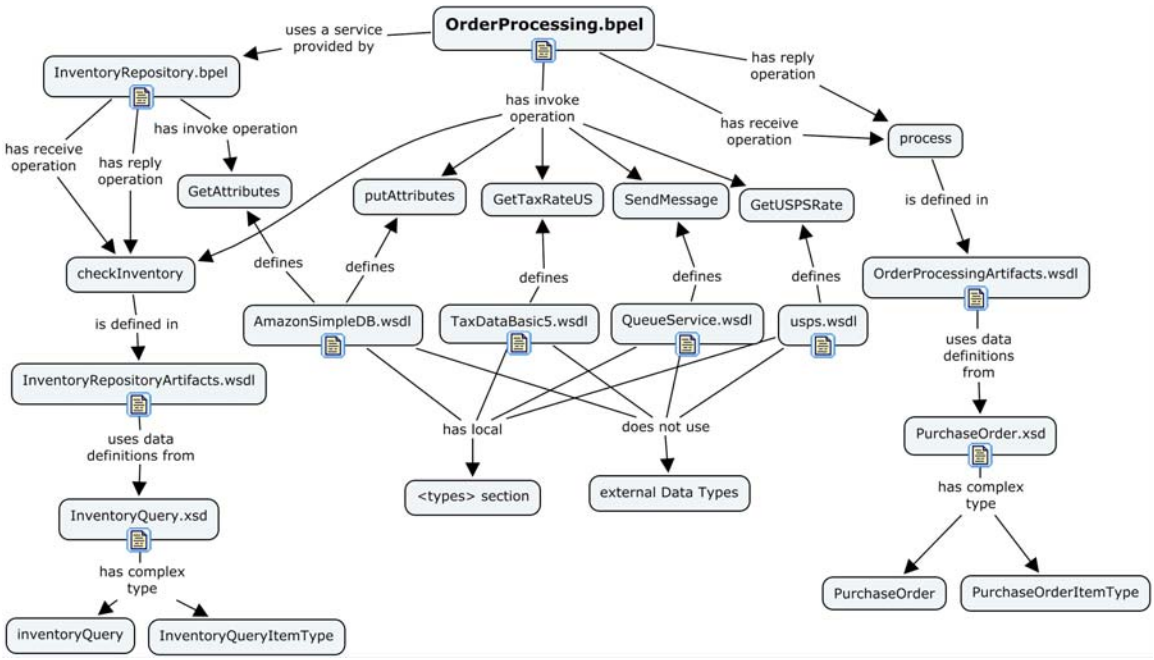

Figure 3. A Conceptual Model of the Web Auto Parts Composite Application.

As can be seen in Figure 3, the `OrderProcessing` BPEL (the main method in the composite application) has an invoke operation on the `CheckInventory` service and the `InventoryRepository` BPEL has `receive` and `reply` operations on the same service. Making explicit caller-callee relationships such as this one is an important abstraction afforded by CARET. It can also be readily seen that the `OrderProcessing` composite application utilizes four other services, `putAttributes`, `GetTaxRateUS`, `SendMessage`, and `GetUSPSRate`. The WSDLs in which these services are defined

and the locations of their data structures are all made evident by the generated concept map. Also, as can be seen in Figure 3, the actual BPEL, WSDL, and XSD files have been made available as linked documents (indicated by the icons under the appropriate nodes in the concept map).

In the current case study, a total of 43 operations (individual Web Services) were made available through the five included WSDLs. Of that total, seven services were utilized in the Web Auto Parts Composite Application. CARET filtered out references to irrelevant services - those that were not utilized in the current composite application. The seven services that were invoked may be viewed across the middle of Figure 3. If the BPEL utilized different services, CARET would generate a different conceptual model for that composite application, even though it was based upon the same WSDLs and XML Schemas.

7 Conclusions

This article contains a description of CARET, a prototype tool that detects items of interest to maintainers who are trying to understand the structure of SOA-based composite applications. CARET is used to process artifacts of BPEL and WSDL-based composite applications to produce triples that can be imported into CmapTools. CmapTools automatically renders them in one or more concept maps. The concept maps can be ordered hierarchically from more general to more detailed to form what has been called a knowledge model of the software system. These knowledge models may be augmented with the original files that were processed by CARET, any other documentation pertaining to the system, and additional conceptual knowledge elicited from people who are knowledgeable in the workings of the system.

One of the compelling ideas in this approach is the capability to provide a minimal, comprehensible overview of a complex system using a set of machine representations that describe it. The approach of combining structural knowledge of a composite application that can be generated automatically from the processing of input that CARET performs with expert human knowledge of the system provides greater support for system understanding than can be garnered from standard reverse-engineering tools. The modeling of heterogeneous artifacts is supported in a way that has been shown to be intuitive to people trying to answer questions pertaining to a domain of knowledge (Carnot et al, 2001).

Future work includes empirical exploration of additional conceptual relationships that are of value to maintainers of SOA-based systems. These relationships can be captured by extending the SAX parser upon which the system is based. Encoding a characterization of individual triples and storage in a database will afford the capability to construct varying views of the various components of a composite application dynamically. Future work will also address the benefits of augmenting the knowledge model with elicited expert knowledge.

8 Acknowledgement

Work described in this paper was partially supported by the University of West Florida Foundation under the Nystul Eminent Scholar Endowment.

9 References

Cabral, A., & Goncalves, L. (2006). SCORM Standard Compliant Learning Objects Generated from Content Organized with Concept Maps. In: A. J. Cañas & J. D. Novak (Eds.), *Concept Maps: Theory, Methodology, Technology. Proceedings of the Second International Conference on Concept Mapping*. San José, Costa Rica: Universidad de Costa Rica, pp. 519-526.

Cañas, A.J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gómez, G., Arroyo, M., Carvajal, R. (2004). CmapTools: A Knowledge Modeling and Sharing Environment. In A. J. Cañas, J. D. Novak & F. M. González (Eds.), *Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping*, Universidad Pública de Navarra: Pamplona, Spain. p. 125-133.

Cañas, A. J., Ford, K.M., Brennan, J., Reichherzer, T., Hayes, P., & Suri, N. (1996). *An Environment for the Construction and Sharing of Knowledge*, Proceedings of the Ninth Florida Artificial Intelligence Research Symposium, Key West, FL (May 1996).

Canfora, G., Di Penta, M., & Cerulo L. (2011). Achievements and Challenges in Software Reverse Engineering Communications of the ACM 54(4) pp. 142-151. doi :10.1145/1924421.1924451

Carnot, M. J., Dunn, B., and Cañas, A.J. (2001). Concept Map-based vs. Web Page-based Interfaces in Search and Browsing, ICTE 2001: International Conference on Technology and Education, Tallahassee, FL (May). (extended version available as: http://www.ihmc.us/users/acanas/Publications/ CMapsVSWebPagesExp1/CMapsVSWebPagesExp1.htm)

Chikofsky, E., & Cross, J.I. (1990). Reverse engineering and design recovery: A taxonomy. IEEE Software 7(1) pp. 13–17.

Coffey, J.W., Cañas, A.J., Reichherzer T., Hill G., Suri N., Carff, R., Mitrovich, T, & Eberle, D. (2003). Knowledge Modeling and the Creation of El-Tech: A Performance Support and Training System for Electronic Technicians. Expert Systems with Applications. 25(4). pp. 483- 492.

Coffey, J.W., Hoffman, R.R., & Novak, J.D. (2010). Applications of Concept Maps to Web design and Web work. In R. W. Proctor & K.-P. L. Vu (Eds.), Handbook of Human Factors in Web Design, 2nd Edition. pp. 211 – 230, Mahwah, NJ: Erlbaum

Eskridge, T., & Hayes, P. (2006). Formalizing the Informal: A Confluence of the Semantic Web and Concept Maps. In: A. J. Cañas & J. D. Novak (Eds.), *Concept Maps: Theory, Methodology, Technology. Proceedings of the Second International Conference on Concept Mapping*. San José, Costa Rica: Universidad de Costa Rica, pp. 247-254.

Ford, K.M., Cañas, A.J., & Coffey, J.W. (1993). Participatory Explanation. Proceedings of the Sixth Florida AI Research Symposium (FLAIRS '93), Ft. Lauderdale, FL, April, 1993. pp. 111-115.

Graudina, V., & Grundspenkis, J. (2008). Concept Map Generation from OWL Ontologies. In A. J. Cañas, P. Reiska, M. Åhlberg & J. D. Novak (Eds.), *Concept Mapping: Connecting Educators, Proceedings of the Third International Conference on Concept Mapping*, Tallinn, Estonia & Helsinki, Finland: University of Tallinn, pp. 173-180.

Gomez-Gauchia, D-A., Diaz-Agudo, B., & Gonzalez-Calero, P. (2004). Two-Layered Approach to Knowledge Representation Using Conceptual Maps and Description Logics. In A J. Cañas, J. D. Novak & F. M. González, (Eds.), *Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping*, Universidad Pública de Navarra: Pamplona, Spain, pp. 281–288.

Leake, D., Maguitman, A., and Reichherzer, T. (2004). "Googling" from a Concept Map: Towards Automatic Concept Map-based Query Formation. In A. J. Cañas, J. D. Novak, & F. M. González (Eds): *Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping*, Universidad Pública de Navarra: Pamplona, Spain, pp. 409-416.

Novak J.D., and Cañas, A.J. (2004). Building on New Constructivist Ideas and CmapTools to Create a New Model of Education. In A. J. Cañas, J. D. Novak, & F. M. González (Eds.), *Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping*, Universidad Pública de Navarra: Pamplona, Spain, pp. 469-476.

Novak J.D., & Gowin, D.B. (1984). Learning how to learn. New York: Cambridge University Press.

Richardson, R., Goertzel, B., and Fox, E. (2006). Automatic Creation and Translation of Concept Maps for Computer Science-related Theses and Dissertations. In: A. J. Cañas & J. D. Novak (Eds.), *Concept Maps: Theory, Methodology, Technology. Proceedings of the Second International Conference on Concept Mapping*. San José, Costa Rica: Universidad de Costa Rica, pp. 32-35.

Wierzbicka, A. (2006). Semantics, Primes, and Universals. Oxford, UK, Oxford University Press.

Wilde, N., Coffey, N.W., Reichherzer, T., White, L. (2012). Open SOALab: Case Study Artifacts for SOA Research and Education. to appear, 4th International Workshop on Principles of Engineering Service-Oriented Systems, PESOS 2012, Zurich, Switzerland, June, 2012.