A Roadmap on Integrating Applications and Data on the Web*

Rafael Corchuelo
Universidad de Sevilla
ETSI Informática
corchu@us.es

José L. Arjona
Universidad de Huelva
EU Politécnica
jose.arjona@dti.uhu.es

David Ruiz
Universidad de Sevilla
ETSI Informática
druiz@us.es

José L. Álvarez†
Universidad de Huelva
EU Politécnica
alvarez@dti.uhu.es

Abstract

There is steady shift towards integrating business applications to support and optimise business processes. Recently, the Service Oriented Architecture and the Semantic Web initiatives have provided new paradigms and technologies that help integrate applications and information. However, according to recent industrial reports, the cost of a typical integration project is still far from negligible. In this article, we report on a roadmap regarding application and information integration that is intended to guide future research efforts in this area and to produce a number of engineering tools that shall help reduce integration costs.

1 Introduction

The computing infrastructure of a company that has been running for a few years typically includes several heterogeneous, loosely coupled applications. Most companies have realised that integrating them or the data they manage is very valuable to support business processes. In the beginning, the integration was usually ad-hoc; however, as the number of applications to integrate increased, this soon proved not sustainable, which motivated many researchers to work on principled approaches to engineer integration.

The research work on integration can be broadly classified into Application Integration and Information Integration. The former approaches are operative since they model integration solutions as message workflows amongst several integration processes, i.e., the designer is responsible for devising and orchestrating the flow [25]; the latter, on the contrary, are declarative since they model solutions as data schemata and allow to transform queries into the appropriate message workflows automatically [24]. Note, however, that both kinds of solutions are complementary, and that real-world integration problems usually benefit from techniques that come from both worlds. Wrappers are used in both approaches since they help endow applications with specific-purpose programming interfaces by means of which they can be integrated. Roughly speaking, a wrapper helps instruct an application to perform an action or to answer a query in cases in which it does not support this functionality natively or does not deliver it using the appropriate technology.

Our initial hypothesis is that more and more companies shall rely on an increasing number of such automated business processes, which shall require more and more applications to be integrated to support and to optimise them. We think that this hypothesis is sensible on account of recent research reports by CIO Magazine [34], Gartner [17] and DataMonitor [10], according to which most IT companies are worried about integration, which shall be the driving force behind most IT large projects

*Partially funded by "Plan Nacional de I+D+I" (Exp. TIN2007-64119 and TIN2008-04718-E) and "Órden de Incentivos de la Junta de Andalucía" (Exp. P07-TIC-02602 and P08-TIC-4100). Part of the funds come from the European Regional Development Fund (FEDER).

†Additional authors: Rafael Z. Frantz, from UNIJUÍ (Brazil), Carlos Molina, from Newcastle University (UK), Inma Hernández, Carlos R. Osuna, Antonia M. Reina, Hassan A. Sleiman, from the University of Sevilla; Iñaki Fernández y Patricia Jiménez, from the University of Huelva.

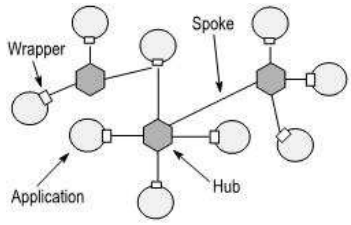

Figure 1: Structure of a typical application integration solution.

in the forthcoming ten years. Unfortunately, even though the technologies provided by the Service Oriented Architecture and the Semantic Web initiatives are helping cut integration costs down, recent reports [6] [54] highlight that integration costs are still from 5 to 20 times higher than developing new functionality, chiefly when web applications and RDF data are involved.

In Section 2, we survey current research efforts regarding application and information integration, with an emphasis on web applications that do not provide a programmatic interface; in Section 3, we report on a research roadmap that builds on a number goals and a justification for each of them; in Section 4 we draw our main conclusions; the article finishes with a listing of references to the literature.

2 State of the Art

In this section, we provide an overall picture of the state of the art regarding application integration, cf. Section 2.1, information integration, cf. 2.2, and wrapping, cf. 2.3.

2.1 Application Integration

The idea behind Application Integration is to devise a workflow of messages that allows to co-ordinate several applications exogenously so that they cooperate to keep their data in synchrony or to support a new piece of functionality.

One of the most successful approaches to application integration is the Hub&Spoke integration pattern [25]. Simply put, hubs provide the brains to an integration solution, since they implement the logic required so that several applications can co-operate exogenously; spokes, on the contrary, are communication channels by means of which messages are transferred from a hub to an application's wrapper and vice versa, or from a hub to another hub. The set of hubs a company runs is commonly referred to as their business bus, cf. Figure 1, and the technologies and techniques involved as Enterprise Application Integration.

The Hub&Spoke approach is commonly implemented using so-called Process Support Systems [21], which is a term that includes both conventional workflow systems [3] and recent orchestrators [32], e.g., BizTalk, Open ESB, Camel, Mule or Spring Integration. The Service Oriented Architecture initiative [42] has been a leap forward regarding application integration. It is not surprising, then, that most Process Support Systems are converging into Enterprise Service Buses that are based on recommendations like WSDL to define interfaces and bindings, SOAP to support message exchanges, or BPEL to support complex integration processes [42]. The role of traders is also important in this field, since they allow to search for the services that best support an integration process [27] [28]. Note that these technologies ease the integration of applications as long as they provide a programmatic interface or the data they manage can be accessed using files, databases or other data-oriented interfaces for which there are a kind of wrappers that are known as binding components. Applications that provide a user interface only, which is commonly the case of end-user web applications, are much more difficult to integrate since they require wrappers that emulate the interactions of a person to extract information from them.

Current Enterprise Service Buses use a database to store messages that come from wrappers or hubs until all of the correlated messages needed to start an integration process are available. The array of tasks that an integration process may execute includes tasks to receive messages, to copy, to manipulate, to

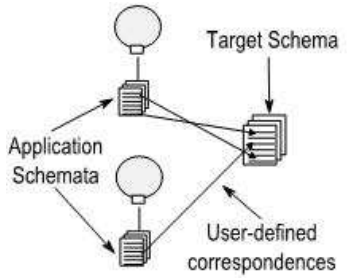

Figure 2: Structure of a typical information integration solution.

route and to deliver them. Without an exception, the run-time system is based on a per-process allocation policy, which means that a thread is allocated to each integration process that is instantiated, and that it is not relinquished until the resulting outgoing messages are delivered.

2.2 Information Integration

The idea behind Information Integration is to have a target schema that integrates the data managed independently by several applications, so that they can be seen as if they were a large database, cf. Figure 2 [24]. Wrappers allow to have access to an application's data, and there are components that map user queries over the target schema into appropriate sub-queries over the applications' schemata, and compose the results they return independently.

In the literature, there is a distinction between on-line techniques, aka Virtual Integration [18] [24], and off-line techniques, aka Data Exchange [16]. They both have been extensively studied in the context of relational, hierarchical and nested relational data. Note, however, that the Semantic Web initiative has constituted a major breakthrough regarding web information [4] [48], since it provides languages that can be used to describe rich graph-based data on the Web and technologies by means of which software agents are enabled to reason on these data and their descriptions.

Data Exchange relies on using mappings, which are queries that translate the data managed by several independent applications into a target schema. Roughly speaking, these mappings are used to materialise the target schema so that it can be queried without interfering with the original applications, and they have been of uttermost importance in the field of Data Warehousing [30].

Virtual Integration techniques rely on mappings, as well, but they materialise the target schema partially. That is, these techniques try to retrieve the minimum data that is possible to answer a query over a target schema. This makes it possible to answer them on-line, at the cost of interfering with the normal operation of the applications being integrated. Virtual Integration techniques rely on the following steps:

- Query rewriting, which takes a user query as input and reformulates it so that the result involves the application schemata only [18].

- Query planning, which divides the rewritten query into a set of sub-queries, each of which involves an application only; then, it produces an execution plan that orchestrates the sub-queries so that they can be executed as efficiently as possible [26]; the result is a set of data that come from the applications being integrated.

- Data composition, which helps aggregate the results returned by each application and transforms them into the target schema by executing the appropriate mappings. Note that data composition does not return the answer to the initial query, but a subset of data on which it must be run.

It is not difficult to realise that mappings are paramount to information integration. Beyond hand-crafted ones, there are a variety of techniques that allow generating them from user-defined correspondences amongst subsets of attributes in the target and the application schemata [45].

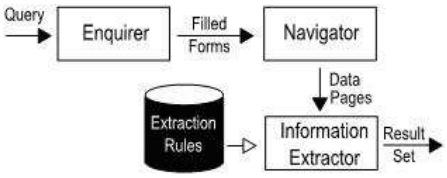

Figure 3: Structure of a typical query wrapper.

2.3 Wrapping

Wrappers play a pivotal role regarding integration since they are modules that allow several applications to interact with each other within an integration solution [1] [56]. We are only interested in so-called web query wrappers, cf. Figure 3, since others are base technologies nowadays.

Enquiring An Enquirer is a module that takes a query as input and maps it onto the appropriate search forms provided by a web application. What a query is may range from a set of field names and values to SQL-like queries. We are only interested in the latter, since current technologies support the former sufficiently.

Current research efforts include a few intelligent techniques to analyse search forms and to extract their search capabilities, i.e., the goal is to have a model that others can use to map high-level queries onto it [22] [57]. Unfortunately, the literature does not provide many other results regarding this topic.

Navigation A Navigator takes care of executing the filled forms provided by an Enquirer and navigating through the results to fetch data pages. Note that this process may lead to a data page in one step, to a no-results page, or to a so-called page hub, which is a set of interlinked pages that provide short descriptions of the information in other data pages together with links to them. (Note that term "hub" is polysemous in the literature on integration.)

Beyond navigators that rely on user-defined navigation sequences, the literature on crawling [47] provides several techniques that can be applied to solve this problem. Focused Crawling improves on traditional crawling in that it tries to avoid crawling pages that do not lead to data pages about a given topic of interest [5] [8] [43]. Other authors have worked on automated navigation pattern learners, which are algorithms that analyse a site to find the navigation sequences that lead from, e.g., a page hub to the data pages of interest [53].

Information Extraction An Information Extractor is a general algorithm that can be configured by means of rules so that it extracts the information of interest from a web page and returns it according to a structured model [2]. Rules range from regular expressions to context-free grammars or first-order clauses, but they all rely on mark-up tags or natural language properties to find which text corresponds to the data of interest.

Beyond hand-crafting information extraction rules, the literature provides a hundred proposals that can be used to learn them automatically, both in cases in which the data of interest is buried into text that is written in natural language [50] and cases in which it is buried into tables, lists and other such layouts [9]. Note that none of these techniques is universally applicable and that there are not any comprehensive empirical comparisons, which makes the decision on which to use very difficult.

The previous techniques deal with web pages. Working with PDF documents is a different setting, but it is becoming more and more important due to the ubiquity of this format, chiefly in scientific environments in which it is the de facto standard to publish articles. Current research results include several techniques that are specific to on-line bibliography databases [35] [40].

Information Verification An Information Verifier is an algorithm that analyses the result sets returned by an Information Extractor and attempts to find data that deviates largely from data that is known to be correct. They are necessary insofar the previous modules rely

on intelligent techniques that may fail if the structure of a site or a web page changes, i.e., if they are confronted with cases that were not seen previously.

Information Verifiers build on feature-based verification models. The literature provides two probabilistic techniques [29] [36] and a goodness-of-fit technique [31] to build them. Given a new unverified result set, it is necessary to calculate its features and determine if they can be considered "normal enough" according to the model. In the case of probabilistic techniques, "normality" is tested by determining the probability associated with the values of the features; in the case of goodness-of-fit techniques, "normality" is tested by checking if these values can be considered statistically equal to the values in the verification model.

3 A Research Roadmap

In this section, we first report on a number of research goals, cf. Section 3.1; we then provide a justification for each of them that builds on our analysis of the state of the art, cf. Section 3.2.

3.1 Specific Goals

We think that it is necessary to devise a framework to help software engineers develop application and information integration solutions, with an emphasis on web applications and web data. The specific goals we envisage include:

Application Integration: (G01) It is necessary to devise a Domain Specific Language for application integration. **(G02)** It is necessary to devise a series of transformations to compile it into a subset of technologies in current use. **(G03)** It is necessary to devise a new run-time system for application integration.

Information Integration: (G04) It is necessary to devise a tool to integrate web data that is represented using RDF and described using RDFS.

Web Query Wrapping: (G05) It is necessary to explore how to unify existing search form models and how we can enquire them using high-level structured queries. **(G06)** It is necessary to devise a navigator that is able to navigate intelligently from filled forms to data pages so that the number of irrelevant links that are visited is kept to a minimum. **(G07)** It is necessary to devise a software framework by means of which software engineers can build new information extraction algorithms and rule learners. **(G08)** It is necessary to explore new techniques to extract information from PDF documents. **(G09)** It is necessary to devise a new technique to build verification models that can deal with large result sets accurately.

3.2 Justification

In Sections 2.1, 2, and 2.3, we reported on the state of the art regarding application integration, information integration, and wrapping. A thorough analysis reveals that the proposals in the literature have a number of weaknesses that globally support our specific research goals, namely:

1. Current tools to engineer integration solutions are rather low-level because they tend to be general purpose, i.e., they build on constructors like interfaces, bindings, messages, orchestrations, and others that are actually intended to increase the level of abstraction at which distributed systems are devised, but do not provide any specific-purpose integration constructs, e.g., splitters, aggregators, content enrichers, or claim checkers [25], except for binding components.

2. Historically, data has been relational, which does not apply currently, with more and more data available on the Web in hierarchical, nested relational or graph-based formats, i.e., XML and RDF [48]. Integrating such data is challenging insofar the techniques in the literature have a

focus on relational data. In other words, many information integration solutions need to resort to ad-hoc techniques, which generally lead to higher development and maintenance costs.

3. Current tools focus on applications that provide programmatic or data-oriented interfaces that can be accessed by a kind of wrappers that are usually referred to as connectors, adapters or binding components. There are, however, a large number of applications that do not provide such an interface, but a user interface only, which is typically the case of end-user web applications. Integrating such applications is challenging insofar building a wrapper amounts to writing a module that emulates a human interacting with them.

In the following subsections, we delve into the justification of each specific goal.

Application Integration (G01) Working on a Domain Specific Language amounts to increasing the level of abstraction at which application integration solutions are designed, which is appealing insofar this may help reduce development costs [23]. This language must rely on constructs to represent processes, integration tasks, hubs, spokes, and other common application integration patterns [12] [13] [14] [25].

(G02) Note that such a language is not a contribution by itself, since tools like BizTalk or Camel provide similar languages. What makes the contribution unique is our emphasis on decoupling the language from the transformations devised to compile it into current technologies [49]. The previous tools do not provide such a decoupling; this makes them extremely dependent on today's technology, which, as was the case with previous advances, is expected to fade away in years, if not months. Roughly speaking, we think that engineering application integration must be sheltering it into the Model Driven Architecture, which heralds the idea of using models at three different levels of abstraction and automatic transformations from higher-level models into lower-level models [37]. This approach proved to cut off development and maintenance costs [19] [23].

(G03) We also think that it is important to devise a run-time system for application integration that is based on a per-task thread allocation policy [15]. A notorious limitation of current runtimes is that they use threads inefficiently in cases in which an integration process requests an application to perform a task and has to wait for the results; note that this case is very frequent, and that an application may take from seconds to hours to react, e.g., think of an action that depends on a person or a setting in which integration requests are given less priority during work hours. The literature proposes a technique called dehydration/rehydration to deal with these cases [11] [55]; the idea is to detect integration processes that have been waiting on a request for too long, saving them to disk (dehydration), and resuming them when the results arrive (rehydration). Although this solution is very common, our experience with our partners proves that it is far from scalable. We think that it is necessary to devise a new runtime system that allocates threads per task, instead of per process. This finer granularity allows threads to be used more efficiently since they can be relinquished the sooner as a task completes or a request is sent to an application [33].

Information Integration (G04) Recall that a mapping is a query that translates data from a number of application schemata into a target schema. Current research results regarding mapping generation focus on the relational and the nested relational models [18] [45]. To the best of our knowledge, there are not any results to deal with graph-based data [39], which is becoming more and more pervasive due to the increasing popularity of RDF and RDFS [48]. The main problem with these languages is that they allow for classes, subclasses, properties, sub-properties and data that have several unrelated types,

which is not the case for previous data models; furthermore, the notion of referential integrity is much weaker than in relational data, and the standard query language is SPARQL, which deviates largely from the SQL and XQuery subsets that have been studied so far.

Wrapping (G05) The problem with Enquirers is that they are a recent research topic and there are not many results available [38]. This clearly justifies exploring new techniques to implement them. Our idea is to model forms as if they were parameterised views, which shall require devising a new form model that allows, not only for the fields and actions available, but also for their semantics [22] [57]. This model shall allow us to transform our problem into the problem of answering queries using views [18] and query-oriented programming interfaces [41]. We also need to delve into the problem of query feasibility, i.e., the problem of checking whether a query can be mapped onto the available forms.

(G06) Note that visiting many irrelevant links is problematic insofar it has an impact on a server's response time and on the bandwidth used. Our preliminary studies prove that typical hub pages include 60-90% irrelevant links [20], which makes it clear the need for intelligent navigators. Traditional crawlers [47] navigate a site by retrieving all of the pages they find, which means visiting all irrelevant links. Focused crawlers only crawl pages about a given topic or pages that may lead to them, but they are not able to set relevant links apart and the ratio of irrelevant links visited is far from negligible [5] [8] [43]. Automated navigation pattern learners can be the solution to our problem, but the existing proposals have some shortcomings that must be addressed. A recent proposal by Vidal et al. [53] is able to learn navigation patterns that set irrelevant pages apart without fetching them, i.e., the decision is solely based on their links; unfortunately, it seems unable to navigate through hub pages and does not guarantee that all relevant data pages are retrieved.

(G07) Developing a software framework is appealing insofar it shall help reduce development costs and shall allow side by side comparisons; note that the literature provides many results regarding information extraction, but they are currently not comparable to each other because they have been developed using different technologies and validated using different data [9] [50]. From an industrial point of view, this is problematic because deciding which the most appropriate algorithm is becomes a matter of trial and error. SRV ranges amongst the most general and powerful information extraction systems [7] since it relies on first-order logic extraction rules and on a set of user-defined predicates to implement features that range from the tag within which a piece of data is rendered to its natural language role. This makes SRV one of the most interesting proposals to be implemented within the previous framework, since it may be used to prototype a variety of other systems. Unfortunately, its learner is based on FOIL [46], which is quite a complex algorithm that hinders its applicability in production environments. It is necessary to devise several optimisations and heuristics to improve the efficiency of the FOIL rule learner so that it can be used efficiently for information extraction tasks [44].

(G08) Extracting information from PDF documents is a recent research area in which there are very few results, most of which focus on digital library settings and have low accuracy rates [35] [40]. We think that it is worth exploring new techniques that build on using visual properties and data mining techniques since our preliminary results seem promising enough [51].

(G09) An important drawback of existing techniques [29] [31] [36] is that the resulting verification models tend to be less and less accurate as the number of features examined increases. Our experience is that real-world result sets usually involve hundreds of features, which makes the existing techniques of little interest [52]. Note that result sets can naturally be represented as multi-dimensional vectors whose components are the values of the features applied to them. The distance between two arbitrary result sets can then be defined as the distance between their corre-

sponding vectors. Thus, an unverified result set must be considered valid if it is "similar enough" to the known valid result sets; otherwise, an alarm must be signalled. Determining what "similar enough" means can be dealt with using intelligent techniques from the field of machine learning. The problem is that these techniques usually rely on the hypothesis that there is a sufficiently large set of both valid and invalid samples, which is not the case for information verification. Our focus shall be on generating invalid, representative data that allows to build verification models that can reliably deal with large sets of features.

4 Conclusions

In this article, we have presented and justified a research roadmap regarding application and information integration, with an emphasis on web applications that do not provide a programmatic interface, but a user interface only. Such applications are common in practice, and, more often than not, they provide rich data from which advanced business processes might benefit. Unfortunately, recent research reports unveil that integrating such applications is challenging insofar they tend to be neglected, current tools are rather low-level, and more research effort is required regarding graph-based data.

The roadmap we have presented provides a straight research path that we hope shall end on a number of engineering tools to deal with the complexity of integrating such applications, i.e., metaphors, technologies, tools, best practices, and methods.

References

[1] Coping with Web Knowledge. J.L. Arjona, R. Corchuelo, J. Peña. AWIC, 165-178. 2003.

[2] From Wrapping to Knowledge. J.L. Arjona, R. Corchuelo, D. Ruiz, M. Toro. IEEE Trans. Knowl. Data Eng., 19(2):310-323, 2007

[3] Workflow Management. W. van der Aalst, K. van Hee. The MIT Press. 2004.

[4] A Semantic Web Primer (2nd edition). G. Antoniou, F. van Harmelen. The MIT Press. 2008.

[5] A Genre-Aware Approach to Focused Crawling. G.T. de Assis, A.H.F. Laender, M.A. Gonçalves, A.S. da Silva. WWW Journal, 12(3):285-319. 2009.

[6] Information Integration in the Enterprise. P.A. Bernstein, L.M. Haas. Commun. ACM, 51(9):72-79. 2008.

[7] Learning to Construct Knowledge bases from the World Wide Web. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S.Slattery. Artif. Intell., 118(1-2):69-113, 2000.

[8] Mining the Web's Link Structure. S. Chakrabarti, B. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J.M. Kleinberg. IEEE Computer, 32(8):60-67. 1999.

[9] A Survey of Web Information Extraction Systems. C.-H. Chang, M. Kayed, M.R. Girgis, K.F. Shaalan. IEEE Trans. Knowl. Data Eng., 18(10):1411-1428. 2006.

[10] Application Integration Services. DataMonitor. 2008

[11] Pro BizTalk 2006. G. Dunphy, A. Metwally. Apress. 2006.

[12] A DSL for Enterprise Application Integration. Rafael Z. Frantz. IJCAT, 33(4):257-263. 2008

[13] Integración de Aplicaciones. R.Z. Frantz, R. Corchuelo. Zoco/CAEPIA, 1-11, 2007.

[14] Advances in a DSL for Application Integration. R.Z. Frantz, R. Corchuelo, J. González. TJISBD, 54-66, 2008.

[15] Towards a Fault-Tolerant Architecture for Enterprise Application Integration Solutions. R.Z. Frantz, R. Corchuelo,

C. Molina-Jiménez. OTM Workshops, 2(2):294-303, 2009.

[16] Data Exchange. R. Fagin, P. G., Kolaitis, R.J. Miller, L. Popa. Theor. Comput. Sci., 336(1):89-124. 2005.

[17] Consulting, Development and Integration Services, 2008-2013. Gartner Group. 2008

[18] Answering Queries using Views. A.Y. Halevy. VLDB Journal, 10(4):270-294. 2001.

[19] Impacts of Changes in Enterprise Software Construction for Telecommunications Model Driven Architecture. M. Herzog, editor. EURESCOM Project P1149. 2002.

[20] Intelligent Web Navigation. I. Hernández. Taller de Trabajo Zoco'09/JISBD, 2009.

[21] Exception Handling in Workflow Management Systems. C. Hagen, G. Alonso. IEEE Trans. Software Eng., 26(10):943-958. 2000

[22] Towards Deeper Understanding of the Search Interfaces of the Deep Web. H. He, W. Meng, C.T. Yu, Y. Lu, Z. Wu. WWW Conf., 133-155. 2007.

[23] Domain-Specific Languages in Practice. F. Hermans, M. Pinzger, A. van Deursen. MoDELS, 423-437. 2009.

[24] Data Integration. A.Y. Halevy, A. Rajaraman, J.J. Ordille. VLDB Conf., 9-16. 2006.

[25] Enterprise Integration Patterns. G. Hohpe, B. Woolf. Addison-Wesley. 2003.

[26] Adapting to Source Properties in Processing Data Integration Queries. Z.G. Ives, A.Y. Halevy, D.S. Weld. SIGMOD Conf., 395-406. 2004.

[27] Involving Web-trading Agents and MAS: An Implementation for Searching and Recovering Environmental Information. L. Iribarne, N. Padilla, J.A. Asensio, F. Muñoz, J. Criado. ICAART (2), 268-273, 2010

[28] A Trading Service for COTS Components. L. Iribarne, J.M. Troya, A. Vallecillo. Comput. J., 47(3):342-357, 2004

[29] Wrapper verification. N. Kushmerick. WWW Journal, 3(2):79-94. 2000.

[30] The Data Warehouse ETL Toolkit. R. Kimball, J. Caaserta. Wiley. 2004.

[31] Accurately and Reliably Extracting Data from the Web. C.A. Knoblock, K. Lerman, S. Minton, I. Muslea. IEEE Data Eng. Bull., 23(4):33-41. 2000.

[32] Orchestrating Web Services with BPEL. P. Louridas. IEEE Software, 25(2):85-87, 2008.

[33] Designs of Bisimilar Petri Net Controllers With Fault Tolerance Capabilities. L. Li, C.N. Hadjicostis, R.S. Sreenivas. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 38(1):207-217. 2008

[34] Web Services up for Information Integration. V. McCarthy. CIO Magazine, 26/01. 2006

[35] Metadata Extraction from PDF Papers for Digital Library Ingest. S. Marinai. ICDAR, 251-255, 2009

[36] Mapping Maintenance for Data Integration Systems. R. McCann, B.K. AlShebli, Q. Le, H.Nguyen, L. Vu, A. Doan. VLDB Conf., 1018-1030. 2005.

[37] MDA Guide V1.0.1. J. Miller, J. Mukerji, editors. Doc Number 03-06-01. 2003.

[38] From Queries to Search Forms. C.R. Osuna. IJCAT, 33(4):264-270. 2008.

[39] SPARQL Query Splitter. C.R. Osuna, D. Ruiz, R. Corchuelo, J.L. Arjona. JISBD, 320-323, 2009

[40] Enriching a Document Collection by Integrating Information Extraction and PDF Annotation. B. Powley, R. Dale, I. Anisimoff. DRR, 1-10, 2009

[41] Interactive Query Formulation over Web Service-Accessed Sources. M. Petropoulos, A. Deutsch, Y. Papakonstantinou. SIGMOD Conf., 253-264. 2006.

[42] Service Oriented Architectures. M.P. Papazoglou, W.-J. van den Heuvel. VLDB Journal, 16(3):389-415, 2007.

[43] Link Contexts in Classifier-Guided Topical Crawlers. G. Pant, P. Srinivasan. IEEE Trans. Knowl. Data Eng., 18(1):107-122. 2006.

[44] Tuning up FOIL for extracting information from the web. P. Palacios, I.F. de Viana. IJCAT, 33(4):280-284, 2008

[45] Translating Web Data. L. Popa, Y. Velegrakis, R.J. Miller, R. Fagin, M.A. Hernández. VLDB Conf., 598-609. 2002.

[46] Learning First-Order Definitions of Functions. J. R. Quinlan. J. Artif. Intell. Res., 5139-161, 1996.

[47] Crawling the Hidden Web. S. Raghavan, H. Garcia-Molina. VLDB Conf., 129-138. 2001

[48] The Semantic Web Revisited. N. Shadbolt, T. Berners-Lee, W. Hall. IEEE Intelligent Systems, 21(3):96-101. 2006.

[49] Towards Automatic Code Generation for EAI Solutions using DSL Tools. H.A. Sleiman, A.W. Sultán, R.Z. Frantz, R. Corchuelo. JISBD, 134-145, 2009.

[50] Adaptive information extraction. J. Turmo, A. Ageno, N. Català. ACM Comput. Surv., 38(2). 2006.

[51] Extracción de Información usando Características Visuales. M.D. Vargas. Technical Report. University of Huelva. 2009.

[52] Selección de Características para Mejorar los Modelos de Verificación de Información en EAI. I.F. Viana, J.L. Arjona, J.L. Álvarez, P. Abad. JISBD, 21-32, 2009.

[53] Structure-Based Crawling in the Hidden Web. M.L.A. Vidal, A.S. da Silva, E.S. de Moura, J.M.B. Cavalcanti. J. UCS, 14(11):1857-1876. 2008.

[54] Aligning Relationships. J. Weiss. IBM Global Services Report. 2005

[55] Oracle SOA Suite Developer's Guide. M. Wright, A. Reynolds. Packt Publishing. 2009.

[56] A Note on Web Intelligence, World Knowledge and Fuzzy Logic. L.A. Zadeh. Data & Knowledge Engineering, 50(3): 291-304. 2004.

[57] Understanding Web Query Interfaces. Z. Zhang, B. He, K.C.-C. Chang. SIGMOD Conf., 107-118. 2004.