An Approach to Software Selection Using Semantic Web

Elena Daehnhardt and Yanguo Jing

***Abstract*—For a selection of software applications well-fitting to business needs, accurate knowledge of the needs and available technical skills are required. In a survey we conducted, most of the participants indicated that they consult industry solutions and employ Internet search engines to find suitable software products. However, since a keyword search does not consider semantics of terms describing software characteristics, pertinent search results might not be retrieved. We made a hypothesis that the addition of semantics describing software characteristics could assist in finding software-related information in a more structural way, which would improve such a search. In order to support the software selection decision we investigated the application of semantic web technology. In addition, we proposed a new approach for an information exchange between software providers and consumers, with the aim of a platform-independent information management. For the description of software products we employed a Resource Description Framework and SPARQL for further querying their descriptions. Our prototype, assisting in cross-platform information exchange and structured search, demonstrated an information exchange between software vendors and customers. This allows users who purchase software to be supported in their decision, and software developers to receive feedback for further software improvement. Experiments with the prototype revealed a performance overhead of SPARQL queries when compared to MySQL database queries. It is proposed that further optimization for SPARQL queries is needed in order to improve the response time of working with triple store. Overall, the test outcomes demonstrated that the proposed solution supports the process of selecting software and has the potential of improving IT infrastructure management practices.**

***Index Terms*—semantic web, knowledge management, software selection, decision-making support.**

I. Introduction

IT has been recognized that a well organized IT infrastructure has a positive impact on business performance [1]. This is why IT assets should be properly chosen, particularly when they are of importance for an organization [2]. However, software acquisition is a complex process, requiring an understanding of technical and business issues involved [3]. When business processes depend on relevant technologies, system and resource requirements which are not well defined can lead to deployment failures. Ill-suited or low quality solutions can be devastating for businesses. Software faults can lead to significant financial losses and even bankruptcy [4]. Some companies try to minimize risks of software development. For this, improving communication between stakeholders such as software developers and their customers is paramount [4].

Manuscript received April 04, 2013; revised December 5, 2013.

Elena Daehnhardt is a PhD student in Computer Science at Heriot-Watt University, Edinburgh. E-mail: elena@daehnhardt.com.

Dr. Yanguo Jing is an Associate Professor in Computer Science at Faculty of Life Science and Computing, London Metropolitan University. E-mail: y.jing@londonmet.ac.uk

On the other hand, software management policies by Dell [5], Distributed Management Task Force Standard [6], Microsoft guidelines [7], and ISO/IEC 19770 regulations [8] can be employed while managing and acquiring IT assets. The applied system approach helps to minimize the risks of project failure. However, regulations become quickly obsolete and often fail to provide information on how and where from software can be purchased [9].

Moreover, Information Technology (IT) is developing rapidly and constantly new solutions appear. To cope with this, search engines and web directories can be employed to gather software-related information. However, a keyword-based search does not reflect the meaning contained behind keywords used for defining software characteristics. Here, semantic web technologies such as Resource Description Framework (RDF), SPARQL Protocol and RDF Query Language (SPARQL) can assist in the communication between software providers and consumers [10]. A structured search having semantics in mind can help in retrieving software products fitting user needs. This is why semantic-based decision support systems could be considered for managing and analyzing information regarding available software solutions.

In this context, are semantic technology and supporting tools mature enough to facilitate decision support in real time? Would an open world approach be feasible to realize and yet comparable with the traditional approach in which information on software products is stored in relational databases? These questions motivated us to compare these two approaches for the purpose of decision making support in the software selection process. The main purpose of the investigation was to understand software selection issues in depth and learn how semantic web technologies might help in selecting software which is well-suited to business needs. The main research questions included:

1) Which issues are paramount in the software selection process and how is it performed in practice?
2) Which tools are used for supporting decisions in software acquisition?
3) How can web technologies help in the decision making process concerning software selection?

In order to understand software selection problems, an online survey was conducted, which revealed opinions of IT managers and professionals. The online survey helped to find out the main sources of information on software products and the most important factors to consider for software selection. Having these factors as selection parameters, we realized a multi-criteria decision making algorithm with the aim of supporting software selection. The prototype used semantically enriched software descriptions for finding software products which match user queries. The prototype was tested and its performance compared with the relational database approach

to check the feasibility of the proposed semantic approach.

Our main contributions include:

- An analysis of important software characteristics to be considered in the software selection process and information sources used for decision support;
- The development of a semantic web based application to assist in software selection decisions;
- A comparison of relational and semantic approaches to software information storage and query.

In the next section we outline previous research and related background information. In section 3 we describe our research methodology. Section 4 presents the survey results, prototype implementation details and experimental results. In section 5, we discuss questions on semantic web adoption, possible implications for IT management and suggest prototype improvements. Finally, we present our main findings on the basis of the survey results and prototype test and conclude by outlining future research directions.

II. Related Work

When selecting software for an organization, various software products and their characteristics are analyzed to find out the solution which matches best to business needs. This is a typical task for decision making when considering a set of alternatives based on a set of criteria. Various Multi-criteria Decision Making (MDCM) techniques, optimization methods, neural networks, rule-based decision support systems and their variations can practically be applied for software selection [10], or other tasks when the selection is performed amongst alternatives. For selecting geographic information systems, [11] use a rule-based expert system and Analytic Hierarchy Process (AHP) MDCM approach. They employ software evaluation quality traits as software selection attributes. However, their approach is database-based and requires regular database updates with new information on the software [11].

Some of the MCDM methods are more accurate then others, and other methods also consider uncertainty and employ fuzzy logics [12]. For instance, [13] use fuzzy logic and AHP for enterprise software selection. [14] employed fuzzy AHP for decision support in selecting software developers and used software quality criteria as selection attributes. Analytic Network Process (ANP) technique is used in [15] for selecting simulation software packages. In accord to [12] (cited in [10]), ANP is more precise then AHP, since ANP considers also interdependence between different attributes on various levels of hierarchy, which is used to describe software products or other alternatives.

Furthermore, the various methods of decision making, their requirements and suitability for a particular decision making task requires certain decision making skills and knowledge on how to select an appropriate method [16]. For adapting to different skill levels of decision makers and their requirements, [16] proposed the intelligent decision support system working with the rule-based knowledge base, describing various multicriteria decision making methods and providing assistance for the decision maker in accordance with her requirements.

As stated in [16], the selection process for information systems involves many professionals having knowledge on organisational needs assisting in the selection process. When the software products are acquired from external vendors, the software acquisition process [17] comprises typically the steps of communicating with software providers, analyzing outstanding financial issues and determining the system requirements of software products fitting to the IT infrastructures at place. Many software developers market their products globally. Since their customers might not be able to communicate with software providers face-to-face, software products are advertised online and can often be delivered immediately over the Network. The Internet provides a very natural e-commerce platform for marketing and supporting software products.

As stated in [18], software deployment success depends largely on the suitability of software according to user needs. In order to analyze the fitness of software products to business needs, various decision tools are available on the World Wide Web. Online software demonstrations such as provided by Runaware [19], Tucows [20] for software downloads, and platforms such as Technology Evaluation Centers [21] can thereby assist in selecting software solutions. However, existing web marketing tools are not seamlessly integrated into IT infrastructures and normally do not consider information on particular hardware and software installations at customer places. In the absence of this information, it cannot be guaranteed that a selected solution fits best to user needs [22].

Furthermore, information on the Web is not well structured. Search engines and indexes working with keywords that do not consider semantics might output too many irrelevant results. Meta-tags used for matching search keywords do not possess information on the meaning of the objects described. In order to add semantics to web resources, semantic web languages such as RDF and Web Ontology Language (OWL) can be applied. While OWL provides more powerful reasoning capabilities, it is more complex and less efficient when processing large data volumes [23]. RDF can also be used for describing abstract and real-life concepts, their properties and relationships [24]. RDF provides a means of information representation in a platform and user-independent format, which can further be queried using computer languages such as SPARQL [25]. SPARQL queries can be compared with SQL queries which are performed on relational database records.

One of the principal differences between Semantic Web (SW) and relational approaches is that SW complies with Open World Assumption, and allows working with the distributed over the Network resources rather than being only limited to closed relational storages [10]. SW was already used in applications such as information management in a software development process [26], products information management [23], online education [27] and matchmaking services [28]. In this work, SW was applied to the task of software selection decision support, enabling not only a flexible construction of the software selection queries, but also a platform-independent communication between software providers and their customers.

III. Methodology

A. Survey

For a better understanding of the software selection process, an online survey was conducted. This included open-end questions, which were analyzed with the help of content analysis, and closed questions analyzed using descriptive statistics. All codes and procedures were described, results triangulated and reviewed by survey participants provided with the report [29], as advised in [30]. The focus of the investigation was to identify factors having an impact on software acquisition and whether automated tools can be employed to support decisions in software selection.

As reported in [10], an online questionnaire [31] was filled out by 56 professionals, 39% and 32% of them coming from Asia and Europe respectively, as seen from Figure 1. About 45% of respondents worked in the Communications and IT industry, Table I.

Table II shows that more than 70% of participants had ten or more years of experience in the IT sector and a quarter of respondents were responsible for IT decisions in their organizations, as seen from Table III.

About half of all respondents indicated that their companies follow a systematic approach while selecting software. As seen from Table IV, the majority of survey participants, 77% and 61% respectively, stated that they learn about software solutions used in their industry and with the help of Internet search agents. According to 60% of participants, software selection would benefit from decision support software, which would help to take on technology-related decisions faster and possibly decrease related costs.

Moreover, survey results provided an outlook on participants' opinions on the software selection process and important software characteristics to be considered. Table V shows important software properties for consideration in the software acquisition process. As derived from results of content analysis and descriptive statistics, fitness to business needs, suitability of software functionality, standards compliance and IT infrastructure integration easiness were the most critical factors for survey respondents.

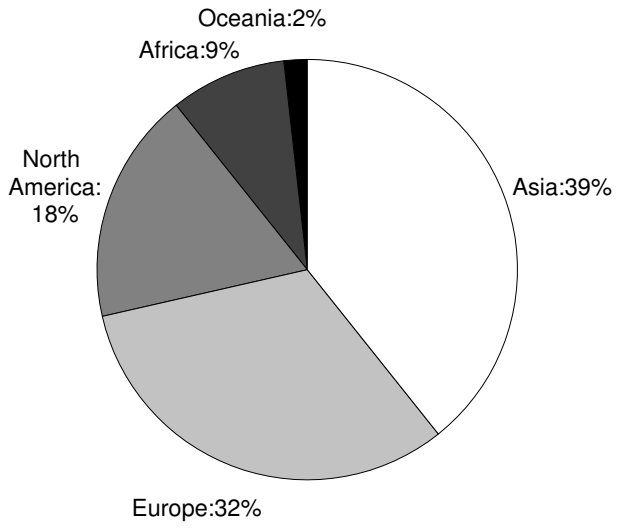


Fig. 1: Survey Participants' Locations

TABLE I: Industries

Industry sector	Respondents
Communications and IT	44.6%
Finance and accounting	8.9%
Healthcare	8.9%
Public sector and education	7.1%
High Tech	5.4%
Manufacturing	3.6%
Trade and marketing	3.6%
Utilities	1.8%
Aerospace and Defense	1.8%
Law and professional services	1.8%
Other	12.5%

TABLE II: IT Experience

IT Experience	Respondents
More than 9 years	73%
From 3 to 9 years	22%
Less than 3 years	5%

TABLE III: IT Decision Roles

IT Role	Respondents
IT Development and Support	32%
Responsible for IT Decisions	25%
Taking Part in IT Decisions	20%
Use IT Systems	11%
IT Consultancy	7%
Marketing and Other	5%

TABLE IV: Information Sources

Sources	Respondents
Industry Solutions' Analysis	77%
Internet Search Engines	61%
Publications and White Papers	54%
Other Businesses, Competitors	52%
Business Partners and Networking	41%
Trade Shows	39%
Consultancies	27%
Vendor Directories, Web Resources	21%
Other	9%

TABLE V: Software Characteristics

Software Trait	Respondents
Fitness to Business Needs	41%
Sufficient Functionality	12%
Standards Compliance	11%
Integration Easiness	11%
Affordable Costs	7%
Ease of Customization	7%
Ease of Maintenance	5%
Performance	4%
User-friendliness	2%

B. Conceptual Framework

Based on conceptual framework outlined from the literature survey and in [18], a model describing software products and their properties was created [10]. The meta-model proposed in [18] was designed to describe software

products and their properties based on the software quality traits defined in the ISO framework. For establishing a set of attributes and metrics, we adopted the general principles outlined in [18]. However, we did not strictly follow the ISO standard and metrics as in [18]. We also did not consider possible dependencies between software characteristics. Instead, based on the survey results and insights described in [10], we proposed to employ software traits such as integration easiness, usability, efficiency, deployment costs and functionality sets with properties such as listed in Table VI for creating our ontology.

TABLE VI: Software Characteristics

	Characteristics
1.	Usability
2.	Maintainability
3.	Performance
4.	Integration Easiness
5.	Functionality
6.	Costs Associated
7.	Usage Industry
8.	Developer

C. Ontology and Data Structures

Our reasoning was, that the set of software traits should be flexible for easy adaptation to any software and user requirements. This way the ontology can be further tailored to a specific usage domain. For instance, an antivirus software was described by a graph such as shown in Figure 2.

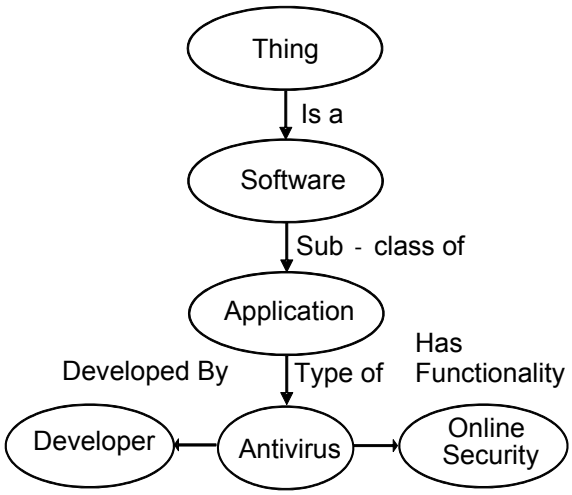


Fig. 2: RDF graph

The ontology was a basis for the creation of a RDF structure used for storing software products information, which example is presented in Figure 3. The description provided can be effortlessly extended with new parameters including software price, usability and performance ratings. Similarly to [18], we defined a set of metrics for assessing the parameters related to user requirements. Sometimes these metrics can be defined as sets of strings ("labels" in [18]).

Conveniently, SPARQL can be used to match the strings using simple string comparisons or regular expressions. For instance, a SPARQL query shown in Figure 4 helps to filter out entities which are not developed by "Developer".

```
<soft:Application_Software rdf:about=
software:hasFunctionality=

dc:description=
software:developedBy=
software:hasURL=                          />
```

Fig. 3: An Example of a Software Description

```
PREFIX software: <http://example.com/rdf/
    soft/>
SELECT * WHERE {
 ?s software:developedBy             .}
GROUP BY ?s
```

Fig. 4: Retrieving Software Developed by "Developer"

Figure 5 shows a template which we used for dynamically generating SPARQL queries in accord with the user request coming from the web interface. We used numerical values to define scores for the relevant attributes such as integration and maintainability easiness. As seen from Figure 5, our SPARQL queries can be quite versatile when compared with the SQL queries. We can define not only fixed parameters, but also include optional parameters, which are not so easy to implement in SQL when working with non-available data, in which case SQL left joins might be considered [32].

```
PREFIX software: <http://example.com/rdf/
    soft/>
PREFIX dc: <http://purl.org/dc/elements
    /1.1/>
SELECT DISTINCT SelectParameters
  WHERE {
    ?s dc:title ?title .
    ?s software:hasURL ?url .
    ?s software:hasCosts ?hasCosts .
    ?s dc:description ?description .
    RequiredParameters
    OptionalParameters
    RegexParameters
    FilterParameter
    }
  GROUP BY ?title
  ORDER BY ASC( ?hasCosts )
  LIMIT 10
```

Fig. 5: Query Template (SPARQL Query Can be Extended with Other Mandatory and Optional Parameters)

Moreover, we can exploit regular expressions when dealing with strings, for instance when matching software vendor names. The FILTER statement is used for filtering out software products which do not correspond to price limits imposed by the user. It is important to mention that other properties such as deployment platform requirements and other quality characteristics could also be used for describing software characteristics. Compared to the relational database schema, the flexibility of RDF schema alteration is advantageous when underlying data structures are likely to change. Therefore, new software properties can be added to the software descriptions with minimal code alterations.

D. Prototype

Figure 6 shows the main components of the prototype. From an user point of view, the prototype consists of two main modules for software providers and their potential customers. These modules were created with the help of CakePHP framework enabling interface generation and database connectivity. Software providers first describe their software products using web interface for capturing software properties and calling procedures for storing data in the database. Software purchasers are then able to query the database for finding software products matching their requirements as shown in Figure 6. For testing, we implemented "Testing module" and "RDF Generator" to simulate users' work with the prototype.

We employed the ARC2 [33] library realizing RDF processing functionality and triple storage. MySQL relational database was used for storing RDF triples describing software products. RDF descriptions helped to abstract from the underlying platform. It is important to mention that RDF language does not enable sophisticated inference capabilities when compared to OWL; however, RDF is more efficient for larger data volumes [34]. Since the aim of the project was to enable platform-independent data integration, RDF format was used for publishing information on software products accompanied with meta-data. Moreover, a structured search over RDF triple stores helped to restrict search results and take into account semantics of the concepts involved.

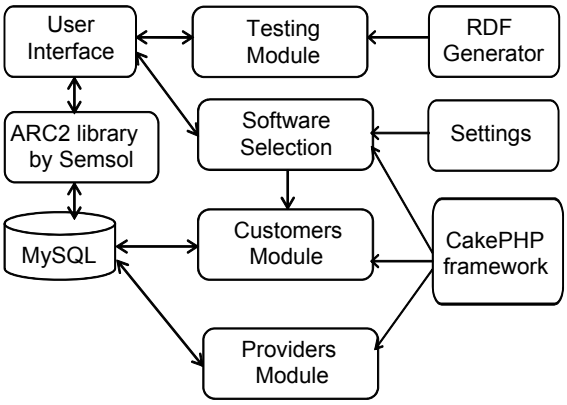


Fig. 6: Prototype Components

For querying RDF records, we employed SPARQL. RDF descriptions help to abstract from the underlying platform of the data storage, while SPARQL end points or specialized web interfaces can be used for querying data stores. Software properties such as functionality traits, deployment costs, integration easiness, maintainability and developers' information were defined as properties of the software product to be selected. The selection requirements are passed for creating SPARQL query such as described above, including essential and optional parameters such as cost limitations when defined by the user.

Since Multi-Criteria Decision Making (MCDM) methods are often practically applied for selecting software within available alternatives [12], for instance, when selecting enterprise information systems [35], the software selection algorithm was programmed using MCDM approach and with the following additive scoring function for rating software

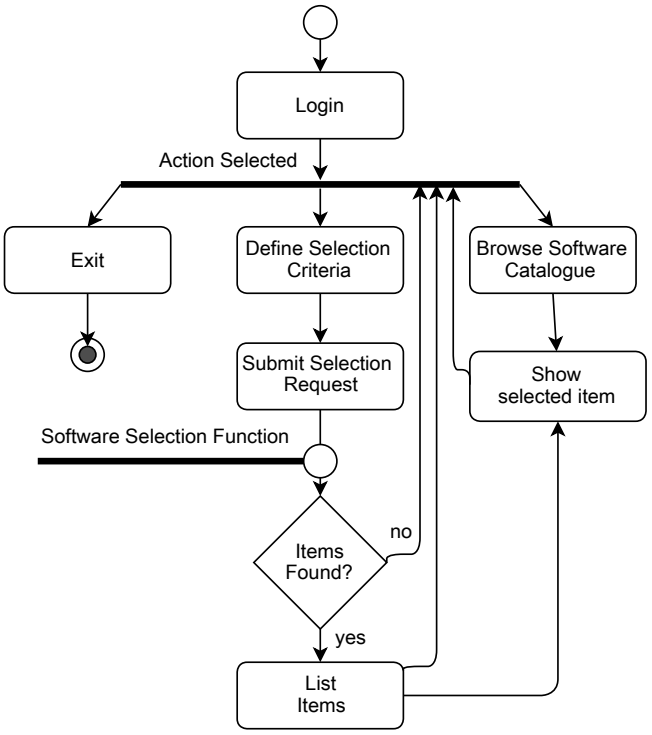


Fig. 7: Software Selection Workflow

alternatives [36] (cited in [10]):

$$R = max_i \frac{\sum_{j=1} p_{ij}}{\sum_{j=1} w_j}$$

In the formula above, a software product i is denoted by its j (attributes) performance parameters p_{ij}, which are weighted as w_j in accord with user requirements [10]. The performance parameters were calculated as products of utility scores and their weights w_j. The software characteristics' scores were defined in the software descriptions to enable their comparison. In case, that an obligatory selection criteria does not match the product descriptions and associated scores, the product is excluded from the result set. When products satisfied the essential selection parameters, non-essential parameters were further considered. In result, information on selected software products and their rating was stored into the decision matrix, in which each row represented one software product and each column one selection criterion.

Figure 7 shows the software selection workflow as realized in our prototype. The main user activities include creating and executing software selection requests, and further browsing software selection results. Filling out the web form, users define the selection criteria consisting of the software traits and their importance ranking. The most important software selection factors mentioned by our survey participants were adopted as software selection parameters in our software selection algorithm.

Overall, we implemented our prototype to facilitate information exchange between software providers and customers. This is achieved using RDF and SPARQL computer languages enabling to realize open-world and platform-independent storage. Next, we tested our prototype in order to compare its performance with the relational

database approach. Based on the results of prototype testing, we discuss possible improvement options while considering user demands investigated in the online survey.

IV. Analysis of Results

A. Experimental Setup and Results

In our experiments, we identified differences in the consumption of computer resources for different prototype usage scenarios. The usage scenarios included creation of software descriptions, creation and execution of the software selection queries, performed using relational (MySQL) and triple store storages. It is important to mention, that we employed ARC2 triple store built on the MySQL storage and we expected related overheads. Our particular interest was to understand the feasibility of exploitation of the triple store for building solutions working under "Open-World" assumption, which is deployed on the existing and well tested "Linux-Apache-MySQL-PHP" platform.

We performed a set of tests for assessing the performance of the prototype. Our test environment was organized on Mac OS X with 32 GB memory and 3.4 GHz Intel processor. We installed PHP and MySQL version 5. For evaluating our test results, we established a set of baseline values including memory consumption of 8Mb, Central Processing Unit (CPU) usage of 4 waiting processes and an execution time boundary of 5 seconds.

Our main goal was to compare prototype behavior in different test settings and usage scenarios. Our tests differed only in their load setup and followed the same algorithm:

- STEP 1. Define number of prototype users, number of added software descriptions and average delay for one user session.
- STEP 2. Emulate users' work for the six scenarios mentioned below in accordance with the settings provided in the first step.
- STEP 3. After running tests, analyse test logs with the help of descriptive statistics.

The testing module emulated work of prototype users creating software selection requests and loading software information into the database. RDF Generator randomly generated RDF structures describing software products. The tests performed included scenarios of Relational Database (RD) queries and queries on the RDF triple store, as follows:

- Scenario A randomly created a software description record stored into the MySQL table. Scenario A was used for filling out the database with software descriptions, simulating the work of software providers.
- Scenario B converted the output of test A into RDF format, which was further stored into the triple store.
- Scenario C generated software selection queries, which were then stored into the relational database.
- Scenario D used the output of test C, which provided the software selection request, and was then transformed into the SPARQL query. Next, SPARQL query was used for filtering software alternatives found to be compliant with the obligatory search requirements. The products selected were then rated applying the MCDM approach algorithm.
- Scenario E emulated the customer searching a software product containing a required functionality, using MySQL query.
- Scenario F was similar to scenario E, using however SPARQL queries performed in ARC2 data store.

We simulated user input by adding software descriptions and performing software selection requests. In our two tests, on each execution cycle we perform all scenarios for all users (5 users for Test 1, and 50 for Test 2) as follows:

- Test 1 measured prototype performance for scenarios A to F in 3000 observations (500 cycles for executing all scenarios), with 5 software products and selection requests added to the database on each cycle and a predefined user delay ranging from 0 to 45 seconds. In total, we added 50 software product descriptions.
- Test 2 measured prototype performance in 1200300 observations (4000 cycles), with 200000 software products and selection requests added to the database and a predefined user delay of 0 seconds.

B. Datasets

Table VII presents two datasets used for performing our experiments. Datasets D1 and D2 were used for performing Tests 1 and 2 to analyze prototype performance on different user delays and load levels, respectively. Each observation corresponds to the execution of one of the user scenarios (from A to E), performed in one cycle.

TABLE VII: Datasets

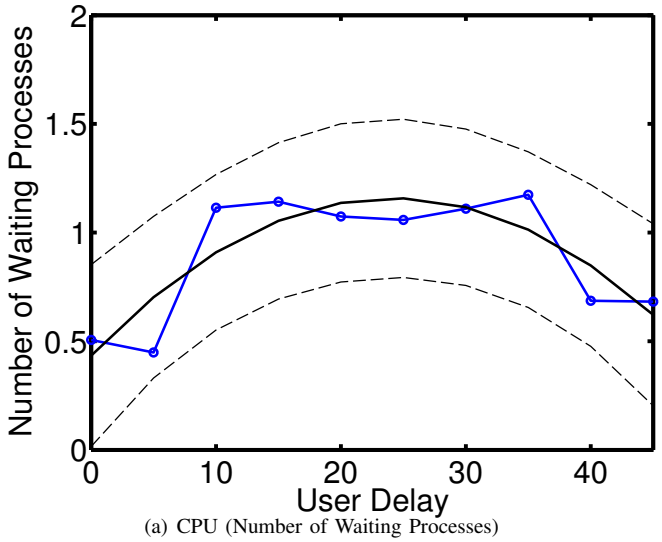
Dataset/ Test	Observations	Cycles	Total products (queries) added	Number of users	User delay
D1/T1	3000	500	50	5	varied from 0 to 45 milliseconds
D1/T1	1200300	4000	200000	50	0 milliseconds

Finally, data on response times, central processor usage and memory consumption were collected.

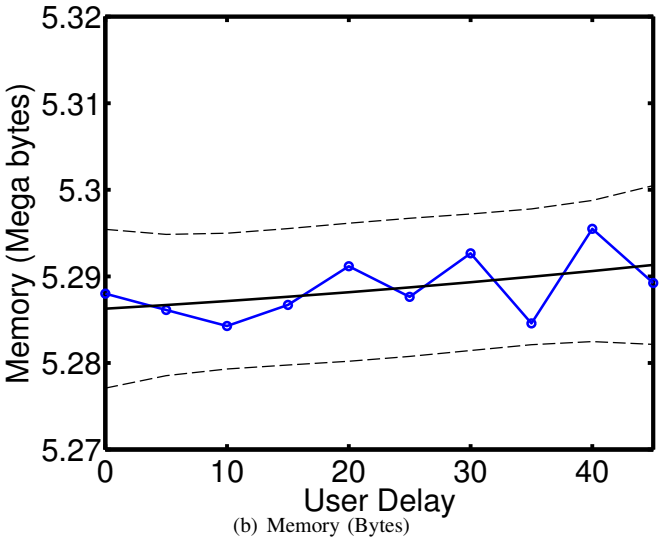
C. Test Results

As seen from Figure 8, with user delays of 5 and 10 seconds, respectively, we achieve minimal average values of CPU (0.45 waiting processes) and memory (5.5MB) consumption. However, in our next test we defined a user delay of zero seconds to assess prototype behavior under heavier load with more products added to the database.

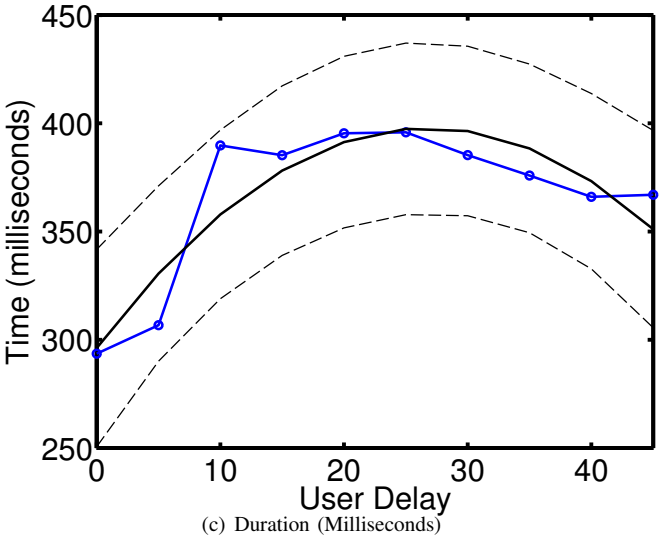
As seen from Table VIII (Test 1), showing mean values of response time, the number of queued processes and memory usage were within predefined upper boundaries of 5 seconds, four waiting processes and 8Mb of memory, respectively. Moreover, Figure 9 and Table VIII show that scenario D (software selection function) takes in average the largest amount of CPU resources, followed by scenarios E, F, B, A and C. Scenario D also takes more time to execute, followed by scenarios B, F and the rest. In contrast, as Table VIII shows, A and E scenarios take less than a millisecond in average to execute operations on the MySQL database.



(a) CPU (Number of Waiting Processes)

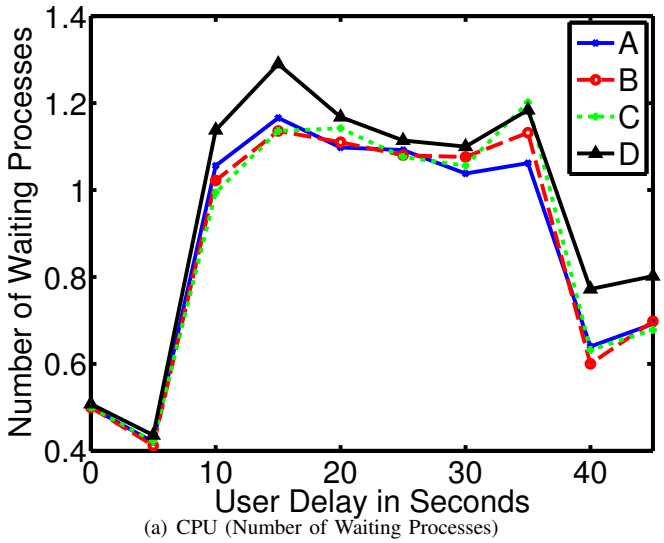


(b) Memory (Bytes)

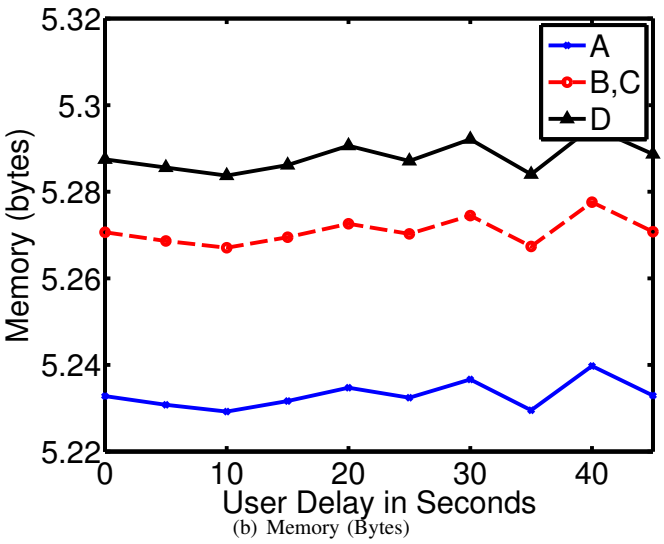


(c) Duration (Milliseconds)

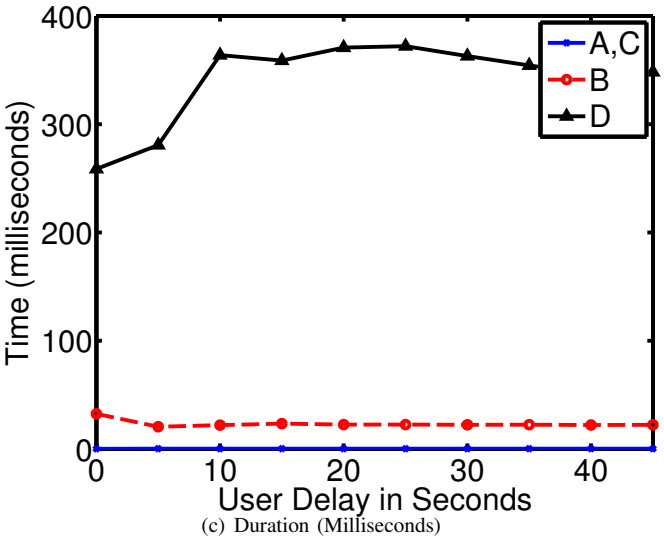
Fig. 8: Average Resource Consumption per Observation in Test 1 (Dotted Lines Represent Error Curves for the Respective Trending Line Shown in Black Line)



(a) CPU (Number of Waiting Processes)



(b) Memory (Bytes)



(c) Duration (Milliseconds)

Fig. 9: Average Resource Consumption for Scenarios A-D (Test1)

TABLE VIII: Test 1 Results (Time in Milliseconds, CPU Usage in Number of Waiting Processes, Memory Consumption in MB)

Scenario	Metric	Min.	Max.	μ	σ	$95\%_{ile}$	$99\%_{ile}$
A	cpu	0.1	2.7	0.88	0.41	1.65	2.1
	memory	2.52	5.34	5.23	0.39	5.32	5.33
	duration	0.10	1.42	0.22	0.08	0.28	0.35
B	cpu	0.1	2.2	0.88	0.41	1.65	2.1
	memory	4.43	5.34	5.27	0.12	5.32	5.33
	duration	17.8	73.6	23.3	6.63	30	61.8
C	cpu	0.1	2.7	0.88	0.42	1.7	2.25
	memory	4.43	5.34	5.27	0.12	5.32	5.33
	duration	0.11	1.55	0.27	0.12	0.47	0.59
D	cpu	0.2	3.4	0.95	0.43	1.7	2.25
	memory	5.27	5.35	5.29	0.01	5.32	5.34
	duration	199	593	342	60	429	521
One cycle (A-D)	cpu	0.1	2.8	0.90	0.42	1.6	2.2
	memory	5.28	5.35	5.29	0.01	5.32	5.34
	duration	224	618	366	59.6	453	546
E	cpu	0.2	3.1	0.92	0.43	1.7	2.3
	memory	5.27	5.35	5.29	0.01	5.32	5.34
	duration	0.09	0.39	0.21	0.05	0.29	0.32
F	cpu	0.2	2.8	0.92	0.43	1.7	2.2
	memory	5.28	5.35	5.29	0.01	5.32	5.34
	duration	8.74	81.9	13.4	9.33	14.7	65.6

1) Memory consumption: In both tests, memory consumption was about 5MB, which was within the predefined baseline of 8 MB. This might indicate that memory consumption was limited for PHP script execution.

2) CPU consumption (measured as number of waiting processes): Table IX shows that maximum CPU usage was slightly over the baseline when performing our tests on the larger dataset D2. The result of more than four waiting processes was achieved in scenario D after adding 28029 software products to our data store. However, in 99 percent of all experiments, CPU usage was below 4 waiting processes.

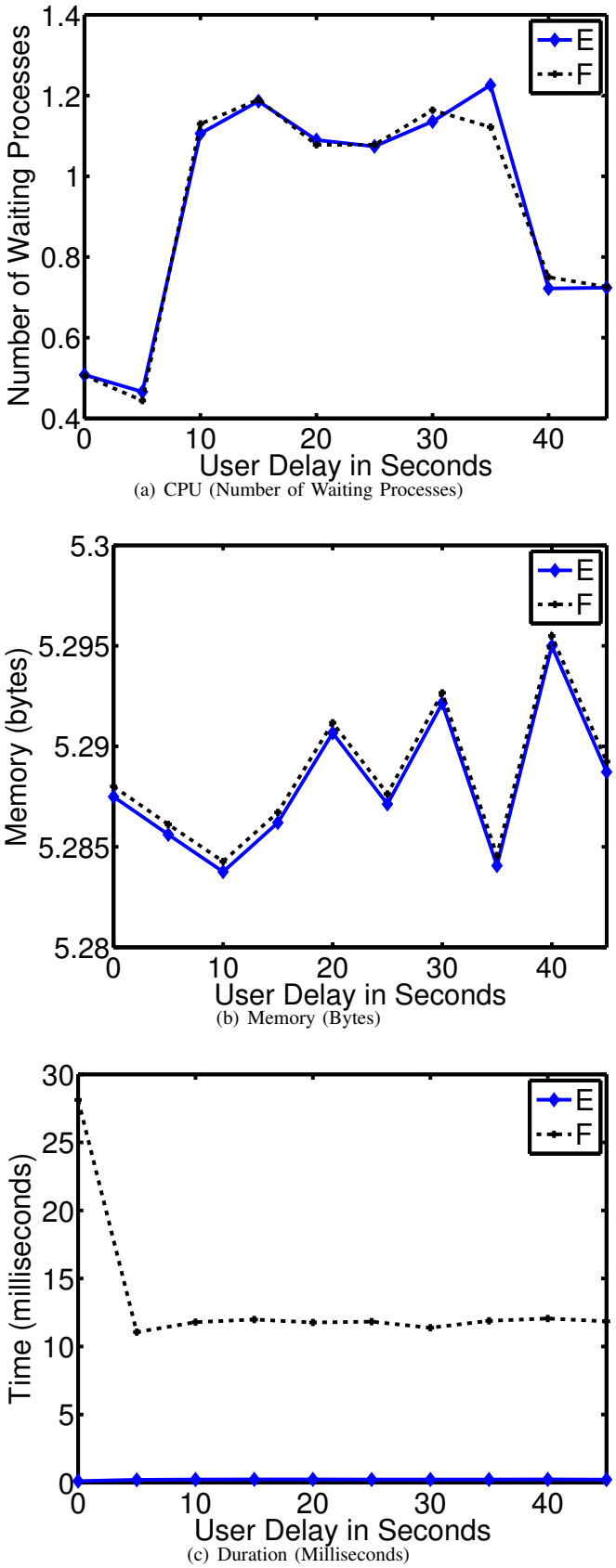
3) Response time (duration): Figure 10 and Table VIII demonstrate that scenario F (SPARQL queries) takes in average more than 10 milliseconds when compared with scenario E (MySQL queries). It is important to note that the duration values in the Scenario F are less than 1 millisecond, which is reflected in the graph in Figure 10(c), which is almost overlapping with the X axis for all user delays tested. Moreover, when adding more software products in Test 2, duration was constantly increasing for scenarios B (storing product description into the triple store) and D (executing the software selection function), as seen in Figure 12.

As seen in Figure 11 the duration time increased as we added more products into the database. As seen in Table IX, while using triple store database, the test results showed a greater response time compared to MySQL transactions.

In Test 2, response times for scenarios B and D were in average about 491 and 2820 milliseconds, while scenarios A and C showed response times below one millisecond in average.

Test 2 showed that the overall duration of one cycle became more than 5 seconds, due to the addition of around 26342 software descriptions into the data storage.

As seen from the 85th percentile statistics in scenario D, duration time is below the baseline value of 5 seconds in 85/cases. When looking at the 99th percentile statistics, duration of scenario D increased up to 5.5 seconds, above the baseline value and resulting in longer execution cycles of more than 10 seconds in some observations. Similarly, 85th

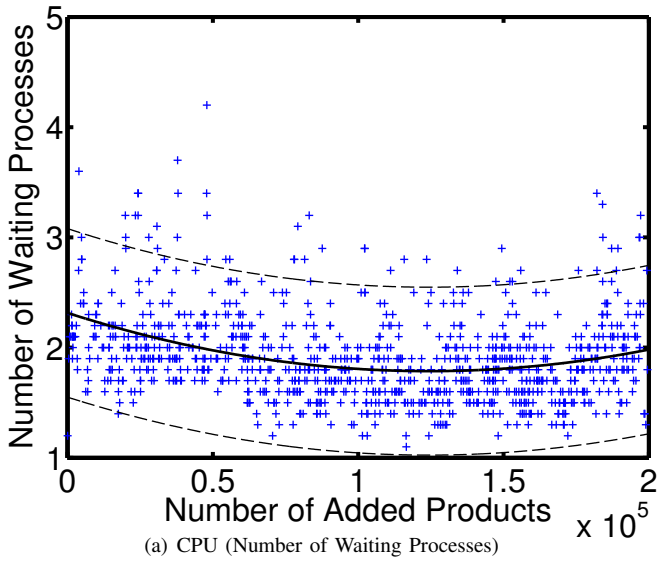


(a) CPU (Number of Waiting Processes)

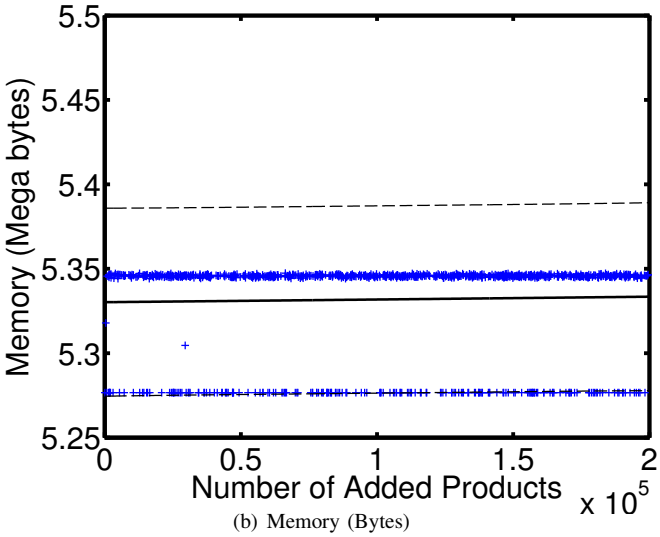
(b) Memory (Bytes)

(c) Duration (Milliseconds)

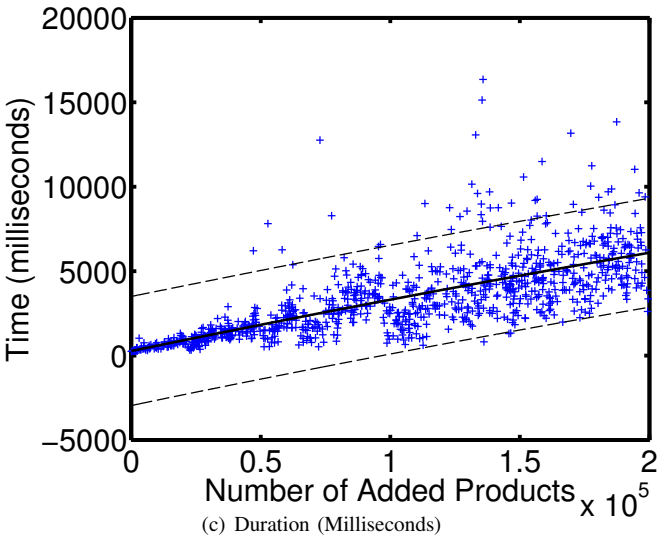
Fig. 10: Average Resource Consumption in E and F Scenarios (Test 1)



(a) CPU (Number of Waiting Processes)

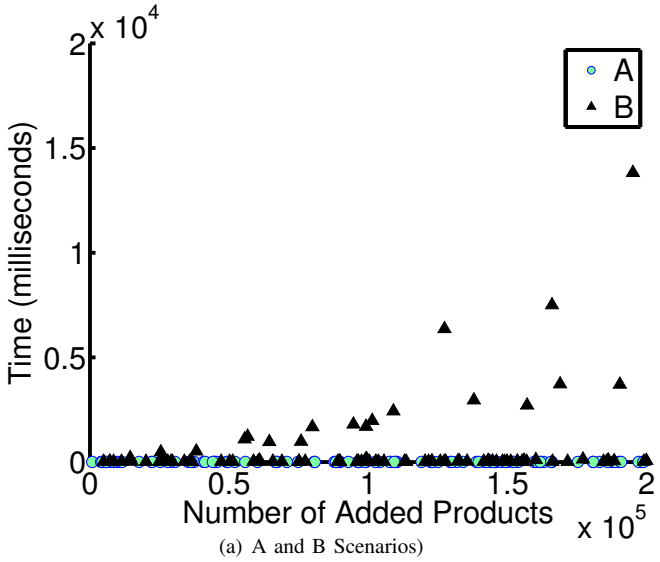


(b) Memory (Bytes)

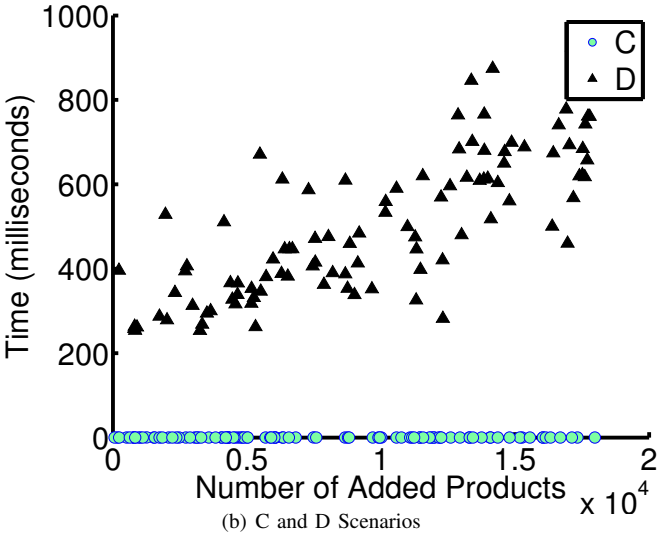


(c) Duration (Milliseconds)

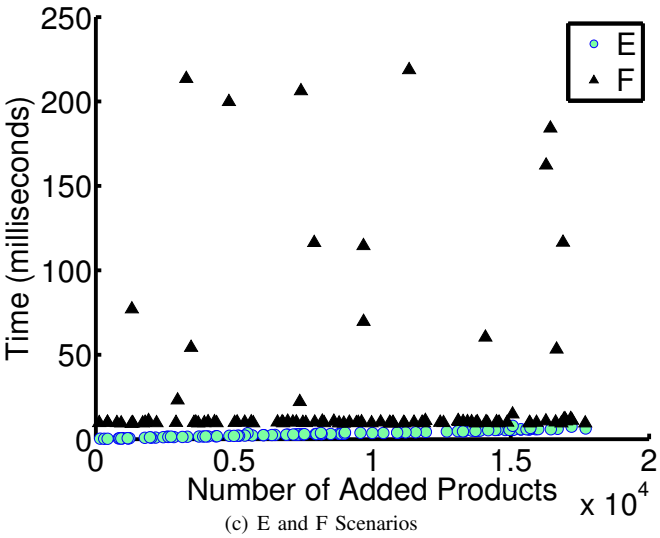
Fig. 11: System Resources Consumption While Adding Products, in 100 Randomly Selected Observations (Test 2)



(a) A and B Scenarios)



(b) C and D Scenarios



(c) E and F Scenarios

Fig. 12: Execution Time While Adding Products, in 100 Randomly Selected Observations (Test 2)

TABLE IX: Test 2 Results for All User Delays (Time in Milliseconds, CPU Usage in Number of Waiting Processes, Memory Consumption in MB)

Scenario	Metric	Min.	Max.	μ	σ	$85\%_{ile}$	$99\%_{ile}$
A	cpu	0.7	4.6	1.91	0.41	2.3	3.3
	memory	2.52	5.35	5.33	0.03	5.35	5.35
	duration	0.08	132	0.14	0.52	0.153	0.23
B	cpu	0.7	4.6	1.91	0.41	2.3	3.3
	memory	4.43	5.35	5.33	0.03	5.35	5.35
	duration	18.2	19300	491	1270	821	6290
C	cpu	0.7	4.6	1.91	0.409	2.3	3.3
	memory	4.43	5.35	5.33	0.03	5.35	5.35
	duration	0.10	195	0.22	1.33	0.2	0.73
D	cpu	0.7	4.6	1.92	0.41	2.3	3.3
	memory	5.27	5.35	5.33	0.03	5.35	5.35
	duration	222	23400	2820	1780	4800	7370
One cycle (A-D))	cpu	0.7	4.6	1.91	0.41	2.3	3.3
	memory	5.28	5.35	5.33	0.03	5.35	5.35
	duration	244	25000	3320	2370	5570	11100
E	cpu	0.7	4.6	1.92	0.41	2.3	3.3
	memory	5.28	5.35	5.33	0.03	5.35	5.35
	duration	0.08	113	37.3	21.5	63.3	75.9
F	cpu	0.7	4.6	1.91	0.41	2.3	3.3
	memory	5.28	5.35	5.33	0.03	5.35	5.3
	duration	8.79	1190	40.4	74.6	77.3	349

percentile of duration is less than one second, while 99th percentile is about 6 seconds in scenario B.

In both tests we reported that minimum duration time in Scenario F more than 8 milliseconds compared to Scenario E. In Test 1, average values of Scenario E and F were around 0.2 and 13.4 milliseconds respectively, while in majority (99/Scenario F was less than 65.7 milliseconds, and duration in Scenario E was less than 0.33 milliseconds. In Test 2 we observed that the 99th percentile of duration in Scenario F is higher than the similar value in Scenario E, and, maximum duration time in Scenario F was more than 10 times longer than the maximum duration time in scenario E, indicating that MySQL queries outperform SPARQL queries.

Overall, Table IX shows that the maximum response time for scenarios B and D could reach up to more than 19 and 23 seconds respectively, while memory and CPU consumption are within the predefined baseline for all scenarios. Therefore, we might suggest that using RDF triple store requires more processing power compared with MySQL storage when adding large amounts of data and the software selection algorithm requires further optimization.

V. Discussion

A. Semantic Web adoption

The Semantic Web application was created to demonstrate the proposed approach using Semantic Web tools. However, difficulties in applying new technologies uncovered a new problem area. There are not yet well-tested and robust best-practices for Semantic Web development. This might have an influence on the development of Semantic Web solutions and as a consequence restrict the adoption of semantic technologies in practice.

Moreover, the most noticeable advantages provided by the Semantic Web are cross-platform data integration and knowledge management. A machine-readable format could be used for data processing, and external web resources could be used for enriching the available data. This was the reason of choosing SW tools for describing and managing information on software products. However, performance limitations of the SW tools employed should be considered and optimized in case of large datasets. When SW storages are built on relational databases, semantic queries are transformed into relational queries, leading to performance penalties [37]. As was demonstrated by our tests, the imposed overheads of working with triple store requires additional processing power and might require more execution time. List sorting [38], optimization methods [39] and SPARQL-to-SQL transformation [40] can be applied for improving the efficiency of information retrieval. Underlying RD tables can also be restructured in order to improve query performance [40]. Alternatively, we might suggest to employ this approach for selecting specialised software in certain domains. This would result in smaller data storage and quicker response times when dealing with triple store.

Furthermore, semantic capabilities can help in the information exchange between software providers and their customers working with different machines. Applying open formats such as Extensible Markup Language and RDF, rather than being dependable on particular proprietary formats can assist in a more efficient communication. Platform-independent and semantically tagged information provided in a well-structured form can help in integrating data from different sources. Further, it could be interesting to extend our prototype's knowledge base with information on the purpose of software products, their application area and customer satisfaction. OWL capabilities for building such a knowledge base, enriching it with new information from external web databases, and machine inference are to be further investigated for creating a more intelligent software selection.

B. Implications on IT infrastructure management

Software products' descriptions provided in a machine-readable format can be semantically queried. With access to a distributed database with software product information, for instance over an easy-to-use interface, software customers would be more independent from third-party advice which is unfortunately not always objective. Overall, IT environments can benefit from decision support when information on pre-installed software products and IT requirements at place is available. This way, solutions which are well-suited to IT infrastructures can be chosen, such that deployment costs and efforts can be minimized.

Moreover, feedback on the performance of purchased solutions can also be considered as providing response to software providers [10]. Such feedback on software applications can be used for improving the product quality and customer satisfaction, while returning customers benefit to business earnings [41]. It seems therefore also rational to suggest a close integration of Internet resources providing software information with IT infrastructures, sharing information on IT assets at place and required software products.

Implications for business security however are to be investigated when dealing with sensitive information.

C. Human involvement and Feedback

Nevertheless, the investigation showed that modern semantic tools can be applied for searching software and

its selection. However, the decision making on software acquisition should be done with caution. As some of the questionnaire participants stated, the responsibility of taking decisions should not be taken away from human users.

Furthermore, attributes of software ontology include functionality and characteristics for assessing software quality. Software quality characteristics can be used for assessing customer satisfaction and assessing software alternatives [42]. It might be reasonable to use simple metrics and graphs, since weighted index values can be influenced by contrasting groups of assessors [42]. The prototype could be further extended to collect user satisfaction. For instance, user feedback can be derived from social media services such as Twitter. in this context, sentiment analysis could be exploited to mine user satisfaction with software products.

D. Interdependencies between selection criteria

For the prototype, the software product scoring algorithm working with software quality characteristics did not consider possible interrelations between software traits, such as dependence between usability and efficiency [43]. For dealing with interdependence between various attributes, we might consider MCDM methods such as ANP as described in [12]. Further research may investigate possible relationships between software quality traits. A better knowledge of software quality traits would help in establishing more accurate weights for the quality criteria used in rating software products.

E. Flexibility of adding new selection attributes

One of the benefits of the proposed approach is the flexibility of adding new selection attributes. This comes from the RDF structure, which allows easy changes in records describing software products as mentioned above. For instance, we might consider to add platform-related system requirements or other software deployment-related information into the set of attributes proposed in [11].

Furthermore, as was suggested in [44], semantic web solutions can be used for developing a collaboratively-shared common domain ontology, which is machine-readable and has reasoning capabilities. As we pointed out above and in [10], OWL could be employed when more sophisticated reasoning is needed. Software selection rules described in OWL could enable an expert-system functionality as in [11]. Moreover, for creating a shared domain ontology, we could help in the information exchange between software purchasers and developers.

VI. Conclusions

In the foregoing, we analyzed factors playing an important role in the software selection process. Fitness to business needs, easiness of integration into existing infrastructures and a well-matching functionality are the most prominent factors to consider while selecting software products, in accord with the survey results. A majority of survey participants agreed in that an automated solution of software selection support is needed. Decision support software could thereby help in the analysis of software characteristics and thus potentially decrease deployment costs and efforts.

To facilitate software traits/requirements exchange between different consumers and platforms, we proposed to employ semantic tools. The developed prototype demonstrated a possible solution using RDF for representing software information, SPARQL for querying stored data, and a multi-criteria decision algorithm for scoring software products in accord to user requirements. Since the semantics' addition resulted in an overhead in terms of resource consumption, we suggested to further optimize operations for storing and querying the database. Nevertheless, the proposed approach requires further analysis due to the undeniable benefits of platform independence, flexibility of storage structures and locations. The proposed approach opens new horizons for improving IT infrastructures in practice and helping software development businesses to market and deliver their products.

In future work, we will investigate query/data source optimization options, possible web sources to be used for mining software-related information and automatically creating software product descriptions. We aim at exploiting web content and social media streams to gather information on software products, their properties, requirements and user feedback.

References

[1] J. Dedrick, V. Gurbaxani, and K. Kraemer, "Information technology and economic performance: A critical review of the empirical evidence," *ACM Computing Surveys*, vol. 35, no. 1, 2003, pp. 1–28.

[2] J. Benamati, A. Lederer *et al.*, "An empirical study of it management and rapid it change," in *Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research*. ACM, 1999, pp. 144–153.

[3] W. Novak, J. Cohen, A. Lattanze, L. Levine, P. Place, R. Williams, and C. Woody, "Software acquisition planning guidelines," 2005.

[4] R. N. Charette, "Why software fails [software failure]," *Spectrum, IEEE*, vol. 42, no. 9, 2005, pp. 42–49.

[5] Dell (2008, June). Essentials of software asset management. policies for software evaluation, purchasing, usage a& compliance monitoring. [Online]. Available: https://portal.asap.com/en-CA/Documents/Essential_Elements_of_SAM_200806.pdf

[6] Distributed Management Task Force (2009, May). Common information model (cim) infrastructure. [Online] Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP0004_2.5.0.pdf

[7] Microsoft Corporation (2009, May). Microsoft operations frameworkmicrosoft operations framework. cross reference itil v3 and mof 4.0. [Online] Available: http://goo.gl/M7PL1N

[8] International Organization for Standardization (2012, June June). Iso/iec 19770-1:2012 information technology – software asset management – part 1: Processes and tiered assessment of conformance. [Online] Available: http://goo.gl/qjxXoF

[9] P. Ulkuniemi and V. Seppanen, "Cots component acquisition in an emerging market," *Software, IEEE*, vol. 21, no. 6, 2004, pp. 76–82.

[10] E. Ilina, "A new approach of software asset acquisition using semantic web," Master's thesis, University of Liverpool, 2010.

[11] K. Eldrandaly and S. Naguib, "A knowledge-based system for gis software selection." *International Arab Journal of Information Technology*, vol. 10, no. 2, 2013, pp. 152–159.

[12] Z. Ayağ and R. Özdemir, "An intelligent approach to erp software selection through fuzzy anp," *International Journal of Production Research*, vol. 45, no. 10, pp. 2169–2194, 2007.

[13] S. Onut and T. Efendigil, "A theorical model design for erp software selection process under the constraints of cost and quality: A fuzzy approach," *Journal of Intelligent and Fuzzy Systems*, vol. 21, no. 6, 2010, pp. 365–378.

[14] K. Yuen and H. Lau, "Software vendor selection using fuzzy analytic hierarchy process with iso/iec9126," *IAENG International journal of computer science*, vol. 35, no. 3, 2008, pp. 267–274.

[15] Z. Ayağ, "Evaluating simulation software alternatives through anp," in *Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia*, 2011.

[16] H. Deng and S. Wibowo, "Intelligent decision support for evaluating and selecting information systems projects." *Engineering Letters*, vol. 16, no. 3, pp. 412–418, 2008.
[17] Carnegie Mellon University (2002, March). Software acquisition capability maturity model (sa-cmm) version 1.03. [Online] Available: http://www.sei.cmu.edu/reports/02tr010.pdf
[18] X. Franch and J. Carvallo, "A quality-model-based approach for describing and evaluating software packages," in *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*. IEEE, 2002, pp. 104–111.
[19] Runaware Inc. (2013). Runaware — On-demand Demos Of Your Software. [Online] Available: http://main.boston.runaware.com/
[20] Tucows Downloads (2013). About tucows downloads. [Online] Available: http://www.tucows.com/about.html
[21] Technology Evaluation Centers Inc. (2013). Tec helps you choose the best enterprise software solutions for your organization. [Online] Available: http://www.technologyevaluation.com/
[22] O. Hauge, T. Osterlie, C. Sorensen, and M. Gerea, "An empirical study on selection of open source software-preliminary results," in *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS'09. ICSE Workshop on*. IEEE, 2009, pp. 42–47.
[23] J. Brunner, L. Ma, C. Wang, L. Zhang, D. Wolfson, Y. Pan, and K. Srinivas, "Explorations in the use of semantic web technologies for product information management," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 747–756.
[24] M. Grobe, "Rdf, jena, sparql and the'semantic web'," in *Proceedings of the 37th annual ACM SIGUCCS fall conference*. ACM, 2009, pp. 131–138.
[25] S. Kulkarni and D. Caragea, "Towards bridging the web and the semantic web," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2009, pp. 667–674.
[26] B. Antunes, N. Seco, and P. Gomes, "Knowledge management using semantic web technologies: An application in software development," in *Proceedings of the 4th international conference on Knowledge capture*. ACM, 2007, pp. 187–188.
[27] J. Tane, C. Schmitz, and G. Stumme, "Semantic resource management for the web: an e-learning application," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004, pp. 1–10.
[28] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology." New York, USA: ME Sharpe, 2003, pp. 331–339.
[29] E. Ilina (2010, August). Feedback to the software selection survey's report. [Online] Available: http://goo.gl/Y4h7rt
[30] W. Orlikowski and J. Baroudi, "Is research paradigms: method versus substance," 1989.
[31] E. Ilina. (2010, August). Software selection process: survey. [Online] Available: http://softologics.com/survey/report/index.htm
[32] E. Prud'hommeaux (2013). Sparql vs. sql - intro. [Online] Available: http://www.cambridgesemantics.com/semantic-university/sparql-vs-sql-intro
[33] semsol (2013, July). semsol/arc2 github. [Online] Available: https://github.com/semsol/arc2
[34] P. Hitzler, S. Rudolph, and M. Krötzsch, *Foundations of semantic web technologies*. Chapman & Hall/CRC, 2009.
[35] D. L. Olson, B. Johansson, and R. A. Carvalho, "A combined method for evaluating criteria when selecting erp systems," *Re-conceptualizing Enterprise Information Systems*, 2012, pp. 64–74.
[36] M. Alkhawlani and A. Ayesh, "Access network selection based on fuzzy logic and genetic algorithms," *Advances in Artificial Intelligence*, vol. 8, no. 1, p. 1, 2008.
[37] D. Abadi, A. Marcus, S. Madden, and K. Hollenbach, "Sw-store: a vertically partitioned dbms for semantic web data management," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 18, no. 2, 2009, pp. 385–406.
[38] S. Groppe and J. Groppe, "External sorting for index construction of large semantic web databases," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 1373–1380.
[39] J. Groppe, S. Groppe, S. Ebers, and V. Linnemann, "Efficient processing of sparql joins in memory by dynamically restricting triple patterns," in *ACM SAC*, 2009.
[40] B. Elliott, E. Cheng, C. Thomas-Ogbuji, and Z. Ozsoyoglu, "A complete translation from sparql into efficient sql," in *Proceedings of the 2009 International Database Engineering & Applications Symposium*. ACM, 2009, pp. 31–42.
[41] S. Jansen and S. Brinkkemper, "Evaluating the release, delivery, and deployment processes of eight large product software vendors applying the customer configuration update model," in *Proceedings of the 2006 international workshop on Workshop on interdisciplinary software engineering research*. ACM, 2006, pp. 65–68.
[42] S. Kan, *Metrics and models in software quality engineering*. Addison-Wesley, London., 2002.
[43] B. Jayaswal and P. Patton (2006). Software quality metrics. [Online] Available: http://www.developer.com/tech/article.php/3644656/Software-Quality-Metrics.htm
[44] J. Malczewski and M. Jelokhani-Niaraki, "A web 3.0-driven collaborative multicriteria spatial decision support system," *Cybergeo: European Journal of Geography*, 2012.

1) Date of modification: 5th December 2013

2) Minor changes in Table I and added reference to the table V, page 3.