Goal of the Experiment

Compare two search engines to find relevant software architectural information on Stack-Overflow. Both search engines are classical keyword search engines. During the experiment, you will not know, which search engines are we comparing.

Experimental Procedure

Software architecture design tasks has a big scope, which involve functional and non-functional requirements, as well as constraints. You will be asked to search for architectural information on Stack Overflow to solve six different design tasks. For each task, you will use only one of two search engines. During your searching, you should record the searching queries (keywords), and the list of relevant Stack Overflow posts and their relevance to each task.

The relevance of each post (in the list of posts identified by the two search engines) to help complete the task is defined on four levels:

- **Highly Relevant** (H): The post addresses a similar problem to the task and contains useful information. The post provides an answer to the searching goal, and fulfills at least one requirement of the task.
- **Medium Relevant** (M): The post addresses another problem not similar to the task at hand, but it provides some relevant information to the task, which could be an answer to the searching goal. Nevertheless, the provided information does not consider specifically the task's requirements.
- **Low Relevance** (L): The post contains relevant information, which is only remotely relevant to solving the given task, but might help for refining the search.
- **No Relevance** (N): The post has nothing to do with the task. It has no relevant information.

When completing the tasks, you can follow several steps:

1) *Read task*: Read the task carefully to understand the requirements and the goals of the search.
2) *Log in to system*: Log in to Decision Buddy using the assigned user name and password. Please make sure that you are still logged when moving from one task to the next. The login link, user name and password are written down in each task.
3) *Access search engine*: Open the search engine given for a task. Each task has a link to the assigned search engine. Please be careful to choose the right search engine. Make sure not to use the search engine from the previous task. You will use only 1 search engine per task.
4) *Conduct search*: Start searching for the relevant posts and information that could help perform a given task. During the search:
   - Try to find as many posts as possible relevant to the task. *Submit **at least four queries** for each task. For each query, **assess only the top 10 posts***, which are returned from the search engine.
   - During assessment, **read the post carefully** to make sure that it is relevant to the problem, and do not assess the relevance just based on the title of a post. The relevance assesses the relevance between the post and the task (not the query).

- **Do not** use any other sources of information (e.g. Google or Vendor websites) other than the assigned search engine to solve the tasks.

- Solve the tasks in their provided **sequence**.

- Try to **execute different queries** to cover all the aspects of the task (e.g. different requirements, technologies, constraints).

- **Close the browser or the tab after each task** to prevent using the wrong search engine for the following tasks.

5) *Record result*: Record your results in the attached Excel file. Record the following for each task:

- *Column "Query String (Keywords used)":* record at least five queries used for each task, even queries which didn't return any relevant posts. (See snap-shot)

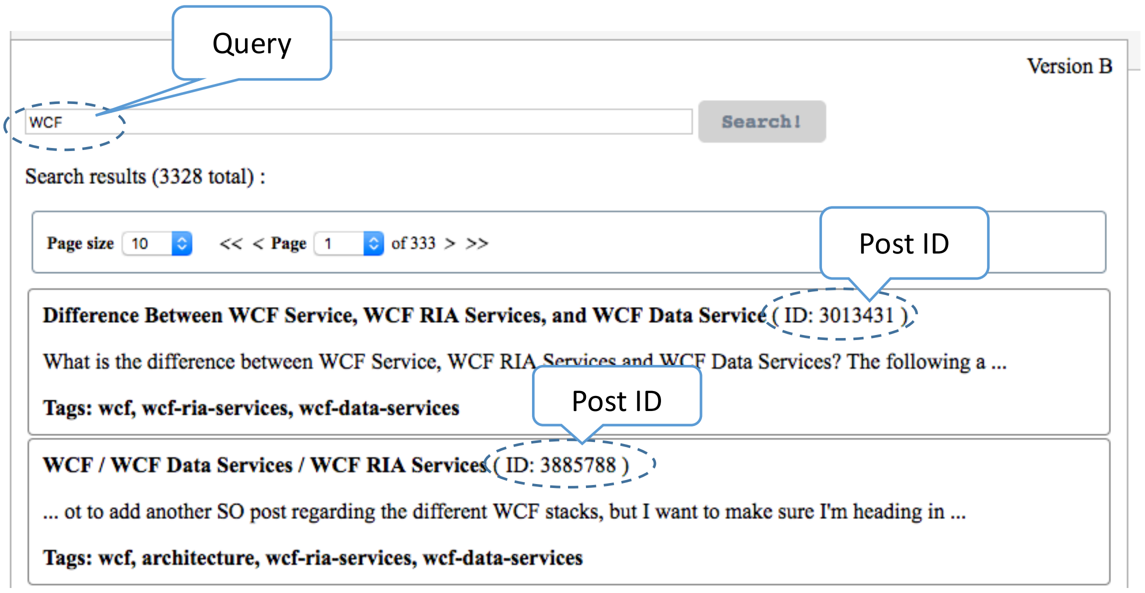
- *Column "Post ID's" of relevant Stack Overflow posts* and their *relevance* level (High, Medium, Low) in column "Post Relevance". Assess only the top 10 posts per query. You do not need to record posts which have no relevance to the task. Record relevant post only one time for each task.

- *Brief explanation of why the post is relevant to the task in column "Relevance Explanation"*. You could copy parts of the post as an explanation, or write brief explanations yourself.

- Answer the question in the Excel sheet regarding the complexity of the task.

6) *Fill Survey*: Complete this questionnaire: https://goo.gl/forms/UIfekXRKHh9lmyA02

Make sure to record your information at the right excel tab. Each task has an ID in its title, as well as in the excel sheet.



Search results (3328 total) :

Difference Between WCF Service, WCF RIA Services, and WCF Data Service (ID: 3013431)

What is the difference between WCF Service, WCF RIA Services and WCF Data Services? The following a ...

Tags: wcf, wcf-ria-services, wcf-data-services

WCF / WCF Data Services / WCF RIA Services (ID: 3885788)

... ot to add another SO post regarding the different WCF stacks, but I want to make sure I'm heading in ...

Tags: wcf, architecture, wcf-ria-services, wcf-data-services

Task ID: Big-Data-Stream-Evaluation

Design Scenario

An internet company provides popular content and online services to millions of web users. Besides providing information to external users, the company collects and analyzes *massive logs* of data that are generated from its infrastructure (e.g. server logs). The size of the log files is in *Tera-bytes*. To cope with the fast infrastructure growth, the company decided to develop a software application to manage the logs. A high level conceptual model has been designed, which consists of a data stream, which sends its data to a batch layer and a real time view. The data stream component dispatches data from multiple data sources in real-time. The architects of the system are discussing the possible technology choices for implementing the data stream component. Three technology families are identified as alternative architectural solutions:

1) Data collector technologies (e.g. Apache Flume, Fluentd)
2) Distributed message broker technologies (e.g. Apache Kafka, Amazon SQS, Active MQ)
3) ETL/Data Integration engines (e.g. Streamsets, Talend)

Non-functional requirements

- *Performance*: The system shall collect up to 15,000 events/second from web servers.
- *Extensibility*: The system shall support adding new data sources by just updating a configuration file.
- *Availability*: The system shall continue operating with no downtime if any single node or component fails.
- *Deployability*: The system deployment procedure shall be fully automated and support a number of environments (development, test, production).

Constraints

The system shall be composed of primary open source technologies (for cost reasons).

Search goal

The architect would like to compare the three technology families regarding their suitability to the described scenario, non-functional requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/d

Task ID: Physical-Design

Design Scenario

An IT service company builds business-to-business solutions, where heterogeneous software systems located in different locations are integrated with each other. The solution architect is working on a project, where several Customer Relationship Management (CRM) applications communicate with other software systems, which include legacy systems used in call centers, and banking systems (e.g. ATM). The CRM applications use both Microsoft and Oracle technologies. The solution architect is currently designing the integration layer, which would facilitate the communication between CRM apps and other software systems. The architect decided on Apache Camel and RabbitMQ as possible integration technologies. The selection of both technologies raises two architectural concerns:

1) Selecting a mechanism for message channeling, translation and routing.
2) Establishing a deployment topology (physical architecture).

Non-functional requirements

- *Availability*: ATM machines need high availability with no down time.
- *Performance*: The integration layer should be prepared to receive 10,000 request/sec from the CRMs.

Constraints

The company has an official agreement with Oracle for Unix servers.

Search goal

The architect would like to search for possible information on *technology features, and components design* which would help him address the aforementioned concerns.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/e

Task ID: Conceptual-Design

Design Scenario

An online shop wants to modularize its Java web applications and expose selected components and services via APIs to external partners (e.g., marketing firms, suppliers, price comparison websites) over the internet. The lead architect of the microservices project that is in charge of this Web API design effort tries to answer the following questions:

1) What is the right service decomposition?
2) How can loose coupling and high cohesion be achieved?
3) Should communication be synchronous or asynchronous?

The architect has already read some books and articles, which (s)he found interesting but insufficient (e.g., too abstract to be actionable on the project). The architect is looking for patterns, components design and design principles for service decomposition.

Non-functional requirements

- Data consistency.
- Versioning.
- Quality of service (API security, service level agreements, performance).

Constraints

Any conceptual component that is found and any patterns should be mature, e.g., implemented in at least two different settings. Best practices, should be either applied in Java or Microsoft technologies.

Search goal

The architect would like to search for possible *architectural principles, patterns, and components design* which would help him answer the aforementioned questions.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/e

Task ID: Middleware-Search

A stock monitoring dashboard software needs to be developed. The dashboard will be available on the internet, and could be accessed by users through mobile devices. An important aspect of the dashboard is the *real-time* change of stock values. Stock information is gathered from various sources and then transformed to be presented on the dashboard. The architect is searching for information about suitable middleware technologies, which could gather and transform the stock information.

Non-functional requirements

- *Performance*: The stock information needs to be updated in real time.
- *Scalability*: The system needs to scale to more than 100,000 users.
- *Security*: The stock information need to be securely transferred from their sources to the dashboard.

Constraints

Technologies need to be based on standard implementations with defined and published specifications.

Search goal

The architect would like to search for suitable *middleware technologies* which would fulfil the aforementioned requirements and constraints.

Find posts, which could support the architect fulfilling their request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b

Task ID: JSON-Search

Design Scenario

A claim management system in an insurance company is currently being modernized; a cross-platform iOS and Android mobile app has been introduced recently. This app needs to communicate with the claim management system backend (which is a JEE application hosted on an application server with IBM WebSphere) via *RESTful HTTP*; *JSON* has been decided as message exchange format. The integration architect, who is responsible for the claim management system backend and its RESTful HTTP interface now looks for *JSON parsing libraries* that come with the JEE application server (WebSphere), but also additional libraries, which could be independent from the webserver used.

Non-functional requirements

As many claims are filed and processed via this interface, the *performance of unmarshalling* (e.g., conversion from JSON to Java classes) is an important architecturally significant requirement on the server side.

Constraints

Not all open source licenses can be used: Apache 2 and Eclipse have been approved for use, GPL has been banned, all others require approval from higher-level management.

Search goal

The architect would like to search for possible *JSON parsers* for Java, which would satisfy the requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b

Task ID: Messaging-Evaluation

Design Scenario

A help desk system is to be produced; it dispatches incoming chat messages to human agents. To save costs, a chat bot is currently being designed to replace some of these agents without compromising user satisfaction. It has been decided to implement an *update notification mechanism* in the scenario. The chat bot implementation has to be integrated with a Natural Language Processing (NLP) system and an Architectural Knowledge Base (AKB). *Asynchronous messaging* has been selected as implementation pattern for the integration channel. A *publish-subscribe* channel should be realized; this channel should be supervised and managed by using one or more systems management patterns. Three messaging technologies have been identified as candidates: 1) RabbitMQ, 2) Apache Kafka, and 3) ActiveMQ

Non-functional requirements

Guaranteed delivery of messages, high throughput and low latency are three important quality attributes. Using standard protocols and formats are required to ensure portability and interoperability.

Constraints

To avoid vendor lock-in, the chosen technology should be implemented and supported by at least three vendors. The learning curve of the technology should be in the hours-to-days range, not in the weeks-to-months range or even higher. Preferably, Java should be supported, but Python, PHP and Scala would also be acceptable.

Search goal

The architect would like to compare the three technologies regarding their suitability to the described scenario, non-functional requirements and constraints.

Find posts, which could support the architect fulfilling his request.

To login, use this link:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/login

user name: user14

password: user14

Use this link to access the search engine:

http://swkvm01.informatik.uni-hamburg.de/DecisionBuddy_MAGeorg/addsearch/b