Design Assistant for NoSQL Technology Selection

John Klein and Ian Gorton
Software Engineering Institute at Carnegie Mellon University
Pittsburgh, PA, USA
{jklein, igorton}@sei.cmu.edu

ABSTRACT
NoSQL databases create tight coupling between data model, deployment topology, and application architecture, and so this technology selection must be one of the earliest architecture decisions. The NoSQL technology landscape is large and evolving rapidly, so architects need efficient and trusted design assistance to explore the solution space. Our solution was to create a queryable knowledge base, populated with curated information, which is rendered dynamically as content grows and changes. We built the knowledge base using Semantic MediaWiki, implementing a novel knowledge model that enables reasoning from quality attributes to architecture patterns and tactics to features implemented in specific NoSQL products. We also provide tabular and graphical visualizations to support both systematic and ad hoc exploration. Our contributions to the field of architecture design assistants are the knowledge model, its implementation in a semantic platform, and the resulting populated knowledge base for big data system architects.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *computer-aided software engineering*. D.2.2 [**Software Engineering**]: Software Architectures – domain-specific architectures, patterns.

General Terms
Design.

Keywords
Design assistant; knowledge base; quality attribute; pattern, architecture tactic; NoSQL; feature model

1. INTRODUCTION
Software architects frequently must choose specific commercial off-the-shelf (COTS) products to realize particular elements in the architecture. Best practices are to design an architecture that is technology neutral, and defer the product selection decision until late in the design process [1][2]. Architects designing big data systems often use NoSQL databases, which deliver high performance, elastic storage capacity, and availability by replicating and partitioning data sets across a cluster of servers. Each NoSQL product implements a different data model and query language, as well as specific tactics to achieve distributed data consistency and availability.

FoSADA'15, May 6, 2015, Montreal, QC, Canada.

http://dx.doi.org/10.1145/2751491.2751494

The data, consistency and distribution models imposed by the selected database have a pervasive impact on the design of the associated applications, and so this technology selection decision must be made early and it becomes difficult and expensive to change [3]. Other COTS product selections can also become "architectural decisions", e.g., operating system or middleware. However, as a still-maturing technology, the NoSQL solution space is large and dynamic[1], with new products constantly emerging and existing products releasing several versions per year with ever evolving feature sets.

As we discovered in performing a NoSQL product evaluation for a customer [4], navigating this solution space can be a hard problem for architects. The predominant information sources are product vendors and experience reports published informally on the Internet. In both cases, the quality is highly variable and difficult to assess. Architects, thinking in terms of system level quality attributes, encounter product descriptions framed terms of their features, and the relationship between particular product features and system-level quality attributes is often not clear.

Architecture design assistants, such as ArchE [5], support reasoning from quality attributes to patterns, but do not directly support reasoning all the way through to COTS selection. Other assistants, such as AREL, capture design rationale for reuse [6]. Both types of assistants take a broad, general approach to architecture decisions. At the other extreme, curated COTS comparisons consolidate product feature lists but do not relate features to qualities or provide insight into tradeoffs, and are not easily queried or filtered [7][8].

Our solution, called *QuABaseBD* (Quality At Scale Knowledge Base for Big Data, pronounced *k-base-bee-dee*) provides decision support to architects, in the narrow but important domain of NoSQL product selection. QuABaseBD supports reasoning from general quality attributes, expressed as scenarios, to architecture approaches and tactics, to features implemented by concrete NoSQL products that realize those tactics. QuABaseBD is an interactive assistant, and does not provide automated suggestions or automated decisions.

The main contributions of this work, discussed in the following sections of this paper, are:

- A generalizable architecture knowledge model, instantiated for the domain of NoSQL distributed databases;
- A feature model for NoSQL technology, which aligns with the instantiated knowledge model;
- An implementation of the knowledge base on the Semantic MediaWiki platform, using forms to add knowledge that conforms to the models, query-driven templates dynamically

[1] See, for example, http://blogs.the451group.com/information_management/2014/03/18/updated-data-platforms-landscape-map-february-2014/

render content as the knowledge base grows, and hyperlinked text, tables, and graphics for presentation and navigation.

2. QUABASEBD KNOWLEDGE MODEL

The QuABaseBD semantic knowledge model, show in Figure 1, is divided into two main sections. The first of the sections, shown in the left half of the figure, represents general software architecture design concerns related to quality attributes, quality attribute scenarios, and architecture tactics. The second section of the knowledge model represents the features of a particular COTS product, shown on the right half of the figure. The purpose of this section is to systematically decompose product features using a normalized taxonomy.

The novelty of this model is the linkage between the two sections through the relationship of an instance of a tactic to the instances of the features of a particular database that implement that tactic, enabling reasoning from general architecture principles to concrete COTS implementations.

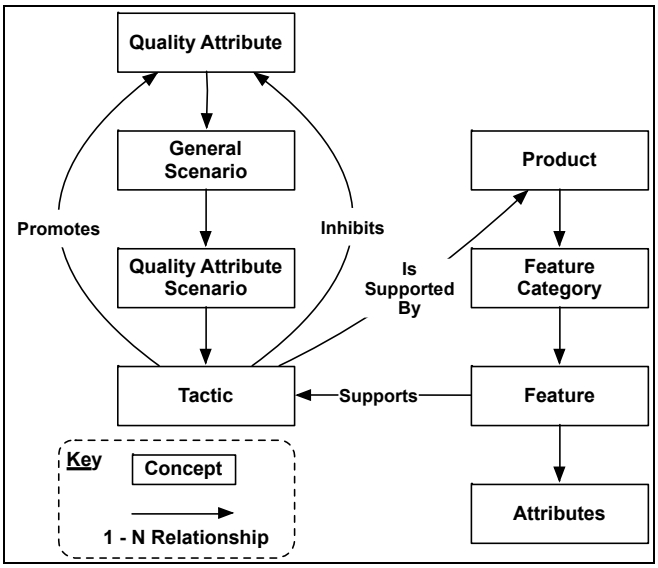


Figure 1. Extract from QuABaseBD Knowledge Model

The left-hand section of the QuABaseBD model is similar to other general architecture knowledge models, such as presented by Babar [9].

While the structure of this section of the knowledge model is very general, in QuABaseBD it is populated only the concepts and properties needed to reason about the domain of big data systems design and database technology selection. The purposes of this section of the model are to support the definition of architecturally significant requirements, to identify the quality attribute tradeoffs that are inherent in distributed data-intensive systems, and to identify relevant tactics to achieve particular architecture requirements.

As such, QuABaseBD is only concerned with the quality attributes of Availability, Consistency, Performance, Scalability, and Security. Each *Quality Attribute*[2] is characterized by a *General Scenario*. The general scenario includes only those stimuli, responses, and response measures that are relevant in big data systems, in contrast to the abstract general scenarios presented by Bass and colleagues [10]. An example of the QuABaseBD General Scenario for Scalability is shown in Table 1.

Table 1. QuABaseBD General Scenario for Scalability

Stimulus:	Increase in load (demand) on a system resource such as processing (OR) I/O (OR) storage.
Environment:	Increase in load is transient (OR) Increase in load is permanent
Response:	System provides new resources to satisfy the load
Response Measure:	Ratio of increase in cost to provide new resources to value of increased load Time to provide additional resources when load increases

A general scenario is a prototype that generates many *Quality Attribute Scenarios*, each of which combines a stimulus and response in the context of a big data system. A Quality Attribute Scenario covers a specific situation, and so we can identify the *Tactics* that can be employed to achieve the desired the scenario response. Table 2 shows a quality attribute scenario derived from the Scalability General Scenario. Note that the stimulus, response, and response measure specialize the general scenario shown above.

Table 2. Scalability Scenario

Scale to handle increased read or write request load	
Quality Attribute:	Scalability
Stimulus:	An increase in read requests is experienced by the system for a finite period of time (e.g typically minutes to days)
Environment:	The system has been operating in production.
Response:	Additional nodes can be added to the cluster and the data set can be repartitioned to use the new resources.
Response Measure:	Downtime during repartitioning, Amount of manual intervention needed.
Tactics:	Automatically maintain cluster membership list (gossip), Shard data set across multiple servers (Consistent Hashing), Shard data set across multiple servers (Range-based), Load balance across replicas (one data center), Load balance across replicas (multiple data centers)
AntiTactics:	None

Tactics represent tradeoffs – each promotes at least one quality attribute, and may inhibit other quality attributes. Although not represented in Figure 1, the knowledge model also includes "anti-tactics", representing design approaches that prevent the desired response from being achieved. In Figure 2, we show a screenshot from QuABaseBD for the Consistent Hashing tactic, referenced in the Scalability Scenario. The tactic definition includes a description that outlines the approach, followed by a table that summarizes the tradeoffs inherent in applying the tactic along with references to related tactics.

Tactics also represent specific design decisions that can be realized by a *Product* implementation, so we can say that a

[2] In this section of the paper, terms in *italics* refer to concepts in the model shown in Figure 1.

database supports a tactic. At the bottom of the screenshot in Figure 2, we see links to the specific NoSQL products that support the Consistent Hashing tactic.

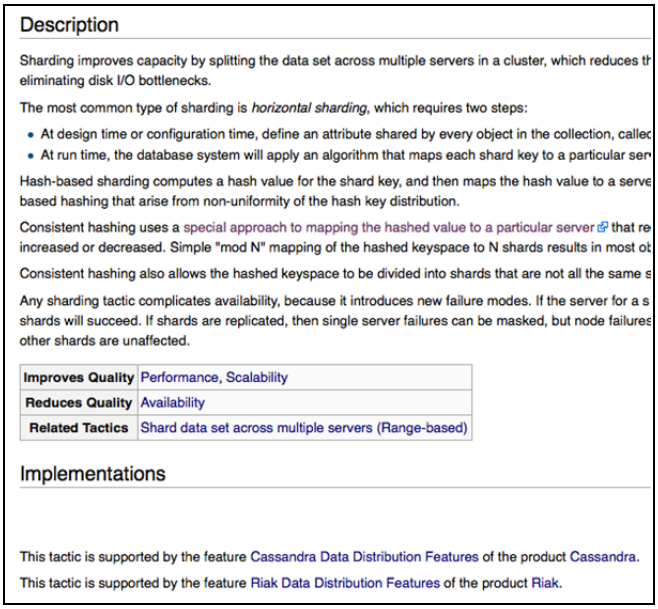


Figure 2. Screenshot of Tactic Page in QuABaseBD

3. NOSQL FEATURE MODEL

The product feature model in QuABaseBD addresses two related concerns:

- It provides the structure and normalized vocabulary (i.e. a taxonomy) to allow product comparisons, and
- It identifies product features that realize the architecture tactics defined in the QuABaseBD Knowledge Model.

The NoSQL feature model was initially based on our work evaluating NoSQL databases [4], and evolved as other products were added to the knowledge base, and as new tactics and tactics were identified and added.

The model has three levels, which are shown on the right side of Figure 1. The first level contained *Feature Categories*, which were identified as dimensions where the COTS product features had significant impact on architecture of the software using the product. The complete NoSQL feature model is described in [11]. Here we summarize the QuABaseBD feature categories Table 3, and show the details of the Data Model feature category in Table 4.

4. SEMANTIC MEDIAWIKI PLATFORM

As depicted in Figure 3, QuABaseBD is presented to a user through a standard Web-based wiki interface. QuABaseBD is built upon the Semantic MediaWiki (SMW) platform (https://semantic-mediawiki.org/), which adds dynamic, semantic capabilities to the base MediaWiki implementation (as used, for example, in Wikipedia).

In contrast to a typical wiki such as Wikipedia, the pages in QuABase are generated dynamically, based on information that users enter into a set of structured forms. These forms, which leverage built-in SMW capabilities, guide and constrain content creation, providing three main benefits:

- Users are guided through the content entry process, which improves usability and efficiency;
- Much of the data entry is constrained, using picklists, radio buttons, checkboxes, and autocompletion, and free text fields can be designated as required, which ensures that the data conforms to the knowledge and feature models.
- When the form is saved, semantic annotation is automatically applied, which enable querying the content.

All QuABaseBD content is rendered as web pages for a user using SMW templates, which extend on the basic MediaWiki template mechanism. The SMW templates are populated by SMW queries, which extract content based on the semantic annotation that was applied by the form when the content was created.

Table 3. QuABaseBD Feature Categories

Feature Category	Description
Data Model	The data model supported by a distributed database dictates both how application data can be organized, and to a large extent, how it can be queried.
Query Language	Characteristics include declarative or imperitive style, key matching options, cursor handling, sorting and filtering of result sets, and programing language bindings.
Consistency	Replica consistency features such as atomic updates, quorums, conflict detection and resolution, and durability; transactional consistency concerns.
Scalability	Features that support concurrent client access, including horizontal partitioning (sharding), replication, request distribution, and write locking strategies.
Data Distribution	Features specific to horizontal distribution (sharding), such as rebalancing strategies and query processing.
Replication	Features supporting replication to improve availability and to improve performance.
Security	Authentication, authorization, encryption features.

Table 4. Features in Data Model Feature Category

Feature	Allowed Values
Data Model	Column, Key-Value, Graph, Document, Object, Relational
Fixed Schema	Required, optional, none
Opaque Data Objects	Required, not required
Hierarchical Data Objects	Supported, not supported
Automatic Primary Key Allocation	Supported, not supported
Composite Keys	Supported, not supported
Secondary Indexes	Supported, not supported
Query by Key Range	Supported, not supported
Query by Partial Key	Supported, not supported
Query by Non-Key Value (Scan)	Supported, not supported
Map Reduce API	Builtin, integration with external framework, not supported
Indexed Text Search	Support in plugin (e.g. Solr), builtin proprietary, not supported

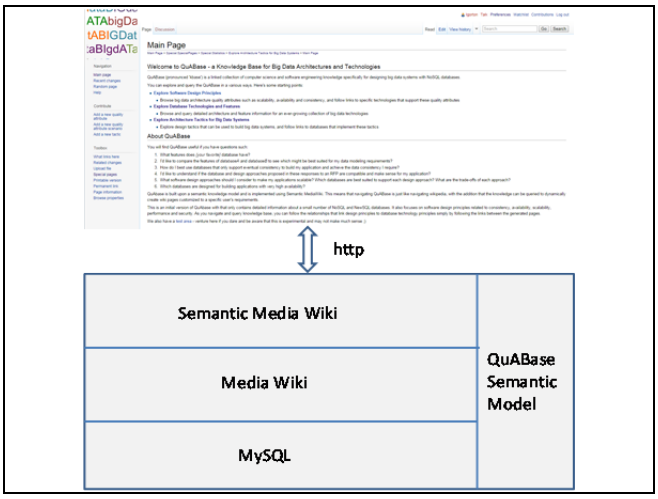
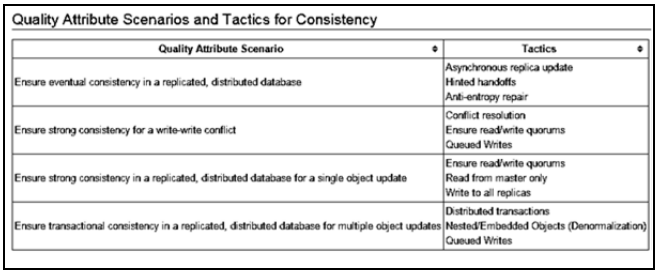


Figure 3. Conceptual Architecture of QuABaseBD

For example, the following SMW query will return a list of all of the Quality Attribute Scenarios associated with a particular Quality Attribute.

```
{{#ask:[[Category:Quality Attribute Scenario]] |
 [[Describes quality attribute::{{PAGENAME}}]]
   |mainlabel=Quality Attribute Scenario
   |?Has Tactic = Tactics
   |order=ASC}}
```

This query is placed in the template that renders the wiki page for Quality Attributes. When this template is invoked, for example, for the Consistency Quality Attribute, it creates the table shown in the screenshot in Figure 4.



Quality Attribute Scenarios and Tactics for Consistency

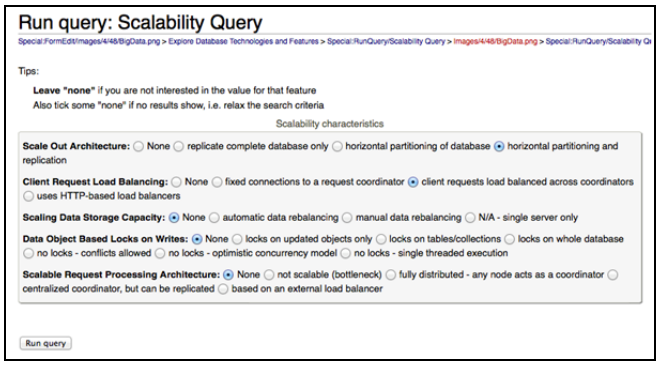
Quality Attribute Scenario	Tactics
Ensure eventual consistency in a replicated, distributed database	Asynchronous replica update Hinted handoffs Anti-entropy repair
Ensure strong consistency for a write-write conflict	Conflict resolution Ensure read/write quorums Queued Writes
Ensure strong consistency in a replicated, distributed database for a single object update	Ensure read/write quorums Read from master only Write to all replicas
Ensure transactional consistency in a replicated, distributed database for multiple object updates	Distributed transactions Nested/Embedded Objects (Denormalization) Queued Writes

Figure 4. Screenshot showing use of query

The use of SMW queries and templates to render content provides several benefits:

- There is consistent "look and feel", page structure, and content placement, which improves usability;
- Content is generated dynamically; new content is immediately included and presented to users. For example, if a new Quality Attribute Scenario were created for the Consistency Quality Attribute, the table shown in Figure 4 would automatically include the new scenario.

Finally, forms can be used to specialize queries to create pages that summarize or filter the knowledge content. For example, Figure 5 shows a QuABaseBD page that uses a SMW form to specialize a query of Scalability features. Executing the query renders a page with the table shown in Figure 6. Note that SMW automatically creates hyperlinks for the table headings and values, wherever possible.



Run query: Scalability Query

Special:FormEdit/Images/4/48/BigData.png > Explore Database Technologies and Features > Special:RunQuery/Scalability Query > Images/4/48/BigData.png > Special:RunQuery/Scalability Qu

Tips:

Leave "none" if you are not interested in the value for that feature
Also tick some "none" if no results show, i.e. relax the search criteria

Scalability characteristics

Scale Out Architecture: ○ None ○ replicate complete database only ○ horizontal partitioning of database ◉ horizontal partitioning and replication

Client Request Load Balancing: ○ None ○ fixed connections to a request coordinator ◉ client requests load balanced across coordinators ○ uses HTTP-based load balancers

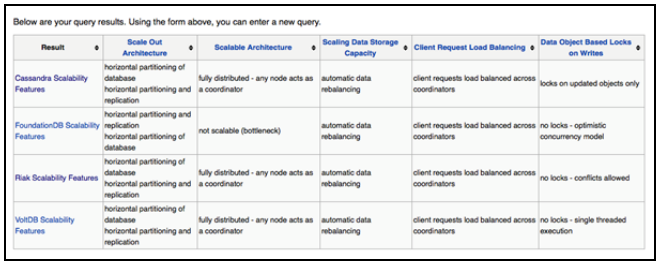
Scaling Data Storage Capacity: ◉ None ○ automatic data rebalancing ○ manual data rebalancing ○ N/A - single server only

Data Object Based Locks on Writes: ◉ None ○ locks on updated objects only ○ locks on tables/collections ○ locks on whole database ○ no locks - conflicts allowed ○ no locks - optimistic concurrency model ○ no locks - single threaded execution

Scalable Request Processing Architecture: ◉ None ○ not scalable (bottleneck) ○ fully distributed - any node acts as a coordinator ○ centralized coordinator, but can be replicated ○ based on an external load balancer

Run query

Figure 5. Scalability query form



Below are your query results. Using the form above, you can enter a new query.

Result	Scale Out Architecture	Scalable Architecture	Scaling Data Storage Capacity	Client Request Load Balancing	Data Object Based Locks on Writes
Cassandra Scalability Features	horizontal partitioning of database horizontal partitioning and replication	fully distributed - any node acts as a coordinator	automatic data rebalancing	client requests load balanced across coordinators	locks on updated objects only
FoundationDB Scalability Features	horizontal partitioning and replication horizontal partitioning of database	not scalable (bottleneck)	automatic data rebalancing	client requests load balanced across coordinators	no locks - optimistic concurrency model
Riak Scalability Features	horizontal partitioning of database horizontal partitioning and replication	fully distributed - any node acts as a coordinator	automatic data rebalancing	client requests load balanced across coordinators	no locks - conflicts allowed
VoltDB Scalability Features	horizontal partitioning of database horizontal partitioning and replication	fully distributed - any node acts as a coordinator	automatic data rebalancing	client requests load balanced across coordinators	no locks - single threaded execution

Figure 6. Results of scalability query

5. QUABASEBD USE CASES

Design and implementation decisions for QuABaseBD were driven by two primary use cases:

- Architecture Design: The target user is an architect who has little experience with big data systems and NoSQL technology. He or she uses the General Scenarios and Quality Attribute Scenarios in QuABaseBD to support definition of architecturally significant requirements. The Quality Attribute Scenarios are used to select appropriate Tactics, and finally, one or more candidate NoSQL products are selected that implement the tactics.
- Identification of Alternatives: The target user is an architect who has some experience with big data systems and NoSQL technology. The architect knows some or all of the product features needed for his or her system, and uses the QuABaseBD feature queries to identify one or more suitable candidate NoSQL products.

Both of these cases might be followed by a use case in which the architect "works backwards" from a candidate NoSQL product. Large COTS products, like these NoSQL databases, implement many tactics, with each tactic embodying a set of quality attribute tradeoffs. An architect uses QuABaseDB to identify a product because it supports tactics he or she desires, but then the architect must ensure that other tactics supported by the product to not embody tradeoffs that would be detrimental to system qualities. QuABaseBD supports tracing from the Tactics that are implemented by a candidate product, identifying the quality attribute tradeoffs that those tactics embody, and using Quality Attribute Scenarios to provide concrete examples of the implications of each tradeoff.

6. USER TRIALS

In order to prepare for public deployment of the QuABase, we have performed usability and utility testing of the resulting Web

site. We publicized QuABaseBD through the authors' professional networks, and opened QuABaseBD to volunteers to perform testing. Volunteers were requested to purposefully explore the knowledge base: when access credentials were issued, they were encouraged to use the knowledgebase to solve a particular design problem that we provided, rather than simply browsing the QuABase content. These open access sessions were limited to one hour in duration. Users were given a worksheet to record their impressions from the testing. There was no additional training, guidance, or instruction provided.

All of the 20 users who provided feedback on their experience were software architects with 5-23 years of experience. All but two users characterized their expertise in big data systems as "somewhat knowledgeable", but they did not have any specific experience in any of the database technologies currently represented in QuABase.

The "main page", where users enter the knowledge base after authenticating, offers three options: *Explore Software Design Principles*, *Explore Database Technologies and Features*, and *Explore Architecture Tactics for Big Data Systems*. The workflow that testers employed was split nearly evenly between those that started with architecture tactics, and those that started with database features. No testers started with design principles, although some testers eventually explored this section of the knowledgebase. These workflows matched our pre-test expectations: The software design principles path is intended to help define architecturally significant requirements. If these are already established (as they were in the test problems), then we expect users to start with tactics (top-down reasoning), or database features (bottom-up reasoning). This gives us confidence that the QuABase design is structured to support this use case.

Testers starting with database features made extensive use of the faceted search capability and the tabular results visualizations (for example, the pages shown in Figure 6). The information in these tables was sufficient to answer the tester's questions. Few of these testers relied on the detailed feature description pages. In contrast, testers who started with tactics relied on following links within QuABaseBD, which led them to the detailed feature description pages. Interestingly, no testers employed the full-text search capability of the SMW platform.

All but one tester said that they were able to answer all of their questions using the content of QuABaseBD. In providing feedback on the utility of the knowledge base, testers were asked to rate their confidence that their answers were complete and correct. The large majority rated their confidence as 3 or 4 on a scale of 1 (no confidence) to 5 (absolutely sure).

Only one tester followed any of the hyperlinks to external resources that are provided in the detailed product feature descriptions. We hypothesize that this may be due to the limited time we allowed for testing, but again this gives us confidence that the current QuABaseBD content is sufficiently extensive to meet many of the anticipated needs of the big data software engineering community.

7. CONCLUSIONS AND FUTURE WORK

QuABaseBD provides an example of how a decision support design assistant can help architects who are not familiar with a system domain (in this case, big data systems) and also those working in a broad and dynamic technology domain (in this case, NoSQL databases). The core knowledge model and the feature meta-model, along with much of the SMW customizations, are not specific to any domain or technology, and so are reusable to create other design assistants.

Based on our experience creating and using QuABaseBD, we see several areas for future work, which we consider in more detail in the following sections:

- Knowledge visualization;
- Automating the population of the feature model;
- Curation practices and processes.

7.1 Knowledge Visualization

Information visualization is a powerful technique for conveying complex concepts and knowledge in ways that are easily understood by humans [12]. Information visualization presumes that "visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways" [13].

QuABaseBD currently implements several basic visualizations that graphically show the relationships based on the semantic annotation of the knowledge and feature information. For example, FX shows part of a visualization that shows the tactics and resulting tradeoffs for a particular NoSQL product.

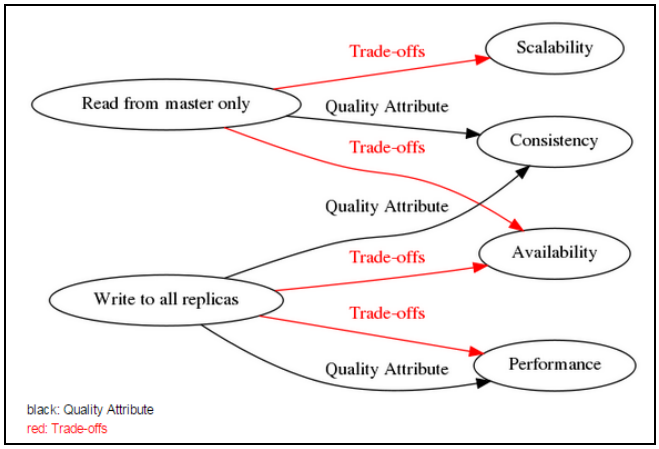


Figure 7. Tactics supported by Neo4j product (Extract)

We have begun experimenting with more sophisticated visualizations, particularly to summarize and compare suitability of two or more products, based on how well the tactics and features match the user's requirements. To this end, we are considering approaches for graphical multidimensional comparison of two or more products such as Kiviat ("radar") diagrams, heat map plots, and bubble charts.

The practical challenges in this area are integrating the graphics generation framework into SMW, so that the visualizations are an integral part of the QuABase user experience.

The research challenges include extending the semantics of the feature model to rank the qualitative values for a particular feature, and how to represent multi-dimensional comparisons of qualitative information in ways that are useful for architects using the tool.

7.2 Automating the Population of the Feature Model

Our goal in creating QuABaseBD was to provide a trusted source of information for architects working in the large and dynamic NoSQL database technology domain, and to design a decision support system that would be reusable in other technology domains. To date, we have populated the feature model in QuABaseBD with data for nine NoSQL products. This has been a time-intensive and labor-intensive process, beginning with identifying and locating the relevant product documentation, learning product-specific terminology, and mapping product feature descriptions into the feature model. Since the feature model was initially populated, most of the products included have undergone at least one significant revision, necessitating review and update to the content. This manual approach is not sustainable: Alternatives are automation and increasing the number of contributors.

We are currently investigating the use of machine learning to automate the feature model population of QuABaseBD, using the Concept Graph Learning (CGL) machine learning method [14]. The QuABaseBD semantic knowledge model provides the necessary linked information for CGL to learn a directed universal concept graph that represents the major concepts in big data systems. CGL will use the learned graph to predict unobserved relations from new data – the documentation pages for specific big data technologies that we wish to include or update in the feature model. This will enable QuABaseBD to be rapidly updated to reflect the characteristics of new and evolving implementation technologies.

7.3 Curation Practices and Processes

In preparation for public release of QuABaseBD, we are consulting with experts on each included product to validate that our curated values for the database features are correct. We believe that, as a curated scientific knowledge base, there are high expectations that the content in QuABaseBD is trustworthy at all times.

Even if some of the feature model population is automated, as discussed above, there is still a need to increase the capacity of knowledge content creation to keep up with the size and pace of change in the NoSQL technology domain.

We are considering several alternative approaches that would allow contributions by a broader group, which might include any QuABaseBD user, or be narrower, for example NoSQL product developers or a pre-qualified group of experts. There are benefits and costs to each approach, and certainly there are lessons to be learned from open source software projects, crowdsourced projects such as Wikipedia, and other online communities.

To this end, we are working to design a systematic curation process where a small cohort of experts will be responsible for changes to the content. We anticipate that visitors to the knowledge base will be able to suggest changes through associated comments pages, and the curators will assess these proposals for inclusion.

8. ACKNOWLEDGMENTS

Copyright 2015 ACM. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute. This material has been approved for public release and unlimited distribution. DM-0002243.

9. REFERENCES

[1] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley, 2005.

[2] "TOGAF Version 9.1." The Open Group, Standard, 2011, http://pubs.opengroup.org/architecture/togaf9-doc/arch/ (Accessed 10 Mar 2013).

[3] I. Gorton and J. Klein, "Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems," *IEEE Software,* vol. PP, no. 99, 18 March 2014. doi: 10.1109/MS.2014.51

[4] J. Klein, I. Gorton, N. Ernst, et al., "Performance Evaluation of NoSQL Databases: A Case Study," in *Proc. of 1st Workshop on Performance Analysis of Big Data Systems (PABS 2015)*, Austin, TX, USA, 2015. doi: 10.1145/2694730.2694731.

[5] F. Bachmann, L. Bass, and M. H. Klein, "Preliminary Design of ArchE: A Software Architecture Design Assistant." Software Engineering Institute, Technical Report, CMU/SEI-2003-TR-021, 2003.

[6] A. Tang and H. van Vliet, "Software Architecture Design Reasoning". In M. A. Babar, T. Dingsøyr, P. Lago, et al., (Eds.), *Software Architecture Knowledge Management: Theory and Practice* (pp. 155-174). Heidelberg: Springer, 2009.

[7] K. Kovacs. Cassandra vs MongoDB vs CouchDB vs Redis vs […] comparison [Online]. http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis (Accessed 7 Oct 2014).

[8] J. Yu and R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, *Journal of Grid Computing*, Volume 3, Numbers 3-4, Pages: 171-200, Sept. 2005.

[9] M.A. Babar, "Supporting the Software Architecture Process with Knowledge Management". In M. A. Babar, T. Dingsøyr, P. Lago, et al., (Eds.), *Software Architecture Knowledge Management: Theory and Practice* (pp. 69-111). Heidelberg: Springer, 2009.

[10] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, 3rd Edition.* Addison-Wesley, 2013.

[11] I. Gorton, J. Klein, and A. Nurgaliev, "Architecture Knowledge for Evaluating Scalable Databases," in *Proc. 12th Working IEEE/IFIP Conf. on Software Architecture (WICSA 2015)*, Montreal, Canada, 2015.

[12] Joe Kielman, Jim Thomas, and Richard May. 2009. Foundations and frontiers in visual analytics. *Information Visualization* 8, 4 (December 2009), 239-246.

[13] James J. Thomas and Kristin A. Cook (Eds.) *Illuminating the Path: The R&D Agenda for Visual Analytics*. National Visualization and Analytics Center. p. 30, 2005.

[14] Y. Yang, H. Liu, J. Carbonell, and W. Ma. "Concept Graph Learning from Educational Data". In *Proc. 8th ACM Intl. Conf. on Web Search and Data Mining* (WSDM '15), Shanghai, China, 2015. doi: 10.1145/2684822.2685292