# An Architecture of Knowledge Assistant

A System for Knowledge Extraction, Searching and Displaying

Stone Fang (Student ID: 19049045)

## Contents

**1 Introduction** **2**
1.1 System Overview . . . 2
1.2 Motivation & Purpose . . . 2
1.3 Related Work . . . 2
1.4 Methodology . . . 3

**2 Architectural drivers** **3**
2.1 Business Goals and Constraints . . . 3
2.2 Technical Goals, Assumptions and Constraints . . . 3
2.3 Primary Functional Requirements . . . 3
2.4 Quality Attributes . . . 4
2.4.1 Stakeholder's Concerns . . . 4
2.4.2 Quality Attribute Scenarios . . . 4
2.4.3 Priority of QAs . . . 5
2.5 Evaluation Criteria . . . 5
2.5.1 Modularity . . . 5
2.5.2 Modifiability . . . 5
2.5.3 Scalability . . . 5
2.5.4 Resource Utilization . . . 5

**3 Views** **5**
3.1 Logical View . . . 6
3.1.1 Primary Presentation . . . 6
3.1.2 Element Catalog . . . 6
3.1.3 Context Diagram . . . 7
3.1.4 Architecture Background . . . 7
3.2 Process View . . . 8
3.2.1 Primary Presentation . . . 8
3.2.2 Element Catalog . . . 8
3.2.3 Context Diagram . . . 9
3.2.4 Architecture Background . . . 9
3.3 Development View . . . 10
3.3.1 Primary Presentation . . . 10
3.3.2 Element Catalog . . . 10
3.3.3 Context Diagram . . . 11
3.3.4 Variability Guide . . . 11
3.3.5 Architecture Background . . . 11

**4 Conclusion and Future Work** **12**

**References** **13**

# 1 Introduction

## 1.1 System Overview

This system is aiming for a set of services of knowledge collection, extraction, aggregation, searching, and visualization. The ultimate goal of this system is to help people to access and acquire knowledge they want more efficiently. For example, a university student can use this system to help their study by searching and exploring the concepts and theories relating to their study. Another example could be that a researcher starting his/her research in a new area can use this system to get a quick and comprehensive understanding of the basis, methodologies, and state-of-art of the that area.

In order to achieve this, the system will have these features:

- automatic knowledge extraction
- knowledge representation
- persistance and storage
- UI for searching and exploration
- knowledge displaying and visualization

## 1.2 Motivation & Purpose

We are living in a data explosion time. We have numerous and rapid-growing amount of data every second, which, however, makes it more difficult than before to extract useful information from it, and even more to acquire real knowledge (Allahyari et al., 2017) (Lupiani-Ruiz et al., 2011). For example, as a common case, when someone gets started to do research on a new topic, first s/he need to get familiar with the basic concepts in that topic, how its position in a bigger background, and how it is related to or different from other concepts. To do this, researchers usually collect surveys in this topic or simply go to the Wikipedia page as a quick start. However, the researcher needs to go through many articles s/he gathered, select important parts and key concepts, and form a mental representation in the brain. (Beel, Gipp, Langer, & Breitinger, 2016) Based on such consideration, it would be better and faster to understand if we can create a system which helps in knowledge collecting, aggregating (Zhao et al., 2017), and presenting in a more structured, intuitive and easier-to-understand way (Liu et al., 2018). This will help the researcher shorten the boarding time, and get more comprehensive catching to a new area.

## 1.3 Related Work

Many efforts have already been contributed to target at this problem. Semantic Web makes meanings from wide spread heterogeneous information and data and thus provides more accurate answer for user query and enables more efficient acquisition of knowledge (Lei, Uren, & Motta, 2006) (Lupiani-Ruiz et al., 2011). However, this kind of technology focuses on making semantic meanings of web data to improve the traditional text search engine, but not incorporate data visualization method to represent the meanings or knowledge in the data.

Knowledge Graph, as an emerging technology recently, which extracts and integrates knowledge from massive unstructured data and represents them in entity-relation triples, is widely used to improve a variety of applications such as search engines and recommendation systems (Zhao et al., 2017). The limitation of Knowledge Graph is that knowledge is only represented in the form of graph.

Some other approaches focusing on specific area were also studies. In academic area, scholar paper recommendation system is studied as a method to decrease information overload (Beel et al., 2016) (Beel, Aizawa, Breitinger, & Gipp, 2017), while a comprehensive overview of scholarly data visualization was given by (Liu et al., 2018). However, the recommender usually works in level of documents. In Computer Science and Software Engineering area, researchers contributed to searching algorithms and/or systems for algorithms (Tuarob, Bhatia, Mitra, & Giles, 2016), software components (Garcia et al., 2006) and design patterns (Ampatzoglou, Michou, & Stamelos, 2013), which are task specific search engines instead of for general knowledge.

### 1.4 Methodology

In this article, the attribute-driven design (ADD) method is followed (Bass, Clements, & Kazman, 2012). In section sec. 2, a list of quality attributes (QA) are identified from a comprehensive analysis of goals and constraints from both business and technical aspects. The QAs identification is based on the Product quality model in (ISO/IEC 25010, 2011). In section sec. 3, three views are formulated based on the architectural drivers with detailed documentation and explanation. UML diagrams (Cook et al., 2017) are used to describe the architectural views.

## 2 Architectural drivers

### 2.1 Business Goals and Constraints

This system will add values to anyone who search or learning new knowledge, mostly academic community such as tertiary students, researchers, or practitioners. Since it is for academic community, it is preferably open to everyone (Molloy, 2011), free to use, and low cost.

### 2.2 Technical Goals, Assumptions and Constraints

There are important constraints in technical respect, which are mainly caused by the huge amount, complex nature of knowledge and the immaturity of current knowledge representation and artificial intelligence algorithms (Allahyari et al., 2017).

- The first constraint comes from the complex nature of knowledge and the lack of universal representation method suitable for all kinds of knowledge and requirements from different knowledge consumers (Ristoski & Paulheim, 2016). Therefore, there are constraints on the knowledge representation and displaying:
  - The system should not only incorporate one representation, but has to be capable to handle a range of representations, and is better to take into account future changes.
  - The system should show knowledge in different views, that is, should support multiple views and visualization methods, switching between and combination of these views (Liu et al., 2018).
- The second constraint is the huge volume of knowledge. Therefore, knowledge gathering and extracting processes are highly desired to be automatic (Allahyari et al., 2017). Thus comes the following constraints:
  - The system should be capable to adopt AI algorithms to do these task
  - There is no universal algorithms for knowledge extraction so this system should be able to handle a variety of knowledge extraction algorithms, and better to take into account future improvements or upgrades. In (Beel et al., 2017), the architecture of a recommendation-as-a-service platform was designed to be open to third-party recommendation algorithm researchers.
  - AI algorithms such as deep learning models are easily error-prone, so the system should provide the features for manual correction of mistakes.

### 2.3 Primary Functional Requirements

The primary features of this system will be as follows:

- Gathering and extraction: the system will gather, extract and structure knowledge from a variety of sources mainly text or semi-structured data, for example, Wikipedia.
- Persistency: the system should persistent the acquired knowledge in certain representation for querying and exploration.
- Displaying: show knowledge in more intuitive, fast-and easy-reading ways.
- Searching: query for desired concepts with related concepts.

## 2.4 Quality Attributes

### 2.4.1 Stakeholder's Concerns

The proposed system has three main kinds of stakeholders and the following general functionality:

- Users: a) The system should be easy to use and intuitive; b) Users should be able to correct errors introduced by automatic knowledge extraction algorithms.
- Developers
  - System developer: The system should be highly decoupled from specific knowledge representation/extraction/visualization algorithms so upgrade on any of these modules won't break the whole system.
  - Algorithmic developer/knowledge engineer: The system should be flexible and extensible so that improving, upgrading, and replacing algorithms is simple and won't affect other modules.
- Operator: The system should be maintained in low cost

### 2.4.2 Quality Attribute Scenarios

**2.4.2.1 QAS 1: Modularity** The knowledge representation part, automatic knowledge extraction part, and visualization/rendering/displaying part may be responsible for different developers so they should be highly decoupled, isolated, and independently changeable.

- Stimulus: different kinds of developers work on different modules
- Environment: development time, maintenance time
- Response: modules of the system and contract between modules
- Response measure: degree of decoupling and dependency of all modules of the system.

**2.4.2.2 QAS 2: Modifiability** A developer may improve or introduce a new knowledge representation or AI algorithm for knowledge extraction or visualization method because he find a new one. Therefore, the system should be highly modular and decoupled so each module can be modified without affecting others.

- Stimulus: better algorithms/methods are discovered and will replace the old ones.
- Environment: development time, runtime
- Response: new methods are successfully created and integrated into the service
- Response measure: degree of the defects or quality degrading introduced by modification.

**2.4.2.3 QAS 3: Performance - Resource Utilization** As the system should be running in low cost, it have to make use of resources very efficiently.

- Stimulus: operation of the system
- Environment: runtime
- Response: the resources required by running the service are satisfied
- Response measure: the cost of operation of system

**2.4.2.4 QAS 4: Scalability** Since new knowledge will be continuously added into the system's knowledge base so the system should be able to scaled.

- Stimulus: new data come into knowledge base
- Environment: runtime, operation time
- Response: system are scaled to handle new data
- Response measure: the time of scaling

### 2.4.3 Priority of QAs

Based on our business/technical goals and the QA scenarios described above, these QAs are prioritized from highest to lowest as follows

1. Modularity
2. Modifiability
3. Scalability
4. Resource Utilization

## 2.5 Evaluation Criteria

The architecture will be evaluated based on the drivers identified in above sections.

### 2.5.1 Modularity

Modularity will be evaluated by:

- the degree of decoupling and isolation of modules; higher is better
- the overall dependencies between modules; lower is better

### 2.5.2 Modifiability

Modifiability will be evaluated by:

- the impact or effort required on other components of the system to modify one module; lower is better

### 2.5.3 Scalability

Scalability will be evaluated by:

- the effort to scale out the system in order to handle larger size of data and amount of traffic; lower is better

### 2.5.4 Resource Utilization

Resource utilization will be evaluated by:

- the resources required by the architecture given the size of data and number of requests per second; lower is better

# 3 Views

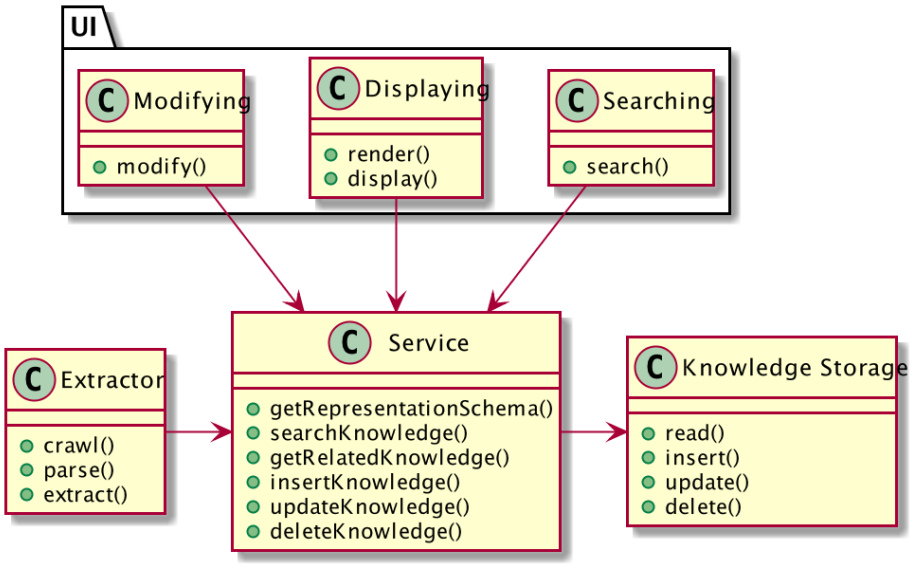
In this section, threw views are described following the "4+1 view" model (Kruchten, 1995), including:

- Logical view
- Process view
- Development view

Each view is presented in UML diagram and justified according to the architectural drivers and quality attributes analyzed in section sec. 2.

## 3.1 Logical View

### 3.1.1 Primary Presentation




### 3.1.2 Element Catalog

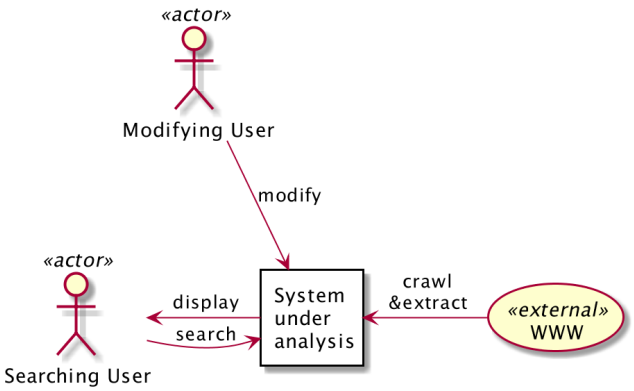
#### 3.1.2.1 Elements

- Storage: store extracted knowledge
- Service: provide unified interface for accessing knowledge representation.
- Extractor: automatic knowledge extraction process enabled by machine learning algorithms
- UI
  - Searching: user search certain concepts
  - Displaying: easy-reading, intuitive display of searched and related concepts, preferably in visual ways
  - Modifying: user modify knowledge in the storage to correct errors introduced by automatic algorithms

#### 3.1.2.2 Relations

- Extractor and Service
  - the automatic Extractor will generate new content of knowledge and write into Storage through unified Service.
  - Extractor may read existing knowledge to improve its performance according to the AI algorithms
- Service and Storage: Storage will not interact with other parts directly. Service will act as an abstract layer to isolate the storage and other parts.
- UI and Service: UI interacts with end user and convert user searching/modifying/browsing into requests to server and display data accordingly.

### 3.1.3 Context Diagram



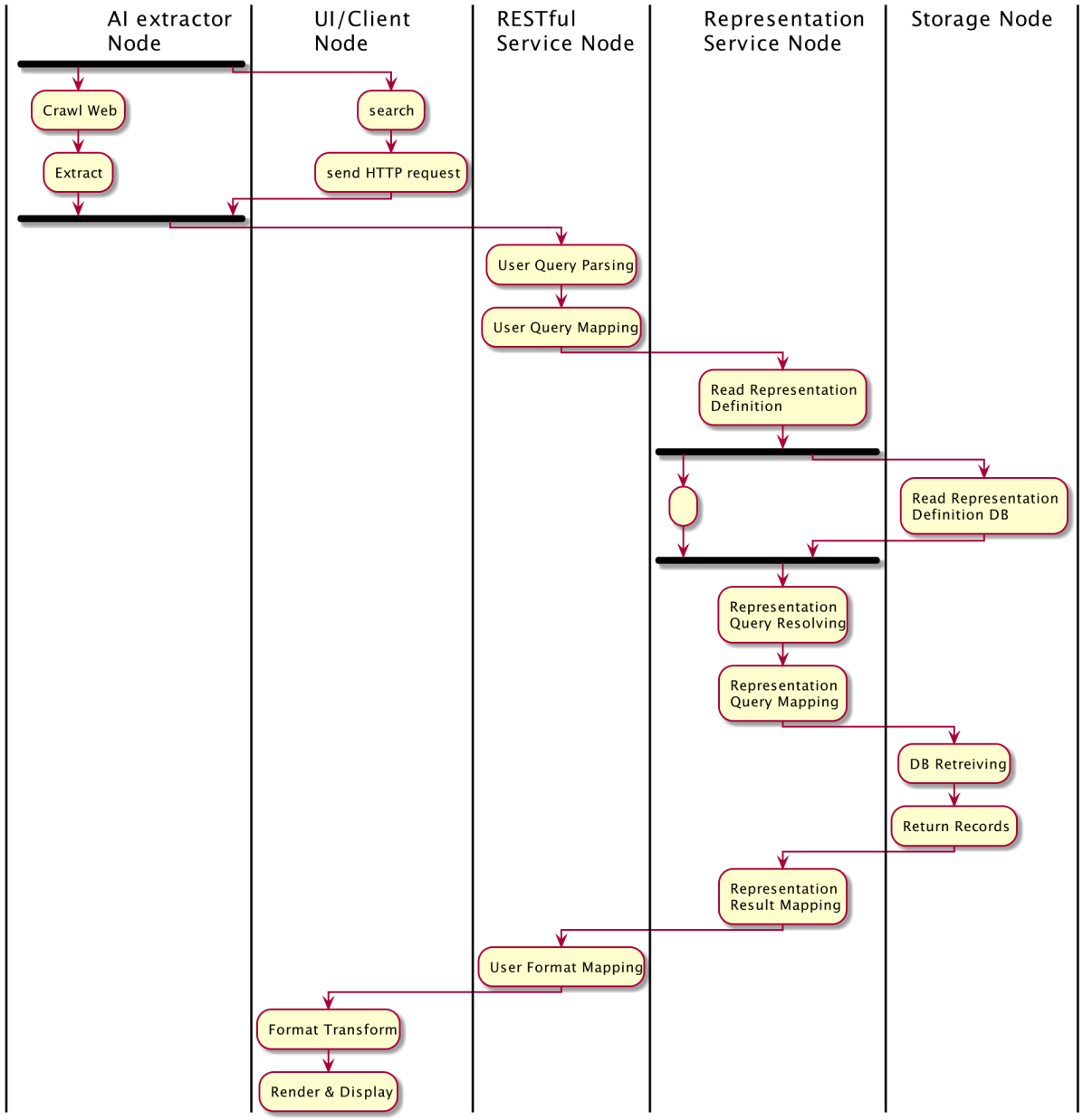

### 3.1.4 Architecture Background

**3.1.4.1 Rationale** As modularity is at the highest priority, the Service module is introduced as an adaptor and isolation layer to keep the knowledge storage base isolated from the Extractor and UI parts. The knowledge representation stored in database and AI algorithms for knowledge extraction are decoupled so that they can change independently.

**3.1.4.2 Other Systems Explored** (Zhao et al., 2017) proposed a system harvesting webpages and build domain specific knowledge graph from them by intensively using machine learning and text mining approaches such as SVM, POS Tagging, dependency parsing. Though this paper mainly focus on the algorithmic aspect of the system, it is a valuable reference on how we can accommodate various processing modules.

## 3.2 Process View

### 3.2.1 Primary Presentation




### 3.2.2 Element Catalog

#### 3.2.2.1 Elements and Their Properties

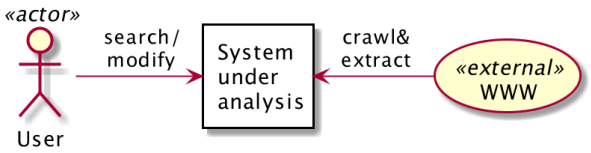
- AI Extractor Node: Nodes running AI algorithms for crawling the web and extract knowledge.
- UI/Client Node: User interface for user to search, browse and modify.
- RESTful Service Node.
- Representation service Node: searching and get definition of knowledge representation
- Storage Node: service node with database and storage abstraction.
- Crawl web: read web data as knowledge source.
- Extract: extract knowledge from raw data with AI algorithms.
- Search: user start to search.
- User query mapping: map user query to knowledge representation and queries.

- Representation query mapping: map representation and mapping into physical storage format and query.
- Representation result mapping: map from physical storage format into knowledge representation.

#### 3.2.2.2 Relations

- UI/Client Node and RESTful service Node: UI will convert user operations to requests to RESTful service.
- RESTful Service Node and Representation Service Node: the former acts as an wrapping and isolation layer of the latter.
- Representation Service Node and Storage Node: abstract representation will be mapped to physical storage by DB querying.

### 3.2.3 Context Diagram



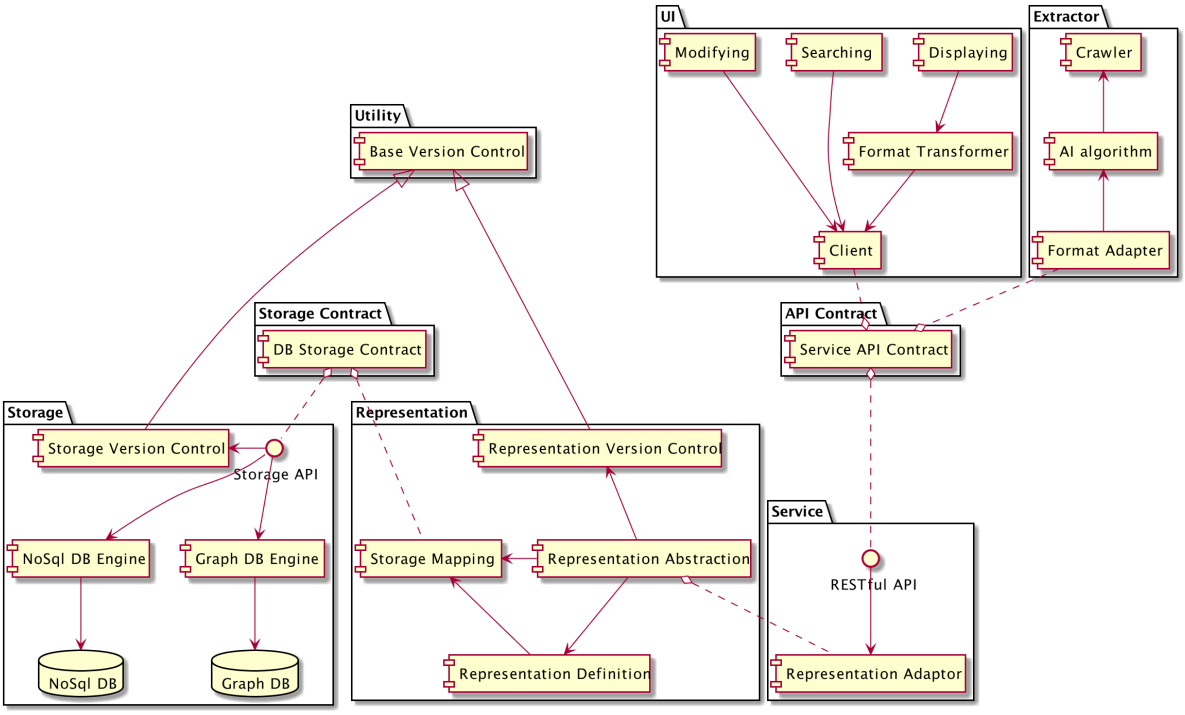

### 3.2.4 Architecture Background

**3.2.4.1 Rationale** Each part of UI, Representation Service and Storage are designed as independent processes so that they can be developed and deployed to different physical machines or cloud computing instance thus reach resource efficiency.

All data is stored in Storage Nodes and other services are stateless so the system can be scaled out by increasing the running instance of services.

## 3.3 Development View

### 3.3.1 Primary Presentation




### 3.3.2 Element Catalog

#### 3.3.2.1 Elements and Their Properties

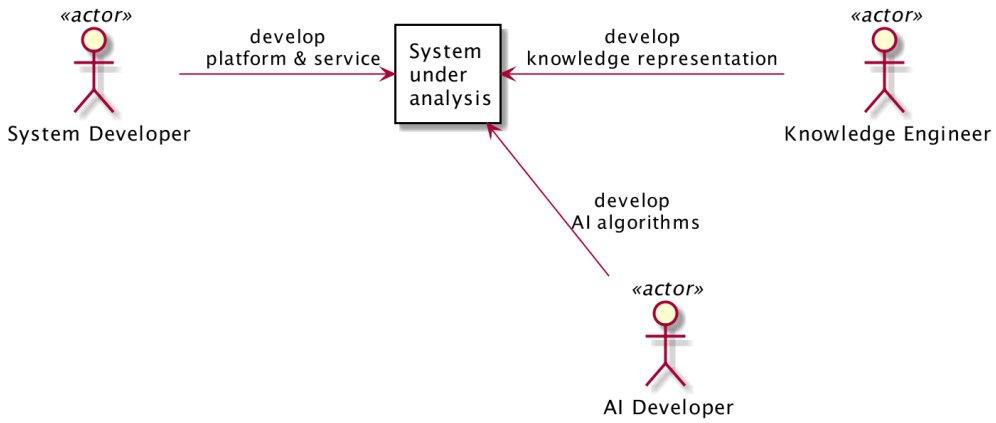
- Storage
  - NoSql DB & Graph DB: database storage with these two used for demonstrating the concept.
  - DB Engines: data access layers for corresponding databases.
  - Storage API: unified API service for knowledge storage, abstract from concrete database.
  - Storage Version Control: management of different storage scheme to handle co-existence and compatibility when changes/upgrade need to be made.
- Representation
  - Representation Definition: definition of knowledge representation, maybe use JSON or XML format.
  - Representation Abstraction: representation abstraction from physical storage format
  - Storage Mapping: map from logical representation to physical representation supported by Storage module.
  - Representation Version Control: management of different knowledge representation scheme to handle co-existence and compatibility.
- Service
  - Representation Adaptor: adaption layer as an isolation between RESTful API and representation module.
  - RESTful API: provide unified service for both automatic extractor and UI.
- UI
  - Client: an HTTP client interacting with service
  - Searching: knowledge searching UI and functionality
  - Modifying: UI and functionality for user to modify contents in knowledge storage for error correction.
  - Displaying: knowledge displaying, visualization and rendering.
- Extractor
  - Crawler: crawls the web and collect contents as the source of knowledge.

  - AI Algorithms: automatic knowledge extraction algorithms which may need to read existing knowledge for better performance.
  - Format Adapter: adaption layer to transform the format of results from AI algorithms to fit the unified service.
- Data Exchange Contract modules:
  - Storage Contract: contract between storage module and representation module.
  - Service API Contract: contract between UI and knowledge service.

#### 3.3.2.2 Relations

- Storage, Representation and Storage Contract: the storage module and representation module are loosely coupled by introducing the storage contract for data exchange. The two modules can be modified independently as long as the keep consistent to the contract.
- Representation Abstraction, Service and Representation Definition: the service will interact with representation module, and representation module will manage different representations to provide abstracted functionality transparent to lower level details.
- UI, Service and Service API Contract: the UI and service are decoupled by the service API contract.

### 3.3.3 Context Diagram




### 3.3.4 Variability Guide

1. MongoDB and Neo4j are shown in the diagram but in design the system should be flexible to be plugged with and abstracted from other kinds of databases.
2. Other kind of services may be used instead of RESTful APIs. For example, GraphQL can be introduced into the system if it is found to be more suitable in some cases.

### 3.3.5 Architecture Background

**3.3.5.1 Rationale** The development of the system is divided into several packages. The UI and backend Service interacts data with Service API Contract. The Service and Representation exchange data by Representation Definition. The Representation and physical Storage exchange data by Storage API Contract. These packages can be developed, modified and tested separately, thus increases modularity and modifiability.

**3.3.5.2 Other Systems Explored** In (Kruchten, 1995) the authors described the architecture of a scholarly recommendation system. It use rational database, MySQL, as primary storage. However, in our case, NoSQL or Graph DB are more suitable because our data structure would be more diverse, complex and flexible. In terms of the implementation of search engine, this paper incorporates Apache

Solr/Lucene. This is also a good choice for our system, but the modularity, abstraction and isolation should be taken into account.

# 4 Conclusion and Future Work

As the posts on the web as well as academic publications are increasing exponentially, it can effectively reduce the cost of acquiring information and learning new knowledge with a system helping users access related articles and read them easily. This article proposed the architecture of an knowledge assist system collecting, extracting, aggregating, searching and exploring knowledge aiming to help people acquire and learn knowledge more efficiently. The ADD methodology is followed to create the architecture. First, the architecture drivers are presented and analyzed. Second, three views are demonstrated, namely logical view, process view and development view.

There are some aspects of the system not covered and can improve the architecture design further:

- Deployment view and use case scenarios are not covered, but they are also useful for demonstrating some quality attributes such as scalability, performance, usability (Kruchten, 1995).
- The core part of the system can be refined. Since the primary QA are modularity, the decomposition, decoupling, abstraction and isolation of components need to be analyzed into more detail and carefully examined (Ampatzoglou et al., 2013).
- Algorithm/representation evaluation and selection system can be introduced for choosing among different AI extraction algorithms, knowledge representations and visualization methods. For example, an A/B testing system can be used for evaluation (Beel et al., 2017). Selection can work in a manual way or automatically by machine learning algorithms.

# References


Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. *arXiv E-Prints*, arXiv:1707.02919. Retrieved from http://arxiv.org/abs/1707.02919

Ampatzoglou, A., Michou, O., & Stamelos, I. (2013). Building and mining a repository of design pattern instances: Practical and research benefits. *Entertainment Computing*, *4*(2), 131–142. https://doi.org/https://doi.org/10.1016/j.entcom.2012.10.002

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). Addison-Wesley Professional.

Beel, J., Aizawa, A., Breitinger, C., & Gipp, B. (2017). Mr. DLib: Recommendations-as-a-service (raas) for academia. *2017 acm/ieee joint conference on digital libraries (jcdl)*, 1–2. https://doi.org/10.1109/JCDL.2017.7991606

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338. https://doi.org/10.1007/s00799-015-0156-0

Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2017). Unified modeling language (UML) version 2.5.1 [Standard]. Retrieved from Object Management Group (OMG) website: https://www.omg.org/spec/UML/2.5.1

Garcia, V. C., Lucrédio, D., Durão, F. A., Santos, E. C. R., Almeida, E. S. de, Mattos Fortes, R. P. de, & Lemos Meira, S. R. de. (2006). From specification to experimentation: A software component search engine architecture. In I. Gorton, G. T. Heineman, I. Crnković, H. W. Schmidt, J. A. Stafford, C. Szyperski, & K. Wallnau (Eds.), *Component-based software engineering* (pp. 82–97). Berlin, Heidelberg: Springer Berlin Heidelberg.

ISO/IEC 25010. (2011). Systems and software engineering—systems and software quality requirements and evaluation (SQuaRE)—system and software quality models.

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE Software*, *12*(6), 42–50. https://doi.org/10.1109/52.469759

Lei, Y., Uren, V., & Motta, E. (2006). SemSearch: A search engine for the semantic web. In S. Staab & V. Svátek (Eds.), *Managing knowledge in a world of networks* (pp. 238–245). Berlin, Heidelberg: Springer Berlin Heidelberg.

Liu, J., Tang, T., Wang, W., Xu, B., Kong, X., & Xia, F. (2018). A survey of scholarly data visualization. *IEEE Access*, *6*, 19205–19221. https://doi.org/10.1109/ACCESS.2018.2815030

Lupiani-Ruiz, E., GarcíA-Manotas, I., Valencia-GarcíA, R., GarcíA-SáNchez, F., Castellanos-Nieves, D., FernáNdez-Breis, J. T., & CamóN-Herrero, J. B. (2011). Financial news semantic search engine. *Expert Syst. Appl.*, *38*(12), 15565–15572. https://doi.org/10.1016/j.eswa.2011.06.003

Molloy, J. (2011). The open knowledge foundation: Open data means better science. *PLoS Biology*, *9*, e1001195. https://doi.org/10.1371/journal.pbio.1001195

Ristoski, P., & Paulheim, H. (2016). Semantic web in data mining and knowledge discovery: A comprehensive survey. *Journal of Web Semantics*, *36*, 1–22. https://doi.org/https://doi.org/10.1016/j.websem.2016.01.001

Tuarob, S., Bhatia, S., Mitra, P., & Giles, C. L. (2016). AlgorithmSeer: A system for extracting and searching for algorithms in scholarly big data. *IEEE Transactions on Big Data*, *2*(1), 3–17. https://doi.org/10.1109/TBDATA.2016.2546302

Zhao, X., Xing, Z., Kabir, M. A., Sawada, N., Li, J., & Lin, S. (2017). HDSKG: Harvesting domain specific knowledge graph from content of webpages. *2017 ieee 24th international conference on software analysis, evolution and reengineering (saner)*, 56–67. https://doi.org/10.1109/SANER.2017.7884609