Software Product Line Architectures and framework for Social Media Applications.

Bhattacharya, Amitdeb
A20402789
abhattacharya2@hawk.iit.edu

Chandrashekar, Pallavi
A20427289
pchandrashekar@hawk.iit.edu

Channarayapatna Mahesh, Karthik
A20383027
kmahesh@hawk.iit.edu

Abstract—The development of social media application opens several challenges for developers. The most challenging among them is porting the application to heterogeneous devices available on market. This requires developers to create and maintain several versions of their applications in order to deal with particular features of each platform, including display size, development libraries, sensors, keypad layout, etc. **Software Product Line** approach seems to be very useful technique to support this kind of development.
Keywords: Software Product Line(SPL)Architecture, Social media.

1. Introduction

A Software Product Line Architecture is the backbone in the field of Software Product Line Engineering. Software product line architectures is drawing more attention in the software research community in the recent days. The adoption of Software Product Line Architecture has augmented the quality of the software magnificently. Software Product Line Architecture allows for the derivation of products, defines corresponding quality attributes Software product lines is a combination of products and reusable assets which play a key role in the functionality of the products.

Software product line is a set of software program-in depth systems sharing a not unusual, managed set of features that satisfy the unique wishes of a particular marketplace phase or mission and which are advanced from a commonplace set of core belongings in a prescribed way. Structures can be custom designed the use of version points within its layout, as a result bearing in mind versions to form the final product based at the configuration. This variability can be modeled the usage of normally used characteristic-modeling, that could assist specific unique required functions and variations of features. Software Product Line (SPL) represents a paradigm change in the regard of the traditional software development. Instead of developing software “project-to-project”, organizations should now focus their efforts on creating and maintaining a core asset, which would be the basis for the construction of specific products for a given domain.

Software product line libraries contain any software related articles like specification design, code user documentation etc. This paper represents the architecture and design of software

product line library. In this paper, we also discuss about the management of core asset library which is one of the software product line library. The idea of SPL is reasonable to areas in which there is an interest in items that have regular elements, which additionally contain an characterized set of variations. Variations is stand out among the most imperative issues in outlining the SPL which reflects the diverse qualities and shared traits of its objects. Hence in this way the exact portrayal of variations makes it possible for the generation of particular products in SPL. The activities of SPL are time consuming and needs much effort as it is not trivial due to the complexities involved in the process.

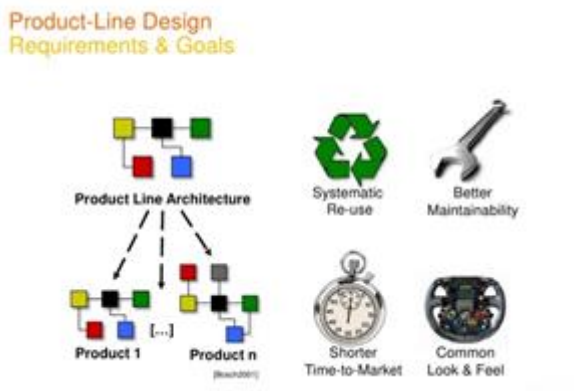

Fig.1. Overview of SPL

2. CATEGORIES IN THE EVOLUTION OF SOFTWARE PRODUCT LINE ARCHITECTURE (SPL)

Mentioned below are the eight categories of evolution that requirements cause on the architecture.

1.Breaking of software product line:

If the new sets of software products are created, there shall be a decision to make sure, whether the new product is kept in the same product line, or it is necessary to split the software product lines into two, that is to use the existing product line architecture as a template for the other product line architecture. The software product line architecture may exist in two different directions but possess similar traits.

2. Derivation of Software product line architecture:

The software functionality of the complete product line is shared, such the rest of the products line could be benefitted from bug fixing and other quality improvements. The software product line is branched into a sub product lines rather copying the product line architecture and continuing evolution of both software product lines independently.

3. New software product line architecture component:

There may be few requirements that could not be solved in the existing software product line architecture components. Consider a big block of functionality do not practically fit in any framework implementation, but used by several components. The components are adjusted if the software product line architecture is extended for a new product line component. This is illustrated in figure 2.

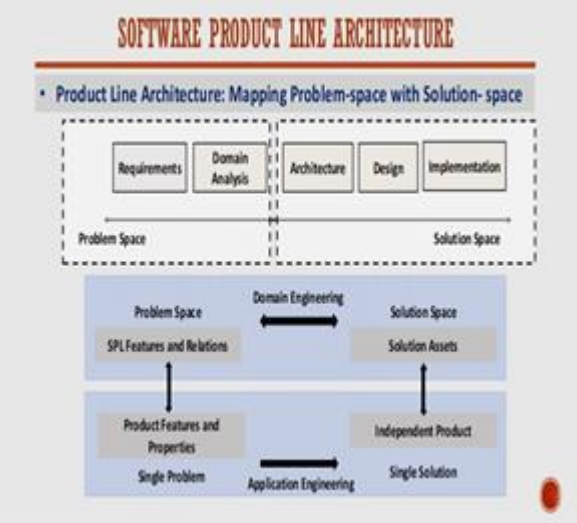

Fig. 2. SPL Architecture

4. Replacement of Software Product line architecture component:

In some cases, old components are removed, and new components are introduced as it is not satisfied enough to modify a framework's interface to support new functionality, in such cases it may need rewriting of the component from the beginning.

5. Split product line architecture component:

The software product line architecture could be obtained by breaking out functionality from one
framework into another framework unit. To ensure a component into many forms is to avoid from configuration management problems, but the newly developed component covers more than one logical type of functionality.

6. Deriving New relation between components:

There exists a connection to the previously unrelated components since a relation between components in the architecture. Establishment of new software product line relation between the components occurs due to the new requirements, as it could communicate with existing components.

3. BENEFITS OF SPL ARCHITECTURE

1. Software Architecture gives a basis for analysis of software systems behavior before the system has been built. The ability to verify and satisfy the stakeholder needs without having to build it represents cost saving and risk-mitigation.

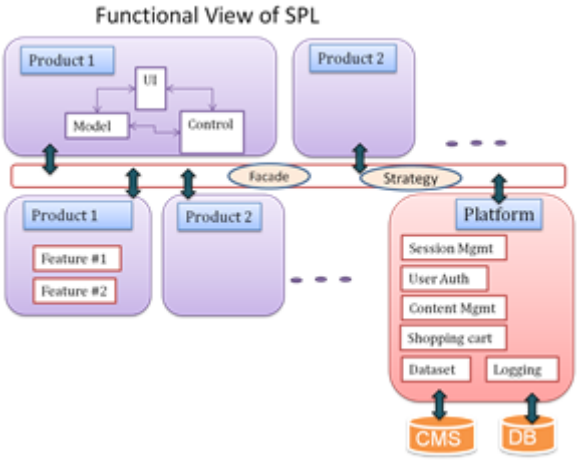

Fig.3. Functional View of SPL

2. Software Architecture provides a basis for reuse of elements from the pre-existing products. A complete software architecture or individual architectural strategies and decisions, can be reused across multiple systems whose stakeholders require similar quality attributes or functionality, saving design costs and mitigating the risk of flaws in the design.

3. Software Architecture supports early design decisions that impacts a system's development, deployment, maintenance. Getting high-impact decisions right is more important to avoid schedule and budget problems.

4. Software Architecture facilitates with stakeholders, contributing to a system to fulfills their needs. Communicating about complex systems from the point of view of stakeholders helps them understand the consequences of the requirements, design decisions based on them.

5. Software architecture helps to reduce risks and chance of system failure.

6. Software Architecture is economic and helps in reduction of cost. Software architecture helps to manage risk and costs in complex projects.

4. SPL Modeling Process

A SPL development process has two main processes

Domain engineering is the process of software product line engineering in which the commonality and the variability of the product line are defined and realised.

The domain engineering process is composed of five key sub-processes: product management, domain requirements engineering, domain design, domain realisation, and domain testing. The domain engineering process produces the platform including the commonality of the applications and the variability to support mass customisation.

Application engineering is the process of software product line engineering in which the applications of the product line are built by reusing domain artefacts and exploiting the product line variability.

The application engineering process is composed of the sub-processes application requirements engineering, application design, application realisation, and application testing..

The framework differentiates between different kinds of development artefacts i.e, domain artefacts and applications artefacts. The domain artefacts subsume the platform of the software product line. The application artefacts represent all kinds of development artefacts of specific applications. As the platform is used to derive more than one application, application engineering has to maintain the application-specific artefacts for each application separately.

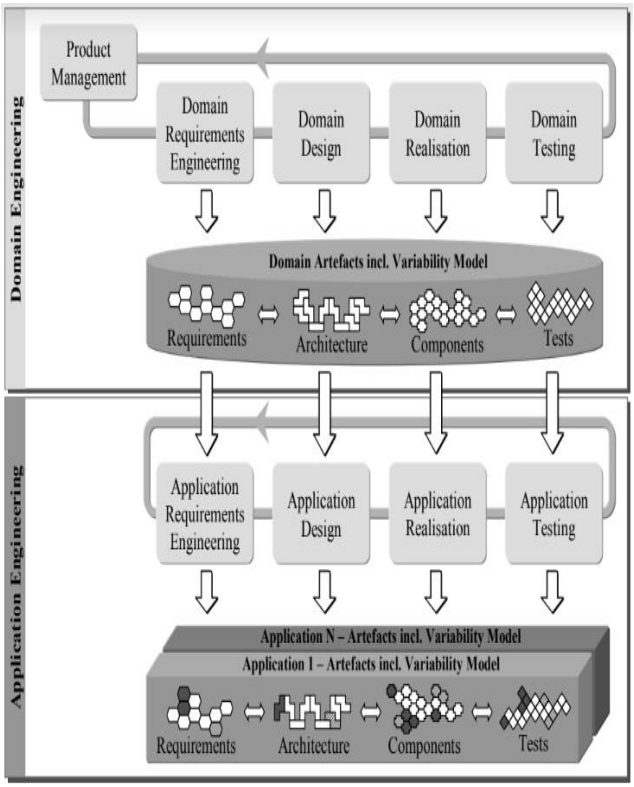

Figure 4.Domain Engineering and Application Engineering

5. CURRENT SOFTWARE PRODUCT LINE APPROACHES

There are many Software product line approaches like respectively FAST, FODA, FORM RSEB, ConIPF, PuLSE.

5.1. FAST (Family-oriented abstraction, specification and translation)

FAST is feature-based model proposed by Weiss. It applies product-line architecture principles into software engineering process. It can be used in those cases where a range of products are being developed which have major shares of common artifacts among themselves. The purpose of FAST is to make software engineering process more efficient by reducing multiple work, by decreasing production costs, and by shortening time-to-market.

In this framework, the processes can be divided into following three sub-processes namely **Domain qualification** under which an economic model of the software product line is generated by cost analysis, **Domain engineering** under which the main agenda is to analyze the commonalities in the potential product line, and then coming up with a family definition and product line infrastructure as well as reusable core assets, **Application engineering** under which the product line family is developed by using the reusable core assets.

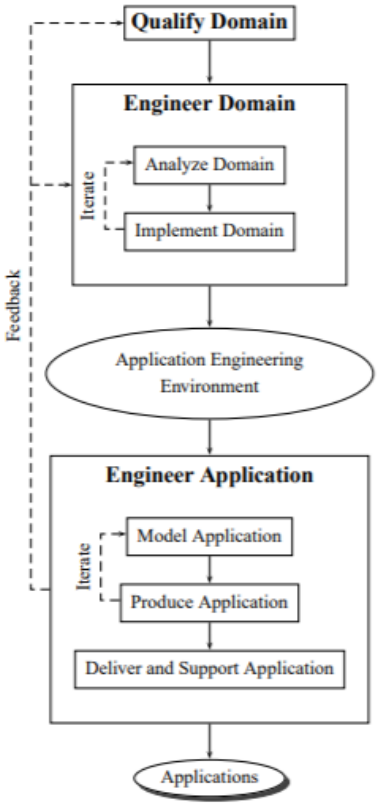

Figure 5.1. FAST flow process

5.2. FODA (Feature-Oriented Domain Analysis)

Feature-Oriented Domain Analysis has been proposed by Kang. FODA identifies and model the features of a software.. FODA identifies the distinct features of a software product line by using domain analysis technique.. It involves three basic processes, namely,

- analyzing of the domain of the product line,
- analyzing the features of the product line , and
- modeling the features of the product line.

Analyzing the domain is to define the domain and finalize the products of the product family. Analyzing the features of the product line is to analyze the features

by performing the commonality and variability analysis.
Modeling of the features is performed as per the core and varied artifacts which helps in developing the product line family which are developed in a structured and smooth fashion.
The three major phases in FODA which guides the success of the process are Context analysis, Domain modeling and Architecture modeling.
FODA uses state activity charts and state charts to model functional and behavioral aspects correspondingly. These charts are proposed by Structured Analysis and Design Technique (SADT).

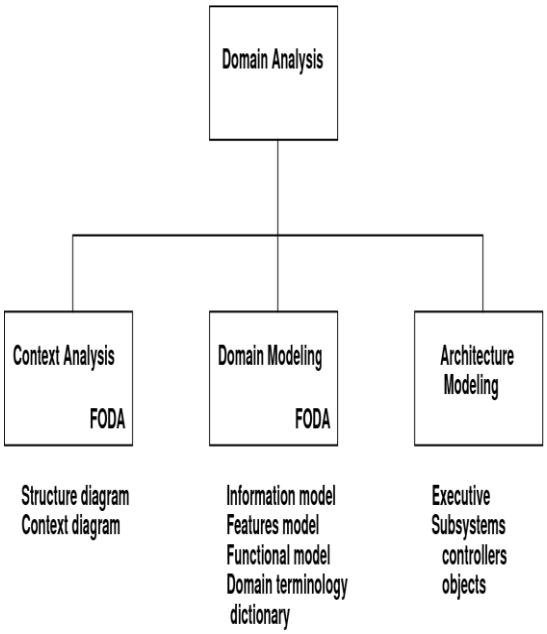

Figure 5.2: FODA flow

5.3. FORM (Feature-Oriented Reuse Method)

Feature-Oriented Reuse Method for product line software engineering is a method based on feature orientation which analyzes the features of the domain, and then use these features to provide the software product line architecture . Reuse of software artifacts is one of the best solutions to the “software crisis”.

FORM is a systematic method that looks for and captures commonalities and differences of applications in a domain in terms of "features" and using the analysis results to develop domain architectures and components. FORM is an extension of FODA to the software design and implementation phases and is used in the analysis of domain features which is further used to develop domain architecture and reusable components.

Once a domain is described and explained in terms of common and different “units” of computation, they are used to construct different “feasible” configurations of reusable architectures. FORM method is specifically used in the domains of telecommunication engineering as well as information technology. However it can be applicable to other specified domains depending on the coherence of the feature model.

5.4. RSEB (Reuse-Driven Software Engineering Business)

Reuse-Driven Software Engineering Business is a systematic, model-driven approach to large-scale software reuse. It is a use-case driven systematic reuse process based on the UML notation. It defines several model-driven software development processes: Architecture Family Engineering, Component System Engineering and Application System Engineering . These processes optimize for robustness and reuse. RSEB uses some part of FODA.

RSEB is an iterative and use-case-centric method which facilitates the development of reusable object-oriented software as well as software reuse. The main focus in this process is on the use cases. In this process, at first we describe the requirements for the product line domain with the help of use cases. Then, secondly the domain architecture and reusable artifacts are designed. Finally, object models are created with the help of these architecture and artifacts which are mapped to the use cases.
The Unified Modeling Language (UML) is used to capture the variabilities which are identified in the use cases and object models. The use cases and object models are structured using variation points and variants.

5.5. ConIPF (Configuration of Industrial Product Families)

Configuration of Industrial Product Families and it is a European FP6 project was put forward by Eriksson in whose words ConIPF is “a project which wants to integrate both the product line approach and the structure-oriented configuration 35 technologies”.
There is more provision for adaptation of configuration methodologies by using artificial intelligence.

As it is similar to such software line approaches development with reuse is the driving principle behind this software product line. There is ample provision for adaptation of configuration methodologies by using artificial intelligence.

5.6 PuLSE

PuLSE is an approach which can be introduced incrementally by augmenting existing software development processes and products with product line specific aspects step by step.
It provides a complete framework that covers the whole software product line development life cycle, including reuse infrastructure construction, usage, and evolution.

PuLSE is composed of three major components i.e, The deployment phase, the technical components and the support components.
PuLSE is used in enterprise applications for deployment of software product line.
PuLSE is modular and customizable: It consists of six technical components that can be selected and instantiated in order to satisfy the needs of specific companies.

6. EVALUATION OF CURRENT APPROACHES AND CONCLUSION

After referring to all those different approaches, we can see that for all the approaches, abstraction level is very high and there is no proper guideline to apply these approaches. The similarity among all the different approaches is that they all follow similar kind of processes in a different way. The starting point in all the cases is context analysis which is then followed by domain engineering as well as application engineering. Exploiting the similarities and variation is one main concern of these processes. However, there are no detailed guidelines for the

application of these approaches owing to the immense level of abstraction.

FODA and FeatuRSEB guarantees to solve the issues primarily in domain engineering phase. On the other hand, FAST and FORM promises to provide a comprehensive solution for all the phases of software product line engineering. However, infact, the domain engineering phases have more attention than application engineering phases in all of these approaches.

In terms of variability, it is observed that some of these approaches use the feature models. Although all of FODA, FORM and FeatuRSEB uses feature models but FODA was first to use it. On the other hand, PuLSE use decision models. FAST method manages the variability in a text format by using commonality analysis.

Software Product Line has been an area of research and innovation for the last two decades. Common assets which lie at the core of this development comprise of requirements, design, architecture, test plans, reusable software components, test cases and other artifacts. Utilizing common assets for product development increases the productivity, reduce cost as well as marketing time. Hence they decrease the overall development effort.

7. Social Media

Social medias like Facebook, Twitter, Yammer and Foursquare are accomplishing basic human desires like communication, sharing of thoughts and have experienced exponential growth in terms of users. Social medias are source of news and a place to build network and meet interesting people all over the world. Social media systems have thus become important means for circulation of information. The architecture of social media for different system is below in the diagram.

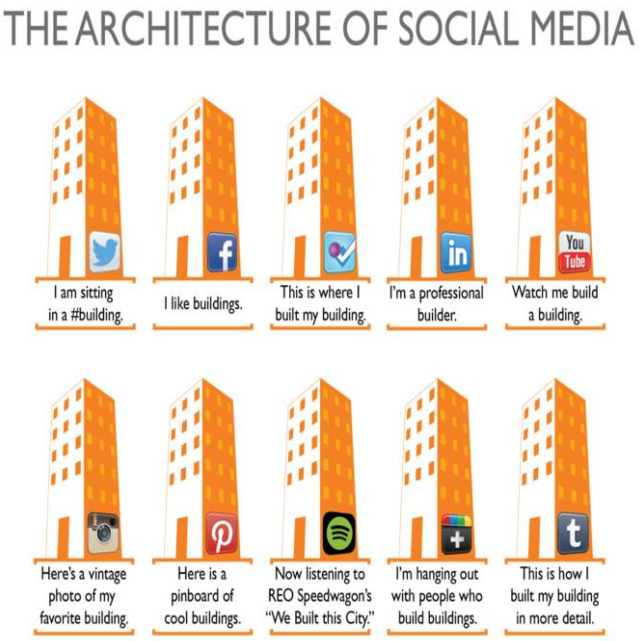

Figure: 7 Architecture of social media

We are going to discuss architecture for below social media:

- Facebook
- Twitter
- Yammer
- Foursquare

7.1 Facebook

Facebook–among the first to publish its API– is currently the most popular SWP(Social Web Platform) with more than a billion users responsible for over one trillion “likes”, over two hundred billion photos and seventeen billion location check-ins. Out of these billion users, six hundred million access the site via a mobile device (BBC, 2012).

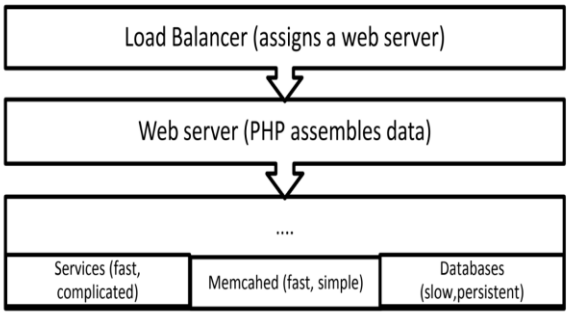

Figure: 7.1 Architecture of Facebook

7.2 Twitter

Twitter as of the mid of 2012 has over half a billion users with about a third of these coming from the United States. The entire backend of Twitter was initially managed using MySQL, but over the years it has undergone several changes and system optimization to meet the drastic increase in load (Eyers et al., 2012) as will be discussed on the subsection on architecture.

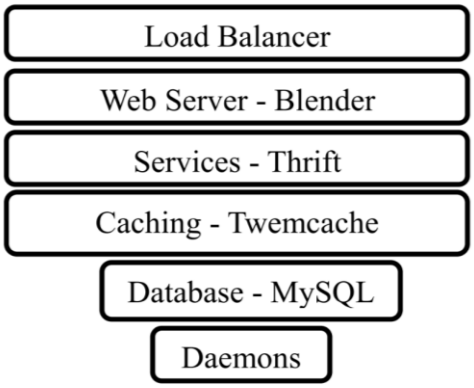

Figure:7.2 Architecture of Twitter

7.3 Yammer

Yammer established in the third quarter of 2008 is an enterprise social network that links employees to content, conversations and business data. It provides a secure and private social network to companies and increases employees' productivity by enabling easier collaboration, faster decision making and self organization into teams to handle business challenges. As networks are private, signing up for a Yammer account requires a valid company email account. Unlike the other SNSs covered here which are free for all users, Yammer is based on a freemium business model; this means it is provided free of charge, but for advanced features a premium is charged (Wikipedia, 2012).

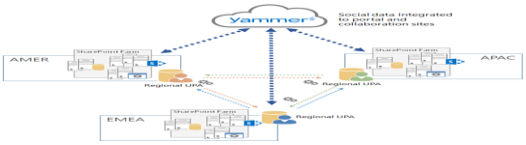

Figure 7.3:Architecture of Yammer

7.4 Foursquare

Foursquare–a social positioning service–(www.foursquare.com) was launched in the first quarter of 2009 and is an application for sharing and saving the places one visits and it provides personalized recommendations and deals (in the case of company offerings) based on where others have visited if they share similarities. Foursquare has over 25 million members who have made over 3 billion check-ins with millions being registered daily (About foursquare, 2012).

8. Software Architecture for Social Media Data Analytics

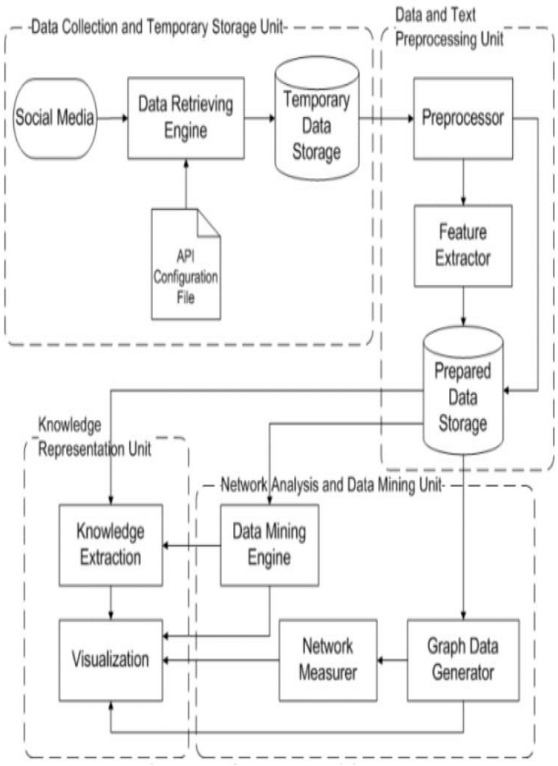

Fig 8.1. Software Architecture

Software architecture is a set of statements that describe the software components and functionality of components of units

.

Data Collection and Temporary Storage Unit for collecting and storing data.

It has Data Retrieving Engine, API Configuration File and Temporary Data Storage. Data Retrieving Engine is used to authenticate (signing in) into social media account with the help of API Key and API Secret which has been stored into API Configuration File. It retrieves data from various types of social media communication according to the configuration of social media APIs and access permissions of third-party applications. It then stores retrieved data into Temporary Data Storage. Data Retrieving Engine built into an abstract class Data Collection that implements the class Facebook API, Twitter API and Instagram API to perform data collection on social media Facebook, Twitter and Instagram. API Configuration File created with flat files (*.INI) that store API Key, API secret and redirect URL. Temporary Data Storage is designed by MYSQL database server.

Data Pre-processing Unit for transforming data.

It has Preprocessor, Feature Extractor and Prepared Data Storage. This unit is used to perform pre-processing data from Temporary Data Storage. Preprocessor perform a variety of data processing techniques to produce a structured social media data that is optimal and perform text pre-processing like case folding, tokenizing, stopword removal etc to get a keyword which is used by Feature Extractor. Feature Extractor used to generate two types of feature extraction and selection techniques using keyword. Prepared Data Storage is used to stores optimized data and selected feature which also designed by MySQL database server.

Data Parsing and Classifying Unit for parsing and classifying data

It has Data Mining Engine, Graph Data Generator and Network Measurer. This unit used to perform data mining process refers to the extraction or gain knowledge from large amounts of data, text mining start from text preprocessing, generating features, feature selection, data mining / pattern discovery and graph measuring Data Mining Engine perform

the various processes of data mining on social media content data by a variety of methods and data mining algorithms. Sentiment analysis is done with rule based classification using wall data from Facebook, tweet from Twitter and photo caption from Instagram. Clustering contact for fun is done by K-Means clustering using profile data from user's contact. Graph Data Generator is used to converts the data into a graph data representation with two types of relationships bidirectional / unidirectional. Network Measurer is used to perform measurements on a graph data to analyze and find the knowledge of the user's networks at social media with various measurement techniques like centrality. Network measuring is done with degree centrality. It will produce the most important person using communication data that generated from tweets and caption that has token mention @.

Knowledge Representation Unit for extracting knowledge and visualizing data.

This unit used to extract information and visualization. It has Knowledge Extraction and Visualization. Knowledge Extraction is used to acquire knowledge and information in a variety of extraction techniques like comparison, query, and others. Top word and popular location are two type of social media analysis that can be obtained by query techniques. Top word generated using bag of word data. Popular location generated using location data from profile data that has been enriched with geo location coordinate.

Visualization is used to visualize the result of data mining, graph data, measured graph data and information extraction result. Clustering result visualized using information panel that shows cluster of users. Sentiment analysis visualized using emoticon to shows whether the wall/tweet/caption is positive, negative or neutral. Top word visualized using chart that shows amount of word and the highest amount of word. Popular location visualized using map. Graph data and measured graph data visualized using graph that shows node (users) and edge (relation) with or without relationship direction (arrow). Visualization is data sink. That means, visualization is the last component that processing data.

CONCLUSION

From the brief study of all architectures among Facebook, Twitter, Yammer and Foursquare it looks like Facebook architecture is better among all for below reasons:

- Facebook architectures are designed in such a way where all open source technologies are being used. As open source technologies are being used company can make software by using free edition. So here company capital is saved from expenditure.
- Facebook architecture is build in such a way that despite of higher number of traffic it can handle the traffic load as it is stateless and distributed, here users sessions are not tied to a particular server and page requests are handled by any of the servers in its infrastructure where as in Twitter it can't handle if traffic load is more. It is often that Twitter stops working due to more traffic load.

References

1. **Clements, P., Northrop, L. Software product lines: Practice and patterns. Addison-Wesley, USA.**
2. **"Product-Line Architectures " by D.Batory Smalltalk und Java in Industrieund Ausbildung (Smalltalk and Java in Industry and Practical Training), Erfurt, Germany, October 1998.**
3. **"Software Reuse Research: Status and Future" (IEEE) by William B.Frakes and Kyo Kang. https://ieeexplore-ieee-org.ezproxy.gl.iit.edu/document/1492369**
4. **"Foundations for the study of software architecture" by Dewayne E. Perry, Alexander L. Wolf https://dl.acm.org/citation.cfm?id=141884**
5. **"A Comprehensive Study of Software Product Line Frameworks" by Md Mottahir Alam, Asif Irshad Khan, Aasim Zafar https://www.researchgate.net/publication/309209280_A_Comprehensive_Study_of_Software_Product_Line_Frameworks**
6. **"Software architecture for social media data analytics" by** Anggi Perwitasari, Saiful Akbar, G. A. Putri Saptawati https://ieeexplore.ieee.org/document/7437000
7. **"Software Architectures for Social Influence: Analysis of Facebook, Twitter, Yammer and FourSquare" http://jultika.oulu.fi/files/nbnfioulu-201304241198.pdf**
8. **"FaceCloak: An Architecture for User Privacy on Social Networking Sites" by Wanying Luo, Qi Xie, Urs Hengartner https://ieeexplore.ieee.org/document/5283227**