17 - 678 MSIT STUDIO PROJECT

Bezirk - Bosch

Summer 2017

Architecture Design Specification

Team

Rajat Mathur

Yu-Lun Tsai

Chandana Kalluri

Version No	Author	Change	Date
1	Chandana	Initial version	Mar 20
2	Yu-Lun	Revise version 1	May 24
2.1	Yu-Lun	Update Functional Requirement	Jun 21
2.2	Chandana	Update Privacy and storage scenario	Jun 22
2.3	Rajat	Comparing processing times	June 23

Business Context

Several applications and services are available in mobile devices and PCs that help gather user information and preferences. For instance, GPS location tracks users movements while e-commerce applications like Amazon can be used to determine a user's taste, likes etc. Social media applications are also functional in assembling user's preferences in music, books, activities, places etc.

Gathering user preferences is helpful in understanding user so as to make predictions and generate recommendations. For example, a user who likes to run every morning at 7 AM and drink coffee at starbucks after, the phone should automatically be able to pull up the weather app and let him know if it's going to rain or if it's too hot outside. Similarly, starbucks should be able to remind the user of a special brew available in the store for that day.

Vendors are interested in collecting these consumer specific information and while customers are keen on enjoying these benefits they are also concerned with sharing their personal data. This is because vendors need to collect all user data from devices, personal computers and put into the cloud in one location to make inferences. Because of this users data is not secure and can be easily hacked. Thus, Bezirk Bosch came up with the Bezirk middleware to facilitate the collection of user data across different devices without sending this information to the cloud.

The Bezirk middleware acts as a smart listener by listening across the different devices surrounding the user both indoors and outdoors, gets the required data and builds models needed by the vendors all on the edge. This ensures that the user privacy is not violated.

Bezirk is interested in developing a normalization framework that will help gather data from different sensor sources and normalize the data in different ways such as time, location, text etc. and abstract it for the inference engines. Normalization is a way to represent the collected data in a unified format. For example America, United States all of them will represented as The United States of America. Since several vendors will be interested in performing different kinds of inferences, the framework allows them to directly use the normalized data without worrying about tailoring the sensor data to their need. Thus, Bezirk is interested in an architecture for the normalization framework that uses Bezirk middleware to maintain user privacy, gather users information and perform inference and recommendations after normalization in an reasonable amount of time so that vendors and customers benefit equally.
The architecture should also be extensible for developers to introduce or add and remove normalization and inference engines as their need dictates. The architecture must also be portable and support only Java APIs.

Functional Requirements

Data Gathering

Create a framework to accomplish collection of user personal data from multiple sources, referred to as sensors. The framework must be able to collect the social media feed from the application users' twitter and facebook accounts. The framework must also be capturing the GPS based location of the application user. Here GPS, facebook and twitter can be regarded as sensors.

The data points collected from the user shall aid at understanding the user's interests. Privacy being a key concern for the project, requires that the best effort be made at preventing the access of this collected data to other services or organizations/persons.

Multiple modes of collection of data

The framework must allow the application developers to choose from multiple modes of operation for the collection of data from the sensors. The three modes required are event-driven, periodic and batch mode for collecting data from the sensors. The application developer must be able to select the event-triggering sensors in the event-driven mode, and so do the other two modes.

An event mode is that the framework pushes a notification if new data is available; otherwise, it will keep silent. A periodic mode is that the framework allows user to specify a period and it will pull the data from the data sources periodically. If no new data are available, it should keep silent as well instead of occupying the bezirk middleware communication bandwidth. The batch mode allows developers to pull historical data from a sensor source as much as it can.

Data Normalization

The framework must normalize the different types of data. Normalization must help at standardization of the data items so that they can be better inferred. External services may be used for normalization. The framework must currently support at least text and location normalization service.

Data Inference

The application framework must support the inference of the normalized data. The inference results would be used by the application to send targeted advertisements to the application users. Framework must be flexible enough to support various kinds of inference algorithms. It must also ensure that previous inference results are stored on the device as they would be used to improve the understanding of the user's preferences. A minimum capability: keyword matching algorithm must be supported as part of the project, which keeps a count of matches with a keyword. The application developer must be provided a mechanism to configure the keywords list.

Availability

The framework must be able to accomplish normalization operation at all time. If the design relying on a specific solution (e.g. a third-party web service) to do the normalization, when there is a failure in network or service provider, the framework must provide a mechanism to bring it back or replace it with other backup solutions.

Quality Attributes

The following tables show a set of quality attributes expected to support by the framework. They are organized in a format that lists six part quality attribute scenario. The order of these quality attributes implicitly reflects the priority of each quality attribute. According to the discussion with our clients, performance in terms of latency has the highest priority.Extensibility is an important attribute as well .Although the rest of quality attributes do not have equal priority as the first three, they should be considered if possible.

Performance (latency): The end-to-end processing time required by this framework. Location data is used to infer the nearby locations of an user. A company or restaurant owner would like to send the users a coupon when they are close to the shops. This discount should be sent timely so that they may stop and purchase something. It must run in the background and fetch user information in a real-time manner. If the latency is too high, the inferred result is not valid because the user may already leave that place.	
STIMULUS	1. The application would like to pull historical data of an user from sensor sources (BATCH_PROCESSING) 2. Our DNDI framework is triggered by an data-available event and perform a series normalization and inference operations (EVENT_DRIVEN)
SOURCE OF STIMULUS	1. The application (BATCH_PROCESSING) 2. External events (EVENT_DRIVEN)
ENVIRONMENT	Runtime (BATCH_PROCESSING and EVENT_DRIVEN)
ARTIFACTS	DNDI framework
RESPONSE	The DNDI framework passes either raw data or normalized data through the Bezirk middleware and perform data normalization and data inference.
RESPONSE MEASURE	The latency should be less than 5 sec under a situation that event-driven mode is selected and a GPS location event occurs.

Extensibility: This refers to the effort required of developers if they would like to add a new Zirks (data gathering, normalization, and inference). This includes even adding new APIs etc.	
STIMULUS	Application developers would like to capture additional information with new data gathering Zirks, process a new data type with a new normalization Zirk, or explore user preference with a new inference Zirks.
SOURCE OF STIMULUS	Developers
ENVIRONMENT	Development phase
ARTIFACTS	The code that developers have to modify or add to the DNDI frameworks. It comprises the I/O operations and callback function implementation.
RESPONSE	Developers change the code and run the new Zirk or algorithm using not more than 4 interfaces (methods).
RESPONSE MEASURE	Modification is supposed to be done in one or two files. The amount of code changes should not be more than 15 lines of code. Developers don't have to worry input and output format and focus on the program logic.

User privacy: It is critical to maintain the gathered user data from different sensor sources to remain on the user's device. All the normalizations, inferences computations should have minimum exposure.	
STIMULUS	The data transferred between different zirks
SOURCE OF STIMULUS	Sensor zirks, Normalization Zirk and inference zirks
ENVIRONMENT	Run time
ARTIFACTS	User data gathered ,normalized data and inferred data
RESPONSE	Encrypt the data when transferring between different zirks
RESPONSE MEASURE	Perform unit tests to check if the data is being encrypted from the sender zirk and the data is being decrypted on receiver zirk.

Storage Efficiency: The amount of space consumed by the framework.	
STIMULUS	A process part of the framework generates data that needs to be persisted for some duration, so as to be used for future reference or as part of logging for debugging.
SOURCE OF STIMULUS	1. Raw data collected from the sensors. 2. Transformed data generated from the normalization phase and yet to be inferred. 3. Data model for future reference.
ENVIRONMENT	Runtime of the DNDI framework
ARTIFACTS	Persistent storage available on the device: SSD, SD Cards, etc
RESPONSE	Save the data in SQLite database..
RESPONSE MEASURE	If the number of entries in the DB reaches threshold value then remove old data using timestamp.

Business Constraints

Consideration	Business Constraint
Time	1. Software product and architecture design document delivery by Aug 10, 2017 2. Continuous delivery is expected by the client
Product	Use the Bezirk middleware
License	Bezirk is an open source platform. The product must be developed following the open source license.
Delivery	Incremental builds for product delivery

Technical Constraints

Consideration	Technical Constraints
Computer operating system(s)	1. The product must be run on android platforms. 2. Architecture design should support porting to other platform that is able to run JVM.
Computer languages(s)	Use Java and related frameworks for development
Commercial hardware or software products	Android mobile phone
Tools and methods	Android Studio, JDK 1.8.0_121
Protocols, interfaces, standards	Bezirk Middleware
Legacy hardware and software	None. Greenfield project

Stakeholders

Application developers

People who import the DNDI framework and plug in their inference algorithms to infer user preference. They may create their own data gathering Zirks and normalization Zirks to capture the information they are interested in. They anticipate the framework should be flexible enough for them to add new Zirks and have a clear configuration flow so that they can focus on inference algorithm expiration instead of spending a lot of time tailoring with a variety of sensor sources.

Customer

Company such as Giant Eagle that would like to explore potential customers by understanding their user preference from a combination of sensors.

The Zirk Team

People who work on creating the software architecture during the spring and summer semester.

End User

People who install mobile applications using the our DNDI as a way to gather user preference. They care about power consumption of apps running on their device.

Comparing estimated processing times for different modes

Modes vs Data-kind	Raw Data	Normalized Data
Batch Mode	**5 mins**	**10 mins**
Event Mode	**1 mins**	**1 mins**
Periodic Mode	**2 min**	**2 mins**

These time estimates are for social feed collection via Twitter and Facebook done as part of the framework.
Significant contributions to the time estimates come from the following components in the critical path:

1. Collection of Raw data from Twitter server in the batch mode. A maximum of 3200 tweets can be collected in this mode.
2. Collection of Raw data from Facebook on page likes in the batch mode.
3. HTTPS request and responses to normalization services for data collected in the batch mode.
4. Data normalization processing done by the Normalization services.

Glossary

The MSIT-ESE Team:
The three (Yu-Lun, Rajat and Chandana) students from Carnegie Mellon working on the project.

Customer:
The retail store, which uses or plans to adopt the framework developed in this project to understand end-user preference.

Application Developers:
Developers who shall be using our the API developed by Team Zirks to draw inference about end-users. These would be employees of the Customers.

Customer Application:
The mobile or desktop application developed using the framework developed by The MSIT-ESE Team.

Platform:
Platform is the ecosystem on which the customer application shall be running: smartphone, desktop or web application. It includes the OS, hardware: storage, memory, processor and battery power capacity.

End-User:
People who use the Android or desktop application developed using the framework. They shall be the one who's data is to be collected and inferred.

Real-Time Processing:
The user data is sampled by the Data Gathering Zirks at a reasonable frequency. The latency from data gathering to a notification should be completed in a minute..

Batch Processing:
For a new user, the existing twitter feeds, facebook data is collected. This data is processed in a batch and the result is sent to the customer application. It can be a one-time activity, daily or as decided by the application developers.

Push Data:
Forward data to the application if available. A proxy will register the streaming API provided by Twitter. This API will forward data if a new post is done by the user. The role of a proxy is like a bridge between streaming API and the application.

Pull Data:
Get the data actively by sending a request or calling a function. The original process will be blocked or it should provide a callback interface until data arrive.

Proxy:
A proxy is a Zirk, which shall be acting as a wrapper for a data source. The proxy shall allow the data source to publish its data over the Bezirk middleware.

Framework/Library/API:
The Java based framework to be developed by The MSIT-ESE Team. This framework would primarily be responsible for data gathering, data normalization and data inference.

Vocabulary:
A set of keywords that is configured by application developers.

Data Normalization:
Normalize a piece of data (time, location, or texts) into a unified format. For example, all kinds of time format are normalized into one format. So do text and location.

Data Inference:
Derive logical consequences on the basis of data collected from a variety of sensor sources

Zirk:
An independent process that uses Bezirk middleware as communication protocol. It can be used to do data gathering and various data transformations .

Middleware:
Software that acts as a bridge between an operating system or database and applications, especially on a network.

Inference Architecture:
A structure that shows how the data flow through different components as a dynamic view. It also includes a representation of how a developer can configure different components.