Software Architecture and Quality

Course Code: PA1410

Name of the system: Magic Voice Weather
Assignment No: 01

<u>Group member name and Id:</u>
1. Adrian Vladu
Social Id: 900521P736
2. Santosh Shah
Social Id: 870424P259
3. Zakaria Mahmud
Social Id: 89011011230
4. Aftri Mariska
Social Id:

Our work can be found also on github: https://github.com/ader1990/PA1410-assignment

Contribution:

Name	Idea	Idea%	Documentation	Documentation%
Adrian Vladu	Factors, Issues, Strategies and conceptual views.	30%	Introduction, Initial Conceptual view and conceptual view.	30%
Santosh Shah	Factors, Issues, Strategies and conceptual views.	30%	Factors, Strategies, Initial view and Conceptual views.	30%
Zakaria Mahmud	Factors, Issues,Strategies and conceptual views.	30%	Factors, Issues and Strategies.	30%
Aftri Mariska	Factors, Issues and Strategies.	10%	Factors and Issues.	10%

Introduction: An easy to use and interactive voice driven software system has a lot of potential these days, as there still exist a large number of cheap simple mobile phones. Those might not be connected to Internet or could be used by people with little technological knowledge for various reasons, from weather information to medical assistance.

The main purpose of the system is to deliver to the user a fast automatic voice response, which is dependent on the user voice or keypad press input. It is supposed to run without whatsoever human intervention. The response has to be easy to understand, the voice interface should be easy to learn and, when needed, it would provide help.

As for the internal system perspective, the voice user interface should be easy to extend with newer data sources, other data transmission systems - VoIP and it has to accommodate more than 5000 concurrent users.

Depending on the user information that can be securely collected and processed, the system has to provide information which better matches his needs.

In the first assignment, our team has to analyze and understand the system, so that at the end of the period, it can produce a proper view of the requirements of the system, factor tables, issue cards and relevant strategies to solve those.[1] Solving the issues has to conclude with the creation of a conceptual view of the system, where strategies are implemented, keeping in mind that there must be traceability between issue cards and strategies.[2]

System Analysis

Organizational Factors:

Organizational Factor	**Flexibility and Changeability**	**Impact**
O1.Management		
O1.1.Skilled manpower	Very little.	-
O2.Staff Skills		
O2.1.Knowledge of OOD	No.	-
O3.Development schedule		
O3.1.We have ample of time	-	-
O4.Development budget		
O4.1.We have enough budget.	-	-

Technological Factors:

Technological Factor	**Flexibility and Changeability**	**Impact**
T1.General purpose hardware		
T1.1 End device storage	It might get higher than 30MB	Low impact
T2.Domain-specific hardware		
T2.1.High load	The user number might increase to more than 5000	High impact
T3.Software technology		
T3.1.Third-party extensions	very likely - purpose/implementation - algorithms can change and get better over time	High impact
T4.Architecture technology		
T4.1.The system works without human intervention	System will work according to its natural creativity	No impact
T5.Standards		
T5.1.Standards will change between the components	medium	Considerable impact

Product Factors:

Product Factor	**Flexibility and Changeability**	**Impact**
P.1.User interface		

P1.1.User voice interface	It may change depending on the area/country	High impact

P2.Functional features

P2.1 Maintenance	Maintenance is very important for the systems	High impact

P3.Performance

P3.1.Fastest output	Fastest output can make customer more satisfied	Considerable impact.

P4.Dependability

P4.1.Should be dependable	Likely	Low impact

P5.Failure detection, recovery

P5.1.Recovery system	not much.	Low impact

P6.Service

P6.1.Serve customers	serve the customer according their expectations	Low impact

P7.Product cost

-	-	-

Issues and strategies:

High Load

Issue

The product has a requirement to support 5000 users. Though it is not very high it is likely that load will increase over time.
Implementing factor
High load - (Technological factors)

Solutions: It can be solved by following strategies:
Strategy 1:
Make the system run on powerful hardware so that it can handle huge load.
Strategy 2:
Make the system distributed so that it can distribute load.[3]

Error Handling

Issue
Product can and will get undesired input. These input may origin from various sources. One of the main reason would be fetching weather report for a place which doesn't exist .
Implementing factor
Wrong input (Technological factor)

Solution: It can be solve by either of the two strategies.
Strategy 1: Filter out the strategy as early as possible and end the dialogue with the caller once the product gets one.
Strategy 2: Filter out the strategy as early as possible and help the caller. Help could be in many ways. One of the basic and most desirable way would be provide the menu option.

Issue
Without distinguishing user input and system output, the whole process will be useless
Implementing factor
Distinguish between user input and system output(Technological factor)

Strategy: Efficient system
Make the system powerful and intelligent so that it can make a difference

Issue
Any time any problem can occur
Implementing factor
The system works without human intervention(Technological factor)

Strategy: Handling problem
Make the system much more efficient so that it can handle any problem

Issue
Third-party extension can be risky because of some security issue
Implementing factor
Third-party extensions(Technological factor)

Strategy: Secure system
Develop the system with high security so that it will be safe for the third party. This involves creating a set of well-defined secure API-s, so that third-parties can easy plug in into the system.[4]

Issue
It can make the system very confusing for the user
Implementing factor
User voice interface(Product factor)

Strategy : System with artificial intelligence
Develop the system with artificial intelligence so that system can identify any voice interface

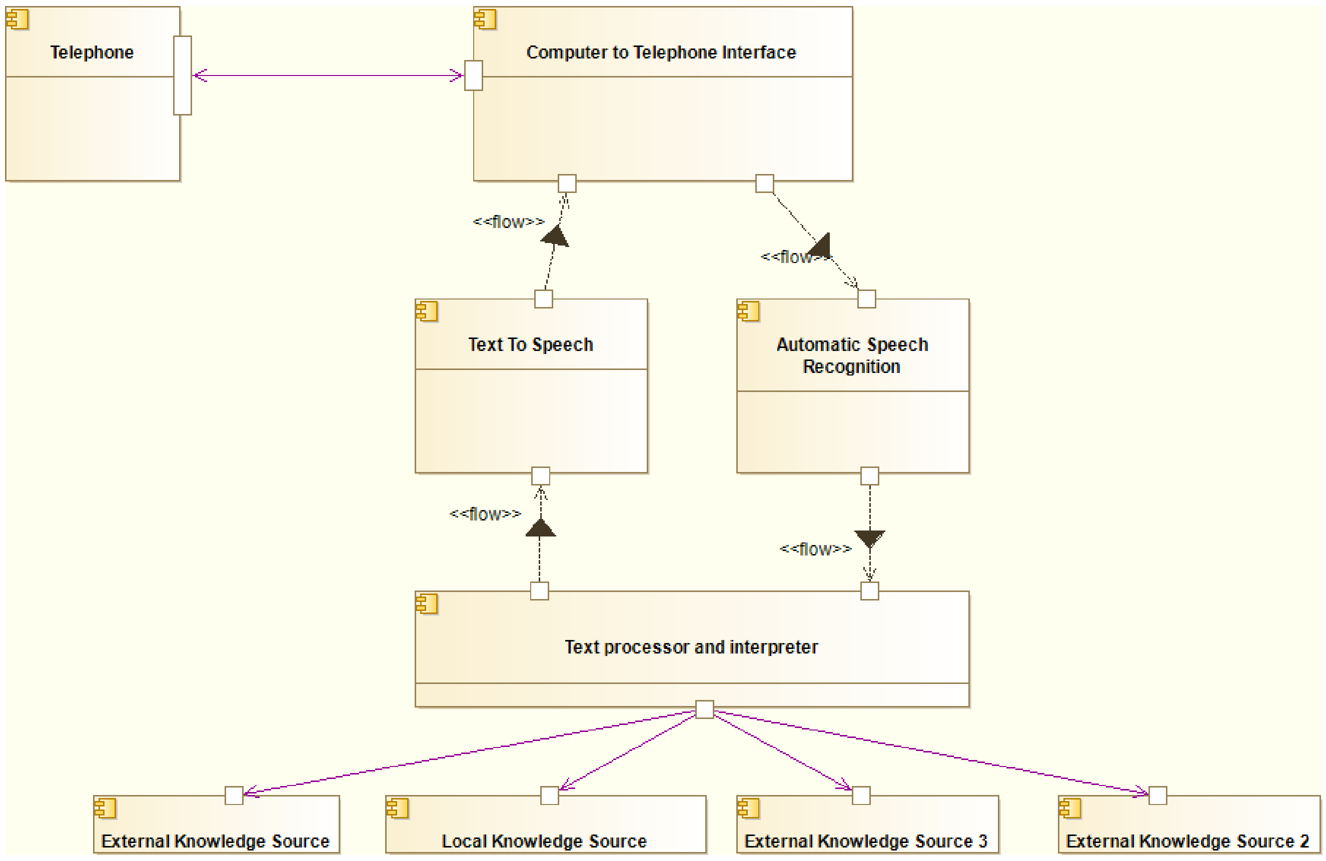
Issue
Customers expectation is always high compare to systems capabilities
Implementing factor
Customer expectations(product factor)

Strategy: Customer friendly service
Make a survey and then try to develop the system as close as the customers expectations

Issue Lack of proper maintenance make the system useless for the customer
Implementing factor
Maintenance (Product factor)

Strategy: Maintain system
Maintain the system in regular basis to satisfy the customer

Preliminary architecture style diagram(we used Modelio 3.0 for drawing the diagrams):


Telephone
Computer to Telephone Interface
<<flow>>
<<flow>>
Text To Speech
Automatic Speech Recognition
<<flow>>
<<flow>>
Text processor and interpreter
External Knowledge Source
Local Knowledge Source
External Knowledge Source 3
External Knowledge Source 2

Preliminary Architecture Style Documentation

The preliminary architecture style is focused on a first top view of how the system can be designed/implemented taking into consideration the requirements.

All the components involved will respect the SPR and will thought of as software services - this way changes in one module should not interfere with the prospect of changes in the other ones.[5][6]

The following components are thought of to reach the goal:

1. The mobile device which the user has. It is the user hardware interface. From our perspective, it is a black box which has memory, the ability to send data and receive data to/from the telecom company system. The data may be transmitted using normal telecom protocols or VoIP.[7]

2. The telecom company system who provides communication services. It is a black box which relays data from the user mobile phone and back. The data is transferred using telecom/VoIP. This black box can abstract the method used. The interface provides methods to get raw user data and send to the user raw data in the same single session.

3. The Automatic Speech Recognition system should take as input the raw data from the telecom system and transform it into relevant text. The string result shall be passed to the Text Interpreter system.[8]

4. The Text To Speech shall take as an input a string, transform it in raw audio data and pass the result to the telecom system input stream.[9]

5. The Text Interpreter system shall receive text and map it to relevant data using the knowledge sources available. The string result shall be sent to the Text To Speech system.

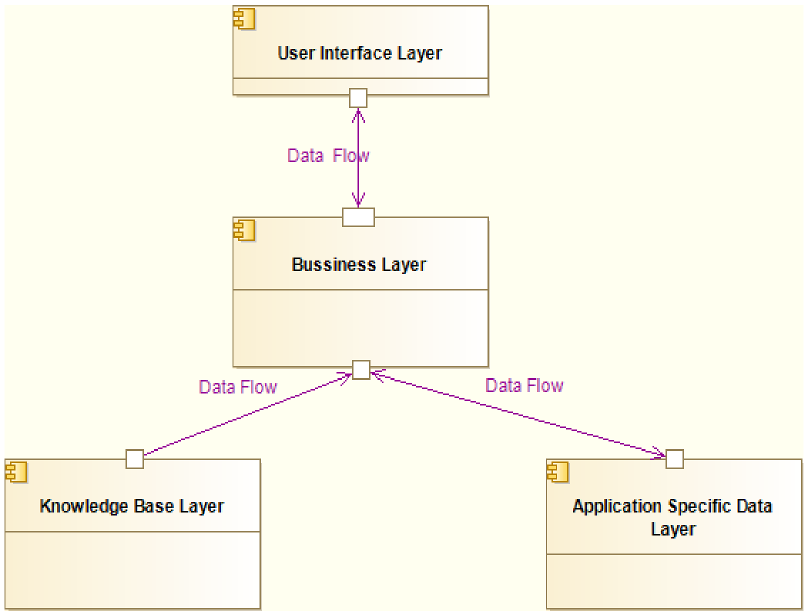
6. Knowledge sources - these sources may be local or Internet based data sources of relevant information. These should have a common interface for querying data. Example: weather information providers, medical assistance, etc.

The components above must exchange messages that can be differentiated between different user sessions.

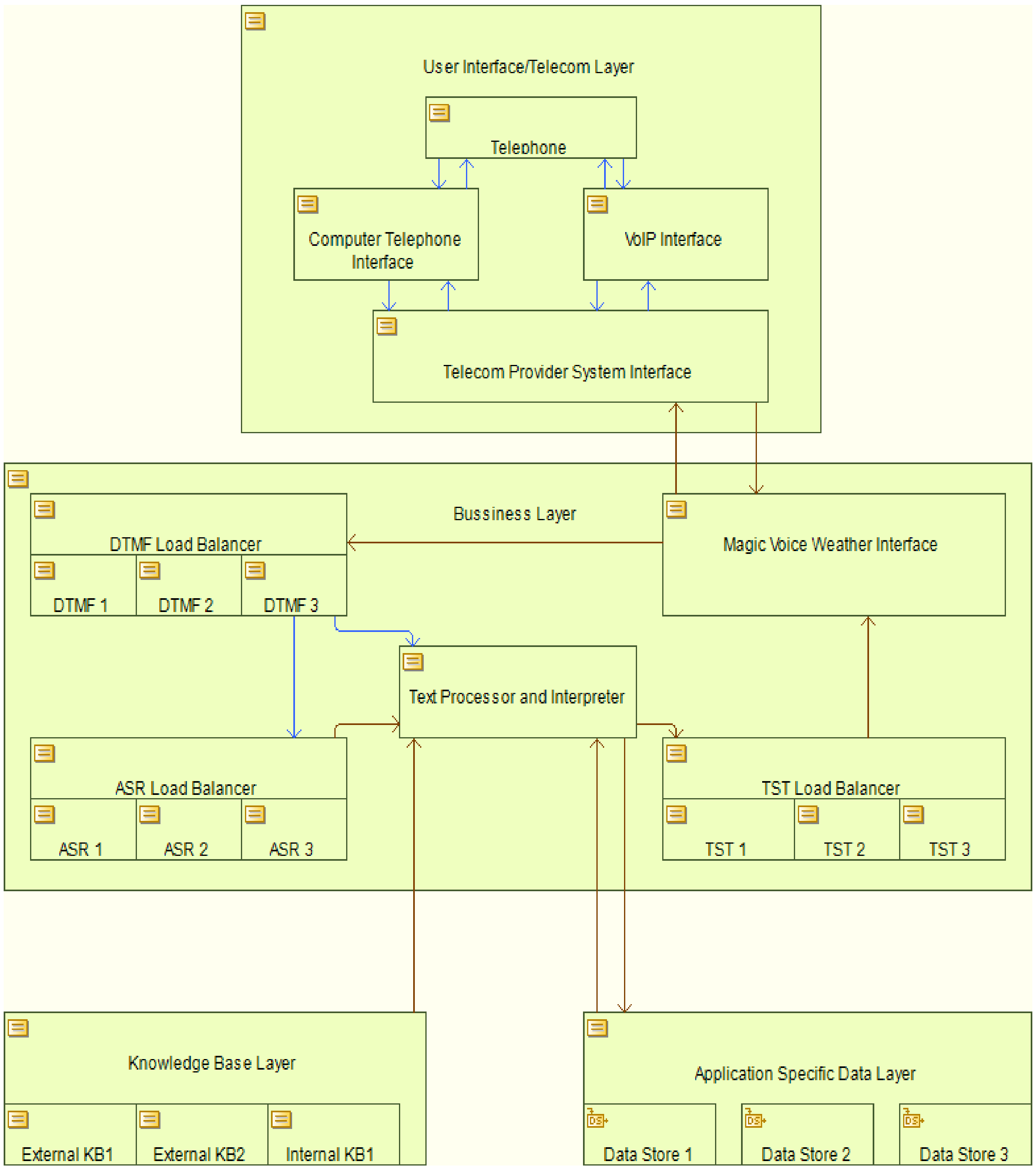
It may be the case that between the Telecom system and TTS/ASR should be a load balancer.

Conceptual Views:

Simple Conceptual View



Full Conceptual View:



List of abbreviations:

PSTN	Public Switch Telephone Network
CTI	Common Telephone Integration
DTMF	Dual Tone Multiple Frequency
TTS	Text To Speech
ASR	Automatic Speech Recognition
SRP	Single Responsibility Principle

Conceptual View

The overall view has the combination of different design principles and patterns to solve issues that came up during the global analysis. Some of them(with the reasons why they have been chosen) are listed below:

- Multi-Layered Architecture:To divide the entire Architecture in higher groups so that it becomes easier to identify the place for modification and improvement whenever necessary. For example: if there is a requirement to change the message given to the client,we can make direct change to Data layer.
- Distributed System: To balance load.
- Service layer: To be able to integrate with third party components seamlessly.
- Module based: To decrease cohesion so that change in one place doesn't affect others.
- Object oriented: For reusability of the code.
- SRP for maintainability.

User Interface/Telephone layer:
This product can get input from different sources . For e.g. : Mobile, PSTN etc. It is then passed to CTI. CTI is responsible for routing the call to Business layer.[10][11]

Business layer:
The message from CTI is received by service interface . This interface accepts input which follows certain standards and are in certain format. After this they are passed through one of many DTMF. DTMF is responsible to determine if the input by the user is from tone or not.[12] If the input is from tone, it goes to processor and interpreter else it goes to one of many ASR through service interface and then to processor and interpreter. ASR is responsible to translate speech to text.[8] Processor and interpreter has the logic to manipulate data and act accordingly. For example: if the user is looking for menu option, it should directly move towards the menu . During this this is no need to check the weather information.
After the information is processed it has to be translated to speech so that user can get a reply. TTS is responsible for this task.

Data layer:
It consist of everything from where we can access information. It can be from local sources like local database or from other applications/services.To facilitate the the transfer of information between these third party applications service interface layer is used.
Error handling:

Error are handled by the product in a centralised way. Error can come up when the user ask for the weather information of the city whose information are not present or by entering wrong city code. In this case we might ask the user to enter the correct code or end the call. In any case our architecture looks up the information in data layer through the business layer .

Application Specific Data Layer
The data about the users, logs and other information have to be saved so that it can be easy to retrieve. This might be the case using databases, files or other persistence methods.

Alternatives to the design

1. Distributed architecture can be replaced by centralised powerful server(powerful hardware). It makes much sense when the product doesn't have a large number of users. Our product doesn't use it because we think it is very likely that the number of user will increase over time.

2.The product would have been much more efficient without the use of interfaces, module based architecture, SRP principles etc. Tightly coupled system are the the most efficient systems . But it is very hard and requires a lot of effort to accommodate change in this system. Changes are inevitable and, besides that, ASR has not yet reached its maturity and it is bound to change more than others.[13]

3. The product will be more robust if there is a system administrator management interface, so that an administrator can change factors like allowing access only to some users depending on the country, phone number or make runtime knowledge source mappings to certain users etc.

Conclusion:

Designing the software architecture of the system above has given us a very good insight on the problems that we have to solve to ensure the system may evolve over time, is easy to maintain, and above all, meets the user requirements.
Coming up with a very good solution may interfere also with the limited time and also with the financial resources at our disposal.
A good architecture can evolve to a better one, over time, if we get better knowledge of the system and user requirements - it comes with experience.

References:

1. L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, Third Edition. Addison-Wesley Publishing Co., Reading MA, 2012.

2. C. Hofmeister, R. Nord, and D. Soni. Applied Software Architecture. Addison-Wesley, Reading MA, 2000.

3. http://en.wikipedia.org/wiki/Load_balancing_(computing)

4. http://en.wikipedia.org/wiki/Application_programming_interface

5. http://www.oodesign.com/single-responsibility-principle.html

6. http://en.wikipedia.org/wiki/Service-oriented_architecture

7. http://en.wikipedia.org/wiki/Voice_over_IP

8. http://en.wikipedia.org/wiki/Speech_recognition

9. http://en.wikipedia.org/wiki/Text-To-Speech

10. http://en.wikipedia.org/wiki/Public_switched_telephone_network

11. http://en.wikipedia.org/wiki/Computer_telephony_integration

12. http://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling

13. http://www.fifthgen.com/speaker-independent-connected-s-r.htm