Intelligent Services Development

John Sum
Institute of Technology Management
National Chung Hsing University
Taichung 402, Taiwan

September 8, 2020

Contents

List of Figures

List of Tables

1 Introduction

To develop an intelligent service, it implies the application of various technologies (either intelligent or non-intelligent) in the service. In contrast to a core AI/ML technology development which is basically a research project, an intelligent service development is an application system development project. In computer science terminology, it is also called a software engineering project. Starting from the imagination on the use of a future service, the software engineers eventually make the system working just like the imagined system.

1.1 Brief Concept on Service Development

Here, I would like to stress that the meanings (resp. concepts) of service development, service system engineering, system development, system engineering and software engineering are all the same. They are the frameworks (methodology) outlining the step-by-step of how to make a thing work. These frameworks are more or less the same a five-stage model, including (1) analysis, (2) design, (3) implementation, (4) testing and (5) maintenance & review.

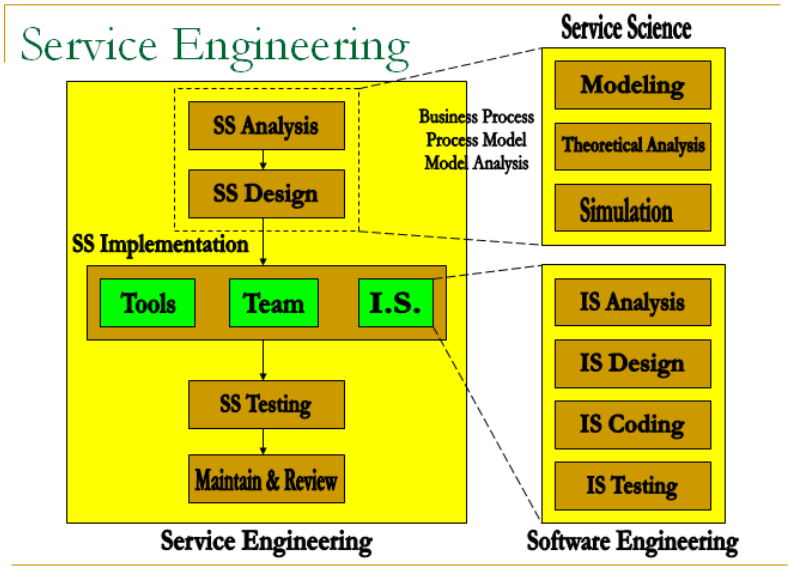
1.2 Service Engineering

Figure 1 illustrates the idea behind this five-stage model for developing a sophisticated service system. Like auto pilot system, the intelligent service system is just a part of it. Intelligent control system would be another part. It is a tool to be built to work together with the intelligent service system. The pilots are the members of the team to interact with this auto pilot system.

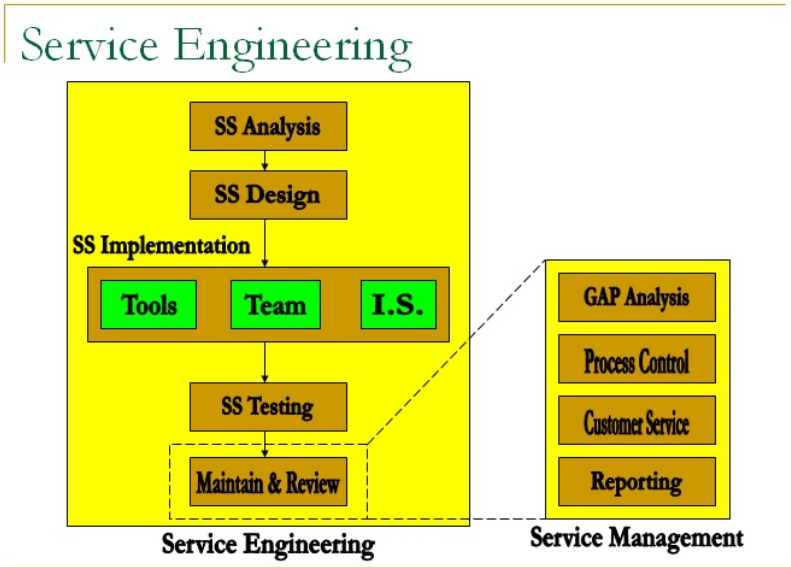
For a simple intelligent service, the framework could be re-stated as another five-stage model – (1) analysis, (2) design, (3) coding, (4) testing and (5) maintenance & review. If you already have an idea of the use of the intelligent service, the framework will be reduced to a four-stage model – (1) design, (2) coding, (3) testing and (4) maintenance & review. No matter under the five-stage or four-stage model, the **analysis and design stages are the most important stages** in the framework. In the rest of the chapter, the key concepts behind each of these stages will be elucidated. Before that, let me introduce driving forces for a new service.

2 Driving Forces for a New Service

If we do not talk about making money, there are two main driving forces behind the development of a new service – a personal desire and an attempt of the application of the new technology. One more driving force for a new product or a new service is **for fun**. All around the world, there are many technical gurus who are leading programmers. Sometimes, they would develop some services for the public to download. Some of them are games. Some of them are simple AI services, like online chatbot.



(a) Service Science & Software Engineering



(b) Service Management

Figure 1: Conceptual framework of service system engineering.

Table 1: Desire-driven new prodcut/service development. The desire generator is also a developer of the new product/service.

Person	Desire	Product (Release)
Henry Ford	Affordable car	Moel T (1908)
Steve Jobs	Affordable computer	Apple I (1976)
Steve Jobs	GUI-command of an OS	Mackintosh (1984)
Tim Berners-Lee	Information sharing	HTTP (1990)
		HTML (1990)
		Web Browser (1990)
		WWW (1991)
Linus Torvalds	Portable OS	Linux (1991)
Jerry Yang	Directory for web sites	Yahoo (1995)
L.Page & S.Brin	Better searching engine	Google (1998)
Steve Jobs	Good music-listen experience	iPod/iTune (2001)
Steve Jobs	Self-developed web browser	Safari (2003)
Steve Jobs	A device can do many things	iPhone (2006)
Mark Zuckerberg	A platform to see friends	Facebook (2006)
Steve Job	Hands-free iPhone	Siri (2011)

2.1 Desire-Driven

For instance, my assistant *Jorman* who is able to make hypothesis and conduct research by himself. This is my desire on *Jorman* who should do those things for me or for himself. Development of a desire-driven new service normally involves a number of core technologies to be developed. That is to say, to develop a desire-driven new service could foster the development of other core technologies. These new services, in the end, will lead to other technology firms to develop similar services. Here, I can give you a few examples from Apple, Yahoo, Google and Facebook the products/services they developed are entirely based on someone's desires. Table 1 depicts some notable products or services their developments were initiated by personal desires.

2.1.1 Steve Jobs

Apple has been developing many phenomenal products/services along this line – should be based on the personal desires of Steve Jobs. The Mackintosh with a graphical user interface-based for commanding the MacOS is one breakthrough product Steve Jobs would like to have. Afterward, Bill Gates copied the idea and then developed the Windows Operating Systems.

The functionalities of a iPod with the iTune platform for music download and the new payment scheme for song listening in the music industry eventually revolutionized the music industry and song listening in the early 2000s. Afterward, other tech firms copy the model and develop similar platforms. One

example is KKBox.

iPhone, a phenomenal product integrating multiple functions in one device, is definitely another product to be mentioned. Telephone call becomes a service in the device. The browser becomes a service for Internet access. Song listening in the same manner as using iPod becomes another service. Once iPhone was released in 2006, android phones and other smartphones were developed.

Siri, likely is from the desire of Steve Jobs, is another phenomenal product. It allows the user to interact with the phone by voice, instead of text and click. The use of the iPhone can be hands free.

2.1.2 Jerry Yang

For every graduate student in computer science, he/she has certainly to conduct an independent research and then compile a master thesis or PhD dissertation for graduation. Here is a story about Jerry Yang, a co-founder of Yahoo. **I do not know how true it is.** While Jerry Yang was a graduate student in the Department of Computer Science at Stanford, he would need to conduct search as other graduate students. The first step to conduct a research is clearly to conduct survey on the technical reports in an area, say AI.

In the early 1990s, web search engines were not so mutual[1]. Nevertheless, many computer science departments archived their technical reports in FTP servers for download. The list of technical reports was included as a browsing paper on a platform called Gopher[2]. Gopher platform is a predecessor of departmental web pages for releasing department information. As the use of Gopher is rather technical, almost only a department in the school of science or engineering would prepare Gopher platform for the department.

To browse and download relevant technical reports, one needed to visit every department in the Gopher system, browsed the list of technical reports and then downloaded the relevant reports. Clearly, it was not that convenient and time consuming. Thus, Jerry Yang developed a searching engine with a directory indexing those pages. Once a user has keyed in a keyword, the system get all the relevant information (i.e. reports) from the indexed pages. The motivation to develop the searching engine is based on the desire of Jerry Yang.

Later, the searching engine was not limited to search for technical reports. The engine could search for webpages over the Internet. Afterward, a number of search engines were developed in accordance with the functions. As far as I remember, there were at least search engines available in the middle of 1990s. In the middle of 1990s, it was known that the performance of each of these search engines was doubtful. A large amount of irrelevant pages were listed. Even meta-search engine had been developed in that period of time, it did not improve much.

[1] `https://en.wikipedia.org/wiki/List_of_search_engines`.
[2] `https://en.wikipedia.org/wiki/Gopher_(protocol)`.

2.1.3 Larry Page and Sergey Brin

In the end, the co-founders of Google Larry Page and Sergey Brin, while they were PhD students in Stanford, developed a new searching engine with a new page-rank algorithm. Before that, the calculation of the relevancy of a web page to the keyword was based on the occurrence of the related words in a page. Then, the pages were ranked by their relevancies. In the end, many search engines reported millions of pages. Many pages listed on top were higher irrelevant.

Thus, PageRank was proposed. It demonstrated that the algorithm could improve significantly the relevancies of the search results as compared with Yahoo and other searching engines in that period of time[3]. The name of the algorithm is called the PageRank [1, 2, 3]. Clearly, their motivations are similar to the motivation of Jerry Yang – to get a better search results for researches.

The idea of PageRank algorithm is rather simple. It just involves a problem to solve a matrix equation. However, realization of PageRank to thousands of millions of webpages is a difficult. To rank thousands of millions pages, the algorithm needs to calculate the inverse of a matrix of size $N_p \times N_p$, where N_p is the total number of pages (i.e. thousands of millions). The run-time complexity for getting the inverse of this matrix is in the order of $\mathcal{O}(N_p^3)$. One can imagine how complex the problem is. No other choice, a distributed computing platform with lot of computers is needed. So, a side product of this PageRank algorithm is the advancement on the **distributed computing technology**.

Nevertheless, the design of the web server to support huge amount of user requests is another terrible problem. Once the Google search engine has been so popular, the web serve has to handle thousands of millions of searches within an hour or even a minute. In this regard, a network of servers is the only solution. For a tech giant, this problem can readily be solved by buying more computers to build the network of servers. Google took the same idea but different approach. Google bought second-handed computers instead of new computers. So, Google engineers needed to solve one problem – how to ensure that the computers while connected together will perform without any problem. In the end, many **server clustering technologies** and **fault-tolerance computing technologies** were developed.

So again, a simple personal desire changes the world of technologies development. Not just the service can be delivered for the users without problem, but also the new technologies developed to support the service change the world.

2.1.4 Mark Zuckerberg

Mark Zuckerberg would like to know how his friends (resp. ex-friends) are living recently. Because of certain reasons, one might deny to make a phone call or send an email to them. Without having direct contacts, like making phone calls and sending emails, In the end, he developed the social network platform 'Facebook'. Social network application was not new in his time of development.

[3] `https://en.wikipedia.org/wiki/List_of_search_engines`.

However, his Facebook changed the world in connecting peoples. No doubt, it becomes the platform with the largest number of members as compared with other social network platforms.

Note that Facebook is the first social network platform in the history. There are many. Some of them were developed even far earlier than Facebook[4]. Some of them focus on certain target groups. Some of them are popularized in certain countries. Some of them have no registration fee. Some of them have. Only very few of them could now compete with Facebook.

2.2 Technology-Driven

Some other products/services in the last decades were developed due to new technologies appeared in the market. The developers (resp. the technical firms) simply assembled the technologies together with the existing technologies to deliver new services. The e-commerce website development and the application services development were two notable examples in the 1990s. The developers did not know much about how the customers accessing the website and how the clients accessing the application services. In the end, many startups were bankruptcies or acquired followed by the dotcom bubble.

Here lists a few exemplar dotcoms which were finally bankruptcy or being acquired by other giant tech firms. They had foreseen the advance in Internet technology and the web technology, and thus developed applications accordingly.

2.2.1 CommerceOne

CommerceOne was used to be a superstar e-marketplace for a merchant to search for a supplier to outsource the product manufacturing. The platform provided a number of functions for merchants and suppliers, like supplier search, request-for-quota (RFQ) & contract signing, supply chain management (SCM) and customer relationship management (CRM). Moreover, CommerceOne had many hubs around the world. In the late 1990s, CommerceOne had been put as a case study in almost every e-commerce textbook.

This Internet technology-driven platform ended up with failure. There are a few possible reasons for its failure.

- One reason is that CommerceOne focused on developing application systems for its clients. CommerceOne did not develop core technology for its application systems. Thus, its model could easily be replicated and its market share dropped as the rivals came.
- CommerceOne did not realized the actual practice in B2B business. CommerceOne did not know what its clients want. So, the functionalities of the application systems might not fit for the clients' needs.
- While a number of firms in the automobile industry had partnered with CommerceOne to build a procurement platform, this platform could soon

[4] `https://en.wikipedia.org/wiki/List_of_social_networking_websites`.

be obsoleted as the automobile giants could developed the same platform for themselves. As a result, the value of CommerceOne will definitely drop.

- Once a merchant has contacted a reliable supplier, the merchant and the supplier will abandon the use of the platform for further connection. Instead, they will arrange physical contact for the future collaborations.

Today, many giants like Apple and BMW have already developed their own platforms for SCM and CRM. Many application services providers (ASPs), like CommerceOne, were failed during the dotcom bubble.

2.2.2 Ariba

Ariba is another technology-driven tech firm founded in the 1990s. It was a rival of CommerceOne in the 1990s. It has developed may application systems for SCM and CRM. It was also an e-marketplace for matching merchants and suppliers. Ariba had any new technologies developed to support such mix-and-match application. However, Ariba faced the same problems as CommerceOne.

Ariba core skill lies in the application systems development. These systems were expected to be useful for a large size of clients. Here is a dilemma. **If a system is developed for a large size of clients, the system can never fit for any one of them.** Nevertheless, Ariba did not realize the practices of the merchants and suppliers. Once a merchant has contacted a reliable supplier, the merchant and the supplier will abandon the use of the platform for further connection. Instead, they will arrange physical contact for the future collaborations.

Thus, the application systems developed by Ariba could have the highest values if they were customized to a few big merchants, but not the small and medium size enterprises (SMEs). Finally, it was acquired by SAP in the 1990s. Now, it is a division of SAP.

2.2.3 PaySafe

PaySafe is a Taiwan-based on-line payment service provider. It is a team under the Taishin International Bank[5]. The key technology of the on-line payment was a self-developed secure electronic transaction system. This on-line payment system was integrated with the Taishin backing system. Generally speaking, it was a reliable on-line payment system and it should attract large amount of on-line shopping merchants to subscribe the service. So, PaySafe could generate revenue from the on-line payment transactions in on-line shopping.

In the end, it failed. One reason, similar to that of CommerceOne and Ariba, is that PaySafe did not know what the customers' needs. The customers needed more than the on-line payment service, even it was developed by using top-notch technologies. They needed to know how to use the service to do

[5] `https://en.wikipedia.org/wiki/Taishin_International_Bank`.

businesses with their customers. In this point, different merchants could have different practices for their customers. So, the actual needs of one merchant could be very difference from the other. A single platform with fixed number of services could hardly make every merchant satisfy. **PaySafe did not know the needs of the customers of its customers**.

2.3 Customer-Driven

It is also an important driving force for a new service development. However, in term technological niche, the new system normally has little or no niche as compared with the existing systems.

2.3.1 Network Security System

For instance, a firm would like to upgrade the network security system. The firm simply calls for a tech firm to provide solution. The solution is an already made network security system. Customization might have to be done to integrate the new system to the existing servers connecting to the local area network. Note that customization has nothing to do with technological advancement. The developer of a customization project does not have to do any technological advancement.

2.3.2 Airline Check-In System

Let me have another example. An airline would like to have a new **global check-in system** to streamline the check-in process. The system is able to let the customer to self check-in through the Internet. The system is able to streamline the check-in process, so as to relive the workload of the ground crew. The administration staffs in the office could monitor the global check-in status through the system.

For this project, the developer will have to contact a French-based airline check-in system vendor for the system, the **backbone system**. The developer has also to understand how the customer, the ground crew and the administration staffs use the new system. The second task is quite airline specific.

Today, we can do the check-in at the airport by ourselves. If we do not have any luggage to be checked, we can simply walk to a check-in kiosk, show the passports and select the seats. The kiosk will then print the boarding passes.

2.3.3 No New Technology Developed

The above examples are going to highlight one point. Usually, a customer-driven project is a system integration project with customization. Developer does not have to introduce new technology in the application system. Moreover, scale of a customer-driven project is usually quite large. Take the airline check-in system for instance, the project has to take years to complete, from analysis to testing. The budget for sure is very high.

3 System Analysis

No matter which force driving the development of a new service, the first question needed to answer is about the **requirement specification** of the **new service or new system**. The second question to be asked is about the **technologies requirements**. The third question is about the **budget**.

3.1 Requirement Analysis

If the system is developed for someone's desire, it is normally a relatively simple project as the desire generator can tell what are the requirements of the system. If the desire generator gets involved in the development, system development would be easier.

The most difficult part is how to get the system requirements from the **customers**. If the system is proprietary, the system requirements can be gathered by a number of meetings with the client firm. If the system is developed for more than one customers, like the application services platform, getting system requirements will be even difficult. The developers have to imagine how the **future customers** are going to use this system. To get the requirements for a system of this type is even difficult.

3.1.1 Understand the Customers of Your Customers

Before the dotcom bubble, many application services providers (ASPs) developed application services to be deployed for the small and medium size enterprise (SME) sector. During the dotcom bubble, almost all of them de-functioned. The others were acquired by giants. In that period, some people blamed the economic crisis in 2000. However, it is not the only reason. From my point of view, another possible reason is that many ASPs had overlooked the following principle.

Principle 1 *A tech firm needs to know (i) the needs (resp. computer competencies) of the customer and (ii) the needs (resp. computer competency) of the customers of that customer.*

Most failure tech firms are suspected that they never concerned on the needs of the customers of their customers. Nevertheless, they ignored the computer competencies of their customers, and the customers of their customers. As the application services were delivered for a customer to streamline his/her business operations, the customers of the customer would also the users of the services. A system developer needs to help them to identify what they need. Moreover, the developer has to help them to identify what their customers need.

However, many ASPs simply marketed their so-called high-tech services to the potential customers. The actual needs of the potential customers and their customers were always missed.

Table 2: Roles and responsibilities of the four parties.

Job	Airline	USA	India	Philippine
Contract	Yes	Yes	–	–
Requirement	Yes	–	Yes	–
System Design	Yes	–	Yes	–
Coding	–	–	Yes	Yes
Testing	Yes	–	–	Yes
Maintenance	Yes	–	–	–

3.1.2 Requirement Gathering

Thus, one task to be accomplished in system analysis is to get the system requirements. If the system is initiated by a desire generator, getting the requirements is easier. If the system is a kind of technology-driven, getting its requirements will be complicated. If the system is customer-driven, getting the requirements will be even complicated and timely.

Like the airline check-in system, the time to gather the requirements is in term of months, not including the time spent on requirement specification revision during the design phase. The project involved four parties, the airline, an US tech firm, an Indian tech firm and a Philippine tech firm, see Table 2. In each development phase, two parties were involved in the corresponding task to be accomplished. After the contract had been signed, the airline and the Indian firm worked together for the system requirement. In the first meeting, the requirements were outlined and an important constraint was stated. The check-in system has to be built on top of the latest French system.

3.1.3 Time Taken

Then, the developers in the Indian firm needed to figure out the new functions provided by the latest French system before the second meeting. After a couple of meetings, the requirement specification had eventually settled. One can imagine that the number of meetings for the requirement specification is not small. Every two meeting might be separated by weeks or months. That is the reason why getting the requirements for the airline check-in system took a long time.

3.2 Technological Feasibility Analysis

Technological feasibility is an important analysis to supplement the requirement analysis. The key problem stems on the availability of a technology. Its analysis result could affect the requirement specification.

3.2.1 Technology Available

Technology availability is one concern to be investigated during the analysis. For the services provided by the application service providers (like CommerceOne, Ariba and PaySafe) and the systems to be built by a tech firm (like network security system and the airline check-in system), the technologies being used were either available on the market or easy to be developed in-house. These systems would have no problem in the selection of technologies. Still, a problem comes up.

Problem 1 *After releasing the service for a period of time, say one year, one of the technologies is upgraded to a new version.*

For this version problem, the client firm normally will opt to not to upgrade the system immediate after the new version has been released. Reasonably, the client firm would wait until the system has to be upgraded to the next version. Upgrading technologies would be considered.

3.2.2 Technology Unavailable

It could happen that the service or product to be developed requires a special technology. This technology has not yet been developed. It always happens in Apple products development.

Problem 2 *The required technology has not been developed.*

For the above problem, there are two solutions – (1) in-house development of the technology and (2) refine the requirement specification.

3.2.3 Talent and Budget

Therefore, the purpose of the technological feasibility analysis is to figure out what technologies should be used and where they should be got. Two factors to be considered are the talents in the development team and the budget.

3.3 Economical Feasibility Analysis

Economical feasibility analysis concerns on the factors related to the economic value of the new service, new product or new system. The factors include (1) the development cost and maintenance cost, (2) the operational cost reduction and (3) the revenue generated by the new service or new system.

3.3.1 Cost of Development and Maintenance

Cost of development and maintenance is also called the investment cost, in the context of software engineering. This cost is the end result after bargaining between the budget allocated from the firm and the quotation from the developer vendor. If the firm has talent with knowledge (resp. experience) about the **cost of the technologies** to be used and the **labor cost in system development**,

the bargaining process would be easier and the time spent in bargaining would be shortened.

3.3.2 Operational Cost Reduction

Operational cost is another factor to be considered. If the new system could reduce the **head count**, it is definitely a benefit to the firm. If the new system is able to reduce the completion time of a job, it will be an additional benefit.

For instance, handling a customer service call is always time consuming. If an AI customer service assistant is implemented, many customer service staffs could be laid off. If the AI assistant is talent enough to recognize the mood of a customer, the AI assistant could react appropriately to the customer and then reduce the time in handling the customer service call.

Another instance could be found from the order fulfillment process. Once a customer has placed an order, the order fulfillment center will have to handle the order, like items collection and packaging. The order is thus delivered to the customer by 3PL. The task of items collection and packaging is normally done by human workers. Imagine that a robotic system, like the one in Amazon or Alibaba fulfillment center, is built. Not just the head count could be reduced, the completion time for items collection and packaging could be reduced. Furthermore, human error could be eliminated.

3.3.3 Revenue Generated

For a new service or a new product, the revenue refers to the revenue generated by selling the service or the product. For instance, the revenue generated by selling the new iPhone is to be analyzed. The revenue generated by subscription of a new service from YouTube is another example to be analyzed. For Google Cloud, the revenue generated from the subscription fees of the Cloud services is the third example.

The analysis of this type is relied on the so-called **market survey**. So, it comes to a problem.

Problem 3 *Market survey has shown that the new service or the new product will gain a large market share. Is it trustable?*

For the airline check-in system, there is an intangible revenue. A check-in system is normally integrated with the flight ticket purchasing system. It is so convenience for a customer to search for a flight and purchase a flight ticket (e-flight ticket). Before departure, the customer could on-line check-in through the system. So, the new check-in system could be designed along the user experience design from the flight ticket purchasing system. If this over user experience design is better than the system provided by other airlines, this new system would able to attract more customers to use the system to buy flight tickets. The revenue could thus be raised.

Again, before the system is implemented, no one knows what the additional revenue will be. So, market survey is a common practice to predict the **future revenue** and the **market share**.

3.4 User Interface Design

Apart from the above analysis, the procedure of how a user interact with the system is also included. In other words, the procedure describes how a user uses the system. If the system provides more than one functions, there will be more than one procedure to be described. These procedures are usually illustrated in diagrams, like flowcharts or sequence diagrams. Finally, the graphical user interface (GUI) for each interaction in the procedure is further shown in a figure.

3.5 Documentation with Exceptional Handling

Once the system analysis has been completed, a **system analysis report** delineated all the above analysis results has to be compiled. System requirement specification is the key, in which the functions to be delivered could also be called the use-cases. One function corresponds to one use-case. This specification could be compiled as a document below.

```
================================================
Functional Specification
================================================
<USE CASE 01: Photo search>
User: Photo taken on July.
App: Show the photos.
*** Possible Error ***
1. Photo file corrupted
2. Creation date missing

<USE CASE 02: Location search (English)>
User: Search for TC Hospital
App: Show the map with direction.
*** Possible Error ***
1. Wrong name

<USE CASE 03: Location search (Chinese)>
User: Search for TC Hospital
App: Show the map with direction.
*** Possible Error ***
1. Wrong name
================================================
```

It could also be a text description with diagrams. In UML, the use case diagram is a good tool for drawing such diagrams.

Here, the description for **Function (Use)** in each use case specifies what function is it and how the user can use this function. The description in **Possible Error** is the anticipated error the user might committee during the use of the function. If it happens, what the system has to do will be specified in here. The solution for handling these erroneous situation is called **exceptional**

handling. This report is thus passed to the system design team to design the system for other teams to do coding and testing. As the system has to handle the anticipated exceptional cases, the actual size of the system to be built is larger than the size of the system with the initial list of requirements.

4 System Design

Once the system design team has received the **system analysis report**, the design team will have to design (1) the program structure, (2) the system architecture, (3) the programming languages for coding the programs, (4) the interaction among the users and the system, (5) the finalized user interface design, (6) the environment for running the system and (7) the testing cases for unit test and system test.

Thus, system design is of paramount important as it elucidates every technical aspect of the system. Missing any one design, the system will not be workable. Recall that partial workable is not acceptable in system development.

Principle 2 *System development is an all-or-none game. Only a system with all functions workable is accepted. A system with 99% functions workable is not acceptable.*

Here, I simply outline some ideas behind the system design. If you would like to know more, a practical example could be found in a master thesis from my former master student [4][6].

4.1 Program Structure and System Architecture

To start with, let me have a simple App development. The App is going to delivered simple intelligent services with NLU capability – using voice command to search for a location on a map. Later, the same principle on system design for an information system will be introduced.

4.1.1 Single Program

One approach is to develop everything in a single program which composed of four major modules – user interface module, NLU module, MAP module and system testing module, as shown in Figure 2. Note that the user interface module, the NLU module the MAP module are designed in accordance with the system requirement – the function core and the exceptional handling. The user interface module is the main program which interacts with the NLU module and the MAP module.

System testing module is an additional module to be used for testing if the system is running correctly. As the overall system is designed as a single program, the design of the testing module is relatively simple. The system testing module interacts with all other three modules. System testing module

[6]The thesis is available online at `http://web.nchu.edu.tw/~pfsum/papers/MyNext.pdf`.

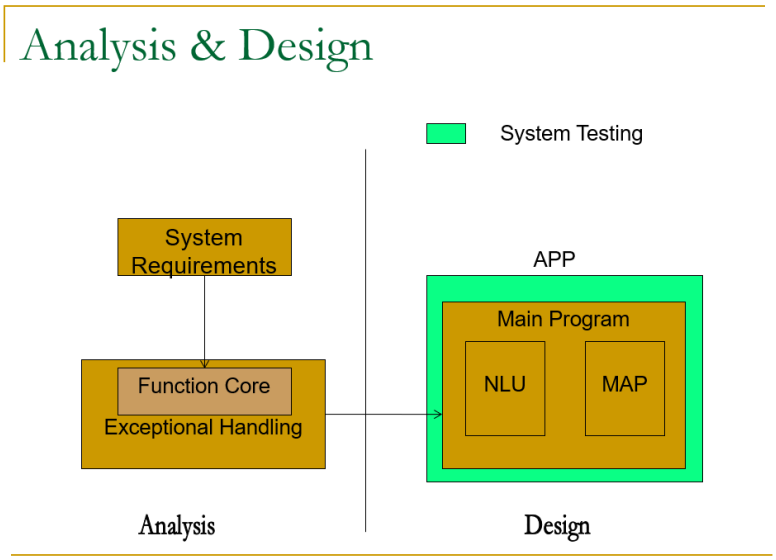


Figure 2: Self-developing everything and putting them in a single program.

is only invoked if necessary. In Figure 2, it is assumed that every module is built in-house, including the NLU module and the MAP module. This design is rather straight forward.

4.1.2 Program with Downloadable Software

A common design approach for the App is to use available software. If the NLU module and the MAP module could be downloaded and used, we could have the program structure like Figure 3. Similarly, the user interface module is the main program which interacts with the NLU module and the MAP module. The system testing module interacts with all other three modules. It is only invoked when it is needed.

4.1.3 Interact with Cloud Services

The three design approach appears when the NLU module and the MAP module cannot be downloaded but available as services on a Cloud. In such case, the program structure will look like Figure 4. This system architecture could be treated as a simple two-tier architecture, in which the App is a tier and the cloud is another tier. The programs to be developed for the App are called **client-side** programs. If the server side is not a cloud but a dedicated server developed by the developer, the programs to be developed and running in the server are called **server-side** programs.

The App provides services to the user. The intelligent services on the Cloud provide services for the App to deliver services to the user. This design approach

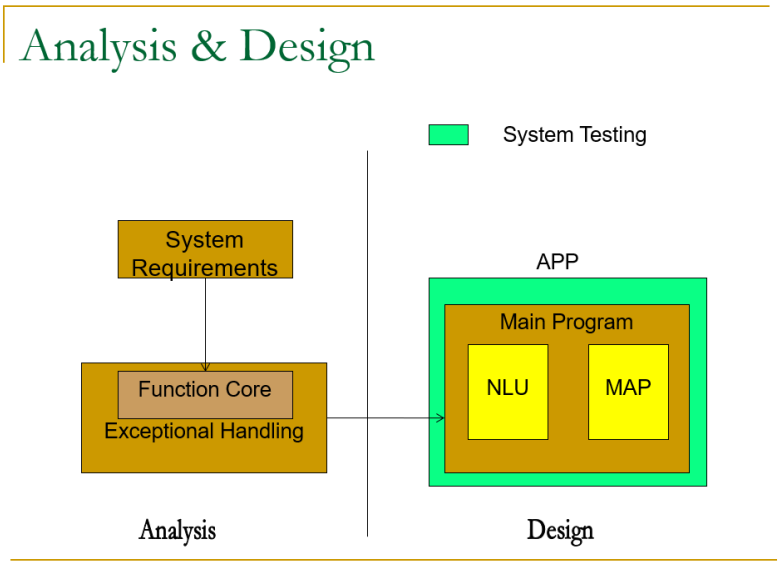


Figure 3: Available softwares are downloaded and used.

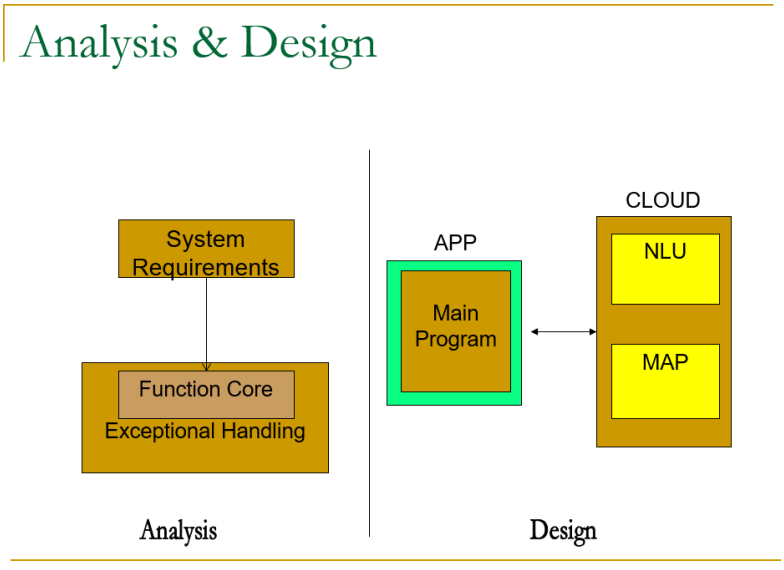


Figure 4: Interacting with a remote Cloud for the intelligence services.

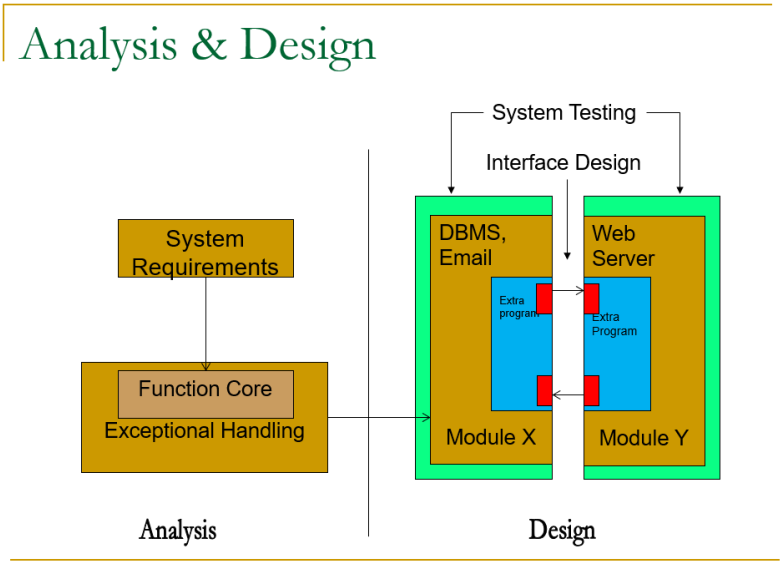


Figure 5: An intelligent information system with two big modules.

is called service-oriented architectural design. The system architecture is a **service-oriented architecture** (SOA). Pretty clear, the Cloud will charge the developer (not the user) for the use of the intelligent services as the registered user for the use of the services is the developer.

4.1.4 Intelligent Information System

For an intelligent information system with two modules (or two servers), as shown in Figure 5, the program design will be a way more complicated and the system architecture is usually a multi-tier architecture. The number of programs to be developed for each module is not small.

4.2 Interaction Design

Interaction design is not just about user interface design, it is also about the interface among different sub-systems and the remote servers like Cloud. It details the step by step how a user can use the system. As the system can provide more than one service, each **use case** will have to compile one interaction design diagram. **Sequence diagram**[7] is a good tool for showing the interaction among different modules and the interface design for a use case.

[7]One point to be noted. In the area of service management or service marketing, **service blueprint** is a diagram for the **service encounter design**. In essence, the **service blueprint** and the **sequence diagram** are the same diagram.

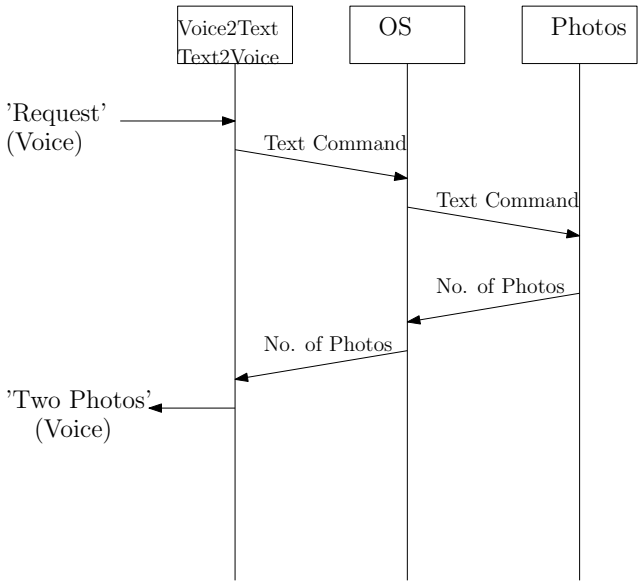


Figure 6: Sequence diagram for photo search use case.

4.2.1 Intelligent Service App

Figure 6 shows the sequence diagram for the *Use Case 01: Photo Search* as mentioned in Section 3.5. Figure 7 shows the sequence diagram for the *Use Case 03: Location Search (Chinese)* as mentioned in Section 3.5. All the use cases are to be delivered by the App in Figure 4. In both sequence diagrams, the OS is put in the diagrams for clarification. In fact, OS is omitted. The sequence diagram shows (1) all the modules involved in delivering the service, (2) the sequence of the interactions and (3) the message to be generated by a module and which module should the message be sent to.

For the photo search use case as shown in Figure 6, two modules are involved. One is the module for voice2text and text2voice. The other is the photo module. Once a voice command has been received, the voice2text converts the voice to a text message. Then, the module invokes the photo module with the text message. The invoked photo module then process the request based upon the information in the text message. A text message is generated upon completion of the process. At the same time, the photo module invokes the text2voice module with the text message. The text2voice module then generates a voice message and the sound is generated for the user.

From this simple application, three (complicated) programs have to be built – voice2text, text2voice and photo-search. The first two programs are included in one module and the last one is included in the photo search module. Here, I call them three programs just for simplicity. One be bare in mind that each program is basically a system. For instance, the voice2text program (equiva-

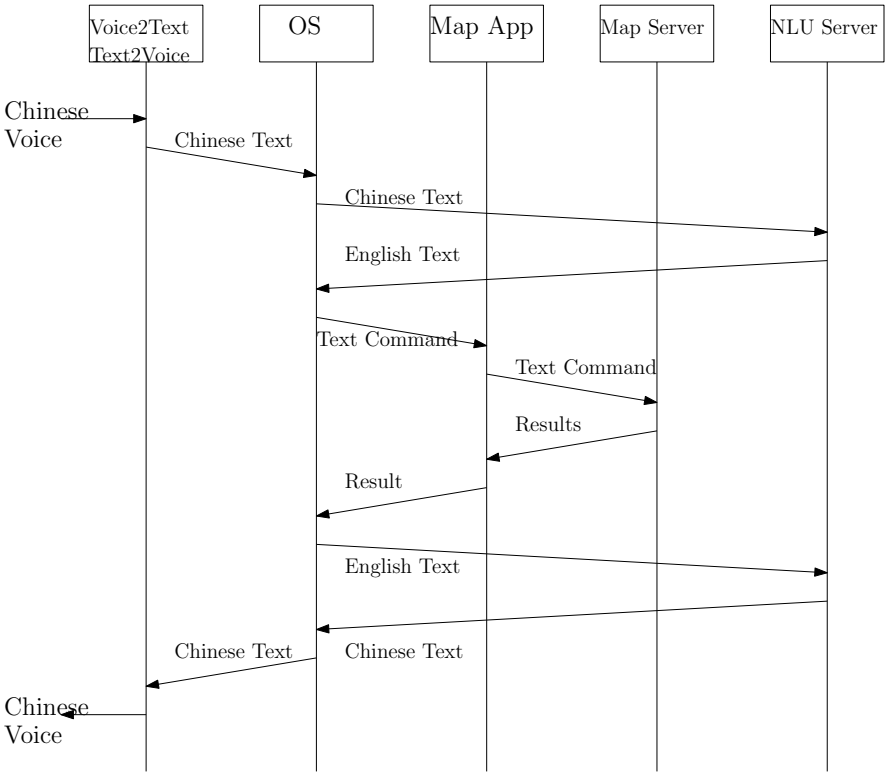


Figure 7: Sequence diagram for location search (Chinese) use case.

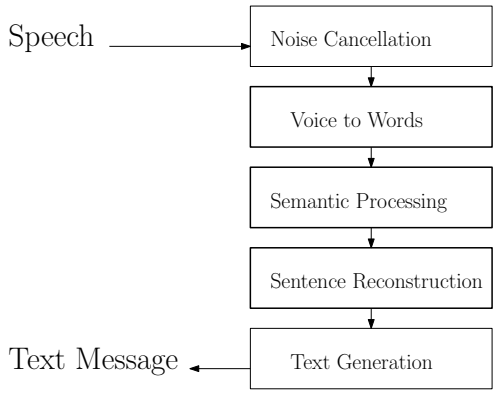


Figure 8: The voice2text system consists of five programs to be developed.

lently system) consists of five sub-programs (equivalently programs) as shown in Figure 8. So, the actual number of programs to be developed is much more than three.

For each program, the event driven the running of the program has to be stated clearly. The pseudo-code of the program has to be described in detail. Its interaction with which programs have to be stated. Finally, the end result is sent to the user. If the return result must be in the form of voice, the user will hear a voice reply. If the return result is a map, the Map App will show the map to the user. These user interface designs have to be defined.

Problem 4 *Should the programs voice2text, text2voice and photo-search be bundled to be single module, two modules or three modules? If they are bundled to two modules, which programs should be bundled together?*

For the location search use case as shown in Figure 7, two modules are involved. One is the module for voice2text and text2voice. The other is the MAP module. Both modules have also been designed to interact with remote servers for the NLU and MAP services.

4.2.2 Intelligent Information System

For the intelligent information system as shown in Figure 5, all the programs are developed in-house. No external service is used. In such case, the scale of a sequence diagram for a use case could be very large. Thus, multiple levels of sequence diagrams would be needed. In principle, the logic of the interface design is the same as for the intelligent service App.

4.2.3 System Design

In Figure 4, the overall system is simply shown as a single system called the *Main Program*. Now, one should understand that this main program has many

programs indeed. Partitioning the system into three modules is just one system design. The programs for voice2text are grouped as one module. The programs for text2voice are grouped as the second module. The programs for photo-search are grouped as the third module. However, this three-module design might not be the best system design.

Principle 3 *Interactions (equivalently communications) among modules should be as minimal as possible.*

While the system is in use, it might have a lot of interactions among the modules. Each interaction is realized by message transfer from one module to another. Operating system will then be involved to co-ordinate this interaction. Delay will likely be happen and memory space is needed.

Suppose that an information system consists of multiple modules. Some modules are running in different machines. This overhead will even be more. Interaction between two modules which are installed in two different servers causes much longer delay, as the interaction has be realized through the network.

4.3 Programming Language

Now, the system design team has to decide which programming languages have to be used to write which programs. The decisions are determined by many factors. Operating system and the supporting systems like DBMS are two common factors which influence the selection of the languages. The programming skill of the developer is another considering factor.

4.3.1 Intelligent Service App

For the photo search and location search applications, the programs are running in a smartphone. So, the programming language for developing such programs has to be compatible with the OS running in the smartphone.

OS	Programming Languages
iOS	C#, C++, HTML5, Objective-C, Python, Swift
Android	C#, C++, Corona, Java, Kotlin, Python

There are two other factors determining the selection of a programming language for App development. First, the interaction between the App and the operating system. Second, the API provided by a Cloud provider.

4.3.2 Intelligent Information System

For the intelligent information system as shown in Figure 5, selection of programming languages for the development of the programs will need to consider a few more factors. For instance, the database management system to be used and the platform (either Windows OS or Linux) for delivering email related services.

4.4 Environment Design

Environment refers to the environmental conditions for running the App or the system. Common conditions for a system running in a computer include (1) the operating system (with version), (2) the database management system (or database server), (3) the web server, (4) the browser, (5) the specification of the CPU and/or the GPU, (6) the size of the RAM and (7) the size of the main memory.

4.5 System Testing Design

Once the programs, the system design and the programming languages have been determined, the system can now be developed – coding.

Problem 5 *What if the system is assigned to a team of developers? Two developers (say A & B) are responsible for the voice2text module. Two developer (say C & D) are responsible for the text2voice. One developer (say E)is responsible for the photo-search.*

4.5.1 Program/Module Level Testing

Each developer is clearly to code the programs assigned to him/her. Before all the programs of the module voice2text have been coded, developer A (resp. B) would need to find a way to test if the programs (not just one program) can work together correctly. Thus, developer A (resp. B) has to create a testing program (yet another additional program) to test the correctness of the programs he/she developed. For the same principle, developers A & B have to work together to create a testing program to test the correctness of the module voice2text.

Developer C (resp. D) has to create testing program for the programs he/she developed. Afterward, developers C & D have to create testing program for the module text2voice. Developer E will have to create the testing programs for the programs in the photo-search module and the testing program for the module.

4.5.2 System Level Testing

Finally, the development team will have to work together to create a big testing program to test all the functions to be delivered by the system. That is to day, the testing program is going to check if the **user cases** specified in the **requirement specification** have been completed. The types of testing programs to be designed and created are depicted in Table 3. In technical terms, unit test refers to the testing of a module. System test refers to the testing of the overall functionalities of a system. The cases to be tested are called the testing cases.

4.5.3 Part of System Design

Normally, the set of testing cases can be designed once the program design, the architecture design, the system design and the programming languages have

Table 3: Types of testing programs.

Level of Testing	Examples
Program	Noise Cancellation Voice to Words Semantic Processing Sentence Reconstruction Test Generation ...
Module (Unit Test)	voice2text text2voice Photo Search
System (System Test)	App Information System

been determined. Therefore, **system testing design** is part of the task to be accomplished in **system design** instead of waiting for the coding team to design.

4.5.4 Automated Testing

One additional advantage of the testing programs is that these programs can be used for self-testing. Suppose the system has been in use. Owing to whatever reason, the system has been accidentally shut down. Once the system is up again, these testing programs can thus been invoked to check the condition of the programs, the modules and the system making sure that the system will be functioning correctly. In case the system shut down is due to file corruption, the testing programs can be used for diagnosing the problematic module or program.

4.6 Document

The final task to be done in **system design** is clearly to compile a document including everything delineated in the above sub-sections. The document will be served as the key document for the coding team to code and test the system.

Principle 4 *Programmers have no need (equivalently no such talent) to understand why the system is designed in that way. They simply do the coding and testing in accordance with the* **design specification**.

4.6.1 For Coding and Maintenance

Another important use of the documents is that they serve as the basis for the future development team to understand the overall design of the system. In

case there is any system upgrade, such as adding new modules for new uses, the future development team could have these documents as reference to figure out how to modify the system.

Fact 1 *The developers involved in the future development team are unlikely the same as the developers in the current development team. The staff in charge of the system maintenance could be changing over time.*

If there is any unclear in the use and maintenance of the system, these documents will be served as the foundation for the staff to search for the technical details.

4.6.2 Communication Gap

One major problem in passing the documents to the coding team is that the programmers in the coding team might not be able to understand the ideas behind the system analysts and the system designers. This is what I refer to the problem of communication gap. It appears almost in every where in the world and in every moment in our living, not limited to system development. Thus, documentation has to be compiled with clear diagrams and descriptions so as to reduce the communication gap.

5 User Manual and Maintenance Manual

Once the development project has been completed, two manuals will have to be compiled – the user manual and the maintenance manual.

5.1 User Manual

Clearly, the user manual is compiled for **all the users of the system**. It describes the detail how the users can use the system. If there are multiple user types, the user manual will have to include for each type of user the information in particular to the user type. That is to say, the user manual will contain five dedicated information for five types of users. Normally, the content from the **user interface design** will be extracted and added in the user manual.

5.2 Maintenance Manual

The maintenance manual is compiled for the **system administrator**. It includes the detail information about the **configuration** of the machines, both hardware requirement and software system requirement for running the system. The requirement on the networking capability might have also included. Moreover, the steps for installation of the system and then testing the system are described. Anticipated problems that the system might appear during running are listed. Solutions for these problems are delineated. In simple words, the maintenance manual can let the system administrator to fix the system if there is any technical problem appeared during the system is in use.

6 Decisions Interdependency

To make a workable intelligent systems, many decisions have to be made, like the system design and the selection of programming languages as mentioned above. These decisions are always interdependent. System architecture could affect the partition of a system into various modules. The programming languages selected for developing those modules would affect the recruitment of programmers to build the modules.

6.1 Programming Language vs Architectural Design

Programmers with different programming language skills could be charged differently. For example, a programmer with knowledge in **network programming** and **system programming** is usually higher paid than a programmer with knowledge in **application programming** and **database system programming** only. It is because the number of programmers with knowledge in **network programming** and **system programming** is fewer than the number of programmers with knowledge in **application programming** and **database system programming**. Moreover, many programmers with knowledge in **network programming** and **system programming** are also with knowledge in **application programming** and **database system programming**.

6.2 Budget vs Programming Language

From the above explanation, one should see that the budget allocated for a system development project would affect the recruitment of programmers. So, the decision on the programming languages is not just determined by the system design but also the budget. In the end, the budget would affect the system design. If the system design is changed, the sequence diagrams for the use cases will have to be modified.

Therefore, multiple reviews and meetings are inevitable to let the analysis team and design team have a **complete** system analysis and system design,.

7 Just Do It Methodology

For a simple service, there is always a simple methodology for the development. Technically, it is called the agile methodology. In layman term, it is called just-do-it. Normally, it is applicable to skillful programmers. Once the system requirement has been identified, the programmers simply do the coding and testing. System analysis and system design are skipped in the first place. Only when the system has been built and tested successfully, the developers come back and put every detail in documents.

Table 4: Exemplar libraries for intelligent services development.

Firm	Library	Languages
Berkeley	Caffe	Python, Matlab, C++
François Chollet	Keras	Python, R
Google	TensorFlow	Python
MathWorks	DL Toolbox	Matlab
Microsoft	CNTK	Python, C++

8 Tools for Intelligent Services Development

Today, a number of tools have been ready for intelligent services development. Specifically, the tools are called libraries. Each library has a collection of programs for call. For instance, the TensorFlow is a library developed by Google for Python developers. Matlab NN Toolbox is developed for Matlab developers. Table 4 lists a few exemplar AI/ML libraries. Some of these libraries even provide pre-trained models for the developers to build their intelligent services.

Clearly, developers could build their intelligent services from scratch. Using the programs and the pre-trained models available in these libraries, the time to build an intelligent service could largely be reduced. The shortcomings are (1) the program might not be easily modified to solve specific problems and (2) the pre-trained model might be problematic.

9 Conclusions

In this chapter, the major concepts in system development are delineated. One should realize that the most intellectual burden in a system development project is in the **system analysis and design**. Once the **requirement specification** and **design specification** have been compiled, the work to be done in **coding** and **testing** becomes relatively easy.

Clearly, problem might still exist during the coding and testing period. The **requirement specification** and **design specification** only include those exceptional conditions that are anticipated. It could have missed some conditions that are discovered by the programmers. In such circumstance, the **requirement specification** and **design specification** will have to make modification.

While the above analysis-design-coding-testing-maintenance model for system development has been elucidated in this chapter, there is no guarantee (no silver bullet) that a system can be successfully development. Without following any model for system development, the situation will be even worst.

References

[1] S. Brin, R. Motwani, L. Page, and T. Winograd, "What can you do with a web in your pocket?" *IEEE Data Eng. Bull.*, vol. 21, no. 2, pp. 37–47, 1998.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation algorithm: bringing order to the web," in *Proc. of the Seventh conference on the World Wide Web, Brisbane, Australia*, 1998.

[3] ——, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[4] D. C.-C. Tsai, "MyNext: A collective intelligence enabled system for cross-checking entrance exam results," Master's thesis, Institute of Technology Management, National Chung Hsing University, Taiwan, 2012.