Development of the IFIS (Interactive Flexible Information System)

Olzhas Abishev and Joon Won Lee

Department of Information and Communication Engineering, Andong National University,
tel: +82-54-820-5644.
oabishev@yahoo.com, leejw@andong.ac.kr

ABSTRACT

The Internet has made a huge impact on the way we, as humans, communicate. During the last decade a series of new communication mediums have emerged and communication protocols have come and gone. This puts new requirements on the development process and architecture of the communication platforms, operated by communities, in order to pro-actively ensure support for future communication protocols. Even further, additional requirements are added when the software itself will be maintained by a community. In this paper we explore the world of Open Source Software. We present our suggestions of usable design-principles and our process in developing a sustainable information system.

1. Introduction

The proliferation of the Internet in the last two decades have given rise to several new communication protocols which supplement older ones, e.g. E-mail and Bulletin Board Systems. With this wide-spread availability and use of the Internet, the concept of "virtual teams" has emerged. Virtual teams can be defined as groups of collaborating individuals to whom geographical location and differences in time-zones are of little consequence, as there are communication mediums where time and synchronicity isn't an issue.

Without the boundaries of space and time there are some obvious benefits to virtual teams compared to "face-to-face" teams, but at the same time this necessitates a good communication infrastructure in order to function effectively.

Agarwal and Maruping use the media synchronicity theory to show how different communication mediums have differing sets of strengths and weaknesses. This theory has come true on the Internet as well, due to the growing plethora of digital communication protocols. An optimized virtual team is thusly a team which has grasped the importance of using various communication protocols and created a framework supporting these.

This puts new requirements on the development process and architecture of modern web platforms. Fielding and Taylor state that "Even if it were possible to build software system that perfectly matches the requirements of its users, those requirements will change over time, just as society changes over time". As the Internet will continue to evolve, new communication protocols are sure to be developed, and modern communication platforms should have the feature to adapt and include new protocols, incorporated in the core design.

For this paper we analyze existing virtual teams existing in the internet, this in turn has resulted in an inconsistency in data storage, poor structure and an overload of information. The current platform used by the virtual teams was deemed too complex for the organization to extend, a new modern web platform needed to be developed. In this paper we describe the research and development that has gone into the creation of the Interactive Flexible Information System, a platform which could potentially evolve and become a sustainable replacement. In addition to this we also explore possible methods for developing sustainable software.

2. Design Principles

Extensible platforms takes on a life of their own post installation, as each individual installation will evolve alongside its own community and their specific requirements. This can easily be verified by observing the difference in features that organizations, which use the same technical solution, offer their members.

In light of this, modularity and flexibility should be key features in the architecture, but important as well is to have a plan for how to handle such evolution, in order to not fragment the community (software users) by breaking compatibility between versions.

In the initial phase of development we identified five interesting principles that could serve us as developers by reducing the development time, and also make the product more maintainable:

- ORM (*Object-Relational Mapping*) - is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. This creates, in effect, a "virtual object database" which can be used from within the programming language.
- DRY (*Don't Repeat Yourself*) - is a principle with clear benefits which ties well into maintainability, as there is less of a risk to introduce inconsistencies in the source code. The objective is to eliminate knowledge duplication, not by reuse, but by having only one source for each distinct piece of knowledge, and let all other components derive their knowledge from that one source.
- CBSE (*Component Based Software Engineering*) - is a way to build systems by using functional or logical components. The components communicate through well defined interfaces, and as long as these interfaces are honored, components may be replaced to provide additional or enhanced functionality without affecting the other components in the system. This method increase reusability as you do not have to rewrite common functions which have already been developed in other systems. This method there for shortens the development time and may reduce the overall cost normally associated with development.
- 4GL (*Fourth Generation Programming Languages*) - while not practices in themselves, can enhance and reinforce other practices. We also deemed it interesting to observe whether the use of a 4GL would provide any significant advantages, as compared to "the usual" web programming languages (e.g. PHP).
- Tracer Bullet development - is a discipline, most commonly used in iterative development, that focus on writing small "proof-of-concept" features which can be refactored into a more functional state if the bullet "hits its mark". This practice can be extremely useful in the initial stages of development of a new component or feature.

3. Design and Implementation

Through the pre-study we had concluded that there was a gap in communication between the members, as the members tended to have a preferred communications protocol, rather than using the protocol best suited for any particular situation. In a perfect world the media synchronicity theory would reign supreme but that is almost never the case, and it would be presumptuous of us to try to change the members' opinions. The next best thing would be an attempt to develop a system which would bridge the gap.

A concern which arose was how to manage the risk of information overload as this system would have the potential to flood all the virtual teams of the foundation with all the information being sent, relevant or not. We then realized that this subject could be abstracted, to not focus solely on an application, but as a more general attempt at a practical solution. To solve the problem, our proposition was to aggregate the information at a central "hub" and categorize it. This would make it easy to search through the information, filter it, and thus distribute it to members based on these categories.

The iterative nature of the action research methodology, among other things, made it seem ideal for our purposes, as we would at the same time develop a new system. The four phases of action research (Plan, Act, Observe and Reflect) could be mapped to the iterative phases of development (Design, Implementation, (User) Testing and Feedback).

We decided that the most appropriate development process was to be an agile one. We felt the need to give the "customer" a central role in the development of the system.

3.1. Customer involvement

During the development, meetings should be held on a weekly basis with representatives from the customer company. During these meetings the overall project progress should be demonstrated, the current state of the system and discussion upcoming features should be held. As this system will later be maintained by the community, it was important that we not only will discuss functionality and usability but also covering the architecture from a maintenance point of view. Their suggestions much consideration, drawing upon their previous experiences should be managed, to produce a better system.

3.2. Implementation

The development of Interactive Flexible Information System is followed, in most aspects, an iterative and incremental development process. The idea is to produce the system in minor iterations which is then released to the client. Usage of this method has decreased the resources spend in post-delivery maintenance significantly and is often used in project, such as this, where a complete list of requirements are not known at the launch of the project.

The use of an incremental process served the project well as we had insight, but there existed no formal requirements specification. We therefore decided to use the concept of tracer bullets in conjunction with close "customer" involvement, in order to create proof-of-concept functionality which could then be demonstrated to the representatives. If they liked the feature, it was further evaluated and implemented.

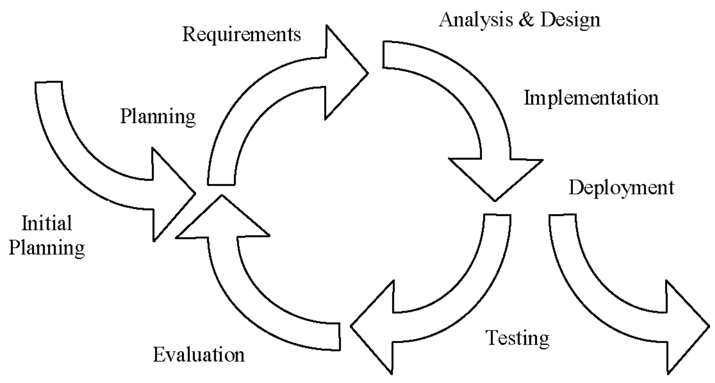

Fig. 1. Illustration of the iterative development process

On the market today, there are a variety of information scraping systems designed to increase the efficiency of communication by merging several different data sources into one. However, one shortcoming of the existing systems is that their primary focus is information gathering, thus offering limited or no functionality to differentiate information with respect to urgency or individual preference. Another common shortcoming prominent in these systems is that they are designed with a specific information distribution technology in mind, and are thus limited with respect to adaptability of, and interoperability with, other technologies. This presents a clear disadvantage from a sustainability and maintainability perspective.

4. Interactive Flexible Information System

To alleviate this problem, we proposing a technical solution which would collect information from sources varying not only in location, but also in format and protocol. The information would then be organized, filtered, and possibly prioritized, before being presented to the user.

There are several ways of organizing information, the most common being to sort it based on existing metadata such as publication date, topic and author. Neither of these sorting options provides the flexibility to allow users an intuitive way of selecting sets of information based on the information itself, i.e., the content.

The most promising solution to this seems to be adding more metadata, in the form of a keyword based summary. This solution too has its own set of problems as was concluded by Paulillo and Penumarthy.

The information passed through various protocols is already "earmarked" with various forms of metadata such as "recipient", "sender" etc. As most of this metadata is in a protocol specific format, the system must be able to understand the various formats, and reformat information from one format into another.

As the goal of this study was to create a first version, we did not only design, but also implemented a system to test the proposition through empirical tests. This system, the Interactive Flexible Information System, is designed to extract data from several computer-mediated communication protocols. In order to achieve this, while still remaining flexible and "future-proof", all data are handled by custom built plugins each handling a specific format and or protocol. The benefit of this design is that all plugins are self-contained making debugging and development simpler.

When information is retrieved, the content is scanned for keywords from which metadata are derived and attached to the content. This technique is commonly known as tagging. Paolillo and Penumarthy describe tags as "a form of metadata, or data which label other data for the purpose of organization and access". This phase is handled automatically, but for convenience, this should also be possible to manage manually in the future, if the accuracy or relevance of the tags with respect to the tagged content is too low.

The decision to use tagging is based on the observed phenomenon and recent popularity in various prominent web-based applications such as StumbleUpon, WordPress and YouTube. It would seem that humans have a natural predisposition towards summarizing context into keywords, although, each individual is likely to have differing definitions for each keyword.

There are a couple of drawbacks with tagging, "neither users nor system can be sure in any specific instance if a particular tag refers to any specific type of information" and "because the tag vocabulary is not 'controlled' or standardized, the categorization produced is informal, and not guaranteed to be the same from one person to the next".

The second drawback is completely avoided in IFIS as only the system operators are able to add tags. This will likely inconvenience users who have their own definitions of what specific keywords mean, but this would be a temporary hurdle until the users have acquainted themselves with the "new" definitions. The benefits of this is however a homogeneous and universal definition, making all similar content related through the same tags.

Finally, the data can be individually distributed by matching the content to another set of tags chosen by the user from a list specified by the system operators. These tags are stored in a user's profile, which also, as a possible future feature, could be determined from a query string at the end of a URL requesting the service.

Having the ability to store settings in user profiles creates a potential for the system to allow distribution of access-restricted information only to authorized members.

Naturally, since the system can gather information from different formats and protocols, IFIS also has the possibility to distribute information in various formats and protocols, e.g, through an RSS-feed, e-mail digest, or via the main web frontend.

In the distribution process, it is of great importance to help the user sort out information which is relevant for him or her. In order to achieve this, information entries which matches the tags specified in the user's profile gets prioritized, and also weighed in terms of number of "hits" in different tags (fig. 2).

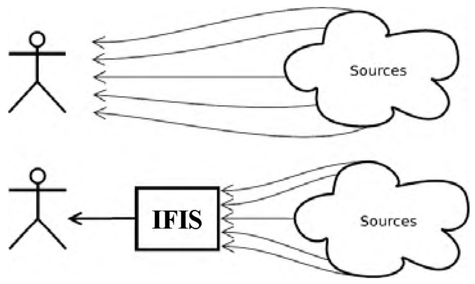

Fig.2. Normal distribution vs. IFIS (Interactive Flexible Information System)

4.1. IFIS in Detail

The IFIS consists of two major components: a *backend* and a *frontend*. The *backend* is meant to be run at all times collecting information from different sources. Consequently, it runs in the background, as a daemon, which means that it is independent of any direct interaction.

The backend works much like an operating system's kernel scheduler, allocating time and processing power for each plugin to execute their operations. The actual plugins does not necessarily need to be limited to a predefined set of tasks. The specification of what a plugin can do, and not do, is purposely "relaxed", in order to allow a flexible behavior of the system, which will, first and foremost, make future requirements easy to adapt into the system, and secondly, it also lowers the complexity of the system.

Since the IFIS frontend will always have a way to display information from various protocols and formats, it is of great convenience to let the frontend hold the one and only true representation of the data models. By doing this, and with the possibility to access models outside of the project workspace, the backend receives the ability to utilize ORM, which means that the DRY principle is followed. As a consequence, this creates an immediate dependency between the backend and the frontend, which from other design principles' point of views, can be seen as a "defect". Although, for this case, with the advantages this choice gives to the system, we feel it is still motivated.

The currently existing plugins for the system, which handles a certain communication protocol, are RSS and IMAP. On the backend side those plugins' responsibility and purpose is to handle information gathering for their corresponding protocol and structure.

The engine which controls the plugins on the backend has fault tolerant characteristics regarding the execution of the plugins. This means that, when a plugin fails to operate, the engine will automatically disable the plugin from executing any further operations until the backend is reloaded. Consequently, an e-mail could be sent to notify the administrator, and the rest of the plugins can safely continue to operate normally without any disturbance.

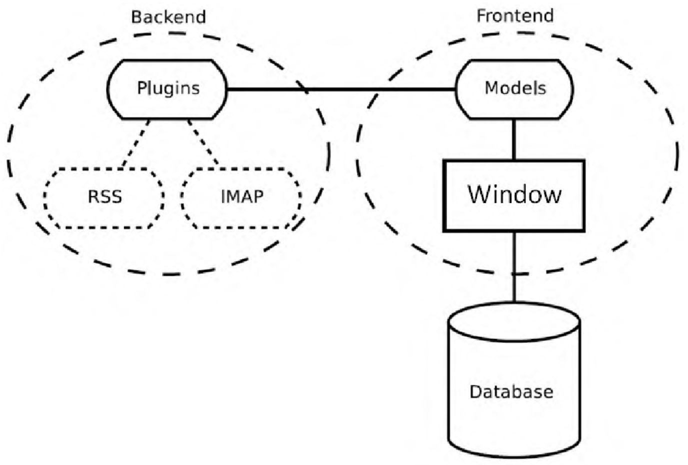

Fig.3. Architectural overview of IFIS

The backend also provides automatic tagging functionality, which the information gathering plugins is intended to use in order to associate tags with the content retrieved. The tagging procedure works by scanning the provided content and matching the keywords that exist in the system, and then map the keywords to the associated tags.

In the frontend, plugins are "applications". Each plugin have their own model of an information entry that corresponds to its format's fields. There is also a way to map the plugin's entry model to a generic entry model. This model shares the most common fields amongst the different formats. By having a generic model, it is possible to mix all information entries without having the need to take knowing of which plugin they came from.

5. Future Research and Applications

It would be most interesting to see how the tagging system could evolve using artificial intelligence to automatically add and adjust the tags for the content. Another study could be conducted about the effects of IFIS with respect to information overload.

We envision that IFIS could operate and expand into areas where this type of application might not dominate the market, such as company intranets. For example, IFIS could be used as a perfect tool for stock exchange and business analysis. With the appropriate sources, such as from business institutes and stock exchanges, a user could be notified about changes in the market which are relevant to him, in other words, changes which match the tags he has subscribed to. Combined with the techniques used in the world cup study he could also be alerted about severe swings where immediate actions are required, thus reducing the risk of losing major investments. This presupposes that someone will write an appropriate plugin to do the job.

Another field could be news-serving or tracking several software development projects.

6. Conclusion

In this study we set out to do a case study on how to develop a system which is maintainable and sustainable in a community. By reflecting on the process, and the system being designed we intended to draw conclusions of the important aspects which exists when developing such a system.

From an architectural point of view we have combined several known concepts and rapidly created a foundation which the community can easily build upon and extend as they see fit. IFIS has proven to be an extremely useful and valuable framework for developing, maintaining web-based software, and incredibly easy to pick up without any prior knowledge of the framework. We feel that this model is an exceptional alternative for creating software destined to be maintained by a community. Paired with the tracer bullet technique, the two create a sum greater than its parts.

We believe that IFIS should be seen as a "role-model" for other frameworks and libraries, and thus be developed in the same "spirit". Much of the reason for IFIS's superiority lies in its use of powerful technical solutions and design principles such as ORM and DRY. We urge all who would consider themselves developers to study these in depth, and to try them out alongside other development techniques such as the tracer bullet.

References

[1] M. Maruping and Ritu Agarwal. Managing team interpersonal processes through technology: A task–technology fit perspective. Journal of Applied Psychology, 2004.

[2] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. 2002.

[3] Adriana Aires Rast´en, Pontus Andersson, David Birath, Shahzeb Iqbal, Shane Kakau, Mattias Lingdell, and Patrik Willard. Suggestions for improving the Free Software Foundation Europe. 2007.

[4] Len Bass, Paul Clemens, and Rick Kazman. Software Architecture in Practice, Second Edition. Addison-Wesley, November 2007.

[5] Eric S. Raymond. The Cathedral and the Bazaar. O'Reilly Media, 1999.

[6] Wikipedia: Object-relational mapping. http://en.wikipedia.org/wiki/Object-relational mapping, May 2009.

[7] Andrew Hunt and David Thomas. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional, 1999.

[8] Ian Sommerville. Software Engineering, Seventh Edition. Addison-Wesley, Pearson, 2004.

[9] E. Sillencea and C. Baberb. Integrated digital communities: combining web-based interaction with text messaging to develop a system for encouraging group communication and competition. 2003.

[10] Infogineering. Understanding information overload. http://www.infogineering.net/articles/understanding-informationoverload.
htm, March 2009.

[11] G. Walsham. Interpretive case studies in is research: nature and method. 1995.

[12] Robert K. Yin. Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series. SAGE, 2002.

[13] Wikipedia: Metadata. http://en.wikipedia.org/wiki/Metadata, May 2009.

[14] John C. Paolillo and Shashikant Penumarthy. The social structure of tagging internet video on del.icio.us. 2008.

[15] Elaine J. Weyuker. Testing component-based software: A cautionary tale. 1998.

Olzhas Amangeldievich Abishev was born in Kyzylorda city, Republic of Kazakhstan, in 1983. He received the B.S. degree in Information Systems department from Kyzylorda State University, Kyzylorda city, Republic of Kazakhstan, in 2004. He received his second B.S degree in juridical faculty from Aulie-Ata University, Taraz city, Republic of Kazakhstan, in 2005. His M.S. degree in Information and Communication Engineering he received from Andong National University, Andong city, Korea, in 2006, respectively. He is currently working toward the Ph.D. degree in Information and Communication Engineering at Andong National University.

Lee, Joon Won was born in Daegu city, South Korea, in 1953. He received the Bachelor degree of Engineering in Seoul National University in 1976. His received M.S. and the PhD from Chungbuk National University, Korea in 1992 and 1997, respectively. He worked at ETRI as the manager from 1980 to 1998. He joined the Andong National University in Korea as professor in the Information and Communication Engineering Department since March 1998. He was a visiting professor at University of Virginia in USA at 2007.