International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-3, Issue-3, March 2015

Intent Driven Design (IDD)

Saurabh Dhupkar

***Abstract*— There is always an intention behind any task or activity we do in our day to day life. This principle is also applicable behind building as well as using a software system or application. The stakeholders have some of their own intentions to provide certain functionalities through software systems and the users have their own intentions behind using the functionalities. The 'Intent Driven Design' suggests focusing on the intention of both, the stakeholders and the users.**

Intent-driven development is an approach that simplifies and standardizes the process of getting detailed technical requirements from non-technical business decision makers so you can develop more complete and consistent requirements as well as design and implement it in such a way that you can keep track of development as well as the usage and also make it more close to end users.

Index Terms—IDD, UI Designers

I. Introduction

Consider a software system which performs many different types of tasks targeting various different kind of users. Mostly it is seen that UI designers face challenges in balancing between number of links at one page Vs minimum number of clicks required to go from one page to another. Many times, either all the links are dumped on Home page or user has to be patient to navigate to the desired page as it will take quite huge number of clicks. Dumping all links on one page makes it conjusted and beautification of page becomes quite a challenge. Otherwise, if user needs to click many times to reach the desired page, after a certain time user starts to think that he/she is going in wrong direction.

Software systems of this size generally also have quite many notifications and putting them all in one page, either makes one notification cycle too long for a user to see all the notifications, or UI designer has to use multiple notification bars. Another issue with multiple notification bars is, one or two notification bars get all the attention and other notification bars get neglected.

It becomes difficult for a new user to find the desired screen and perform the necessary activities. Training becomes mandatory which increases the cost of application and with every new user the need for training stays same, but quality of training decreases as organizations are in hurry to make the new users billable or productive.

If stakeholders are looking for adding multiple functionalities in current running software system, it becomes even more difficult to manage as at the time of original design, it was not forecasted. Therefore, some 'jugad' has to be made. As the number of 'Jugads' increase, maintainability[1] and modifiability of system decreases exponentially. Use of IDD enables architects and developers to create a room for new functionality anytime in the life of system as well as increases modularity[2] of software system.

In 'Intent Driven Design', the architecture and development team considers intentions of both ends, the stakeholders as well as users.

Manuscript received March 20, 2015.
Saurabh Dhupkar, Mumbai, India, +91 9619187377,.

During requirement gathering, architecture and development team should first get to know the intention of stakeholders behind providing certain functionality and later they should group it in suitable user intentions.

When the software system of this size is being developed, following are the main issues faced -

1. Not all the stakeholders have common understanding about functionalities.
2. With different visions and perceptions of different stakeholders, priorities of functionalities differ.
3. Some of the functionalities added in requirements are not clear to even the stakeholders.
4. Functionalities that are introduced in later stages of life of software system create problems as the original design may not be so futuristic that it will be able to adapt the new functionality.
5. It becomes difficult for UI designers to trade-off between no. of links at one page Vs minimum no. of clicks to reach from any page to any other page.
6. User need search for link to the desired screen / activity.
7. User need to look carefully for desired notification at all notification bars.

To handle these, and many more this kind of issues, Intent Driven Design comes in three phases -

1. Requirement Gathering
2. Implementation
3. Post Go-Live

II. Requirement Gathering

As the size of software system grows, the difference between vision between stakeholders also increase. It is difficult for any human being to have complete vision of entire software system of this size. This becomes bigger issue when stakeholders from different departments with different perceptions and different priorities try to push their own ideas up in the priority queue.

The architect and developer teams need to run around all of them to find a common vision.

Therefore, IDD suggests to gather the intentions behind every 'requirement'. This will make the stakeholders think again about the real reason behind having the requirement as well as it will help everybody to place it at appropriate location in the priority queue.

For example, if the main intention behind having the software system is to track employees attendance, then email / sms notifier should take back seat in priority queue. In short, those requirements which are in the list just because some of the stakeholders 'liked the idea', should not be at top priority.

Thus, architects and developers should always first ask, 'why do you want to have this particular requirement' before discussing about 'how this should work'.

Therefore, the stakeholders should describe the intention and answer the following questions -

1 Why is this particular application / requirement is

required ?

2. What exactly will it do for your organization ?
   i. How the particular activity is being handled currently and how it is expected to be handled by the system ? (Record the expected distinct 'before - after scenario')
3. What are the desired roles for this ? (Roles for users for performing tasks, roles for batch jobs for performing tasks etc.)

Too many intentions can make the the system too much complicated. It can make things worse if the intentions are conflicting. However, IDD gives you a chance to handle them much before the application reaches UAT level. This can help architects and developers to identify, understand and handle the conflicting intentions at first stage itself rather than running around and try to find the 'work around' at last stages. Conflicting intentions is one of the major reasons behind failures of the system or systems being 'just satisfactory'. It all comes down to the thoughts like - "Developers took this much time, and yet they build 'this' !!!" and "Developers worked so hard, they even worked overnight and on weekends, but still client is not happy ... ".

Overambitious projects are more likely to end up with these thoughts due to delays or surprises.

Use of IDD can help you reduce or minimize the possibility of system reaching to the above mentioned thoughts.

With the help of IDD, these overambitious projects can be divided into smaller parts based on intentions and the priorities of intentions. Functionalities that come under one intention can be grouped and developed together. Sequence of development of intentions can be decided based on priority and the dependency of the intention.

Thus, the system should be built intention by intention.

IDD is quick and effective methodology for capturing real useful set of functional requirements from non-technical stakeholders and translate them to well-architected working code.

A. PROS

1. Increases clarity behind every functionality to all the stakeholders.
2. Simplifies the prioritization of functionalities.
3. Enables architects and developers to come up with improvement suggestions as everybody is aware of intention behind the functionality.
4. IDD helps all the stakeholders to have common / shared vision.
5. Design becomes more futuristic.

B. CONS

1. It may be difficult trade-off between IDD and Business Secrets. (As it might not be possible always to come up with a dummy intention.)

It is something like "You should not hide anything from Doctor and Lawyer". As nowadays the health of business depends a lot on the software system it is using..

III. IMPLEMENTATION

During implementation IDD shifts its focus from intentions of stakeholders to intentions of users.

Users use the application or WebApp due to some 'Intention'. That intention makes the users to perform certain 'Activity' or a set of 'Activities'. All the other stuff on screen is useless for him/her as the user will be focussed towards the intention behind using the application. Therefore, most relevant things can be shown to user, if the application is aware of the intention behind the visit of user to the application. This can help user to perform desired activities faster on a simplified user interface."

Just to make things more clear, business people are providing certain functionality with some of their own intention(s) and users are using it with some of their own intention(s).

Following are some of the main problems that can be avoided by using IDD –

1. Crowding of links
2. Too many notification bars OR long list of notifications making one cycle very lengthy
3. Difficult to beautify the page
4. User needs to actually 'search' for links to his/her desired activity.

Therefore, in this phase the set of functionalities finalized from the Requirement Gathering phase are divided into groups based on possible user intentions. Architects and developers need to select the possible intentions by which user will use the system and group the activities that can be selected under that particular intention.

The software system should request the user to select the intention from the list of allowed intentions to that user. Based on selected intention, show the links of the activities to user. Also let the user change the 'selected intention' during session. The system should maintain the record of selected intentions and selected activities for future use.

e.g. - While asking user to select intention for current visit, show the list of intention ordered as per his/her choices in past (like most selected to least selected)

A. PROS

1. Development can be done Intention by intention.
2. UI can be made simpler and more uncongested.
3. Having same UIs for Mobile, Computer etc can be made possible as UIs are simpler, which helps to reduce the cost of development.
4. Instead of bombarding user with all types of notifications, only the notifications related to the selected Intention can be shown, which results into shorter notification cycles and lesser number of notification bars. This reduces the possibility of notification bar getting neglected.
5. Extremely easy to adapt new functionalities at any point of time in life of software system.

B. CONS

1. Architecture may become delicate if intentions are not properly listed.
2. Design MUST be futuristic, otherwise application may fail or huge amount of rework will be required.
3. Weak design may lead to poor maintainability.
4. Stakeholders and System Architects MUST be clear about the system and potential users.

IV. Post Go-Live

Every time when a user selects any intention, the system should make an entry of it in database. Same is for activities. This will help stakeholders to understand how the system is being used (by what intention).

Support teams should periodically fetch usage matrices from database. e.g. - Intentions Vs Age of user, Intentions Vs Geographic Location of user, Intention Vs period of year etc. This data will be very helpful for future growth.

A. PROS

1. Stakeholders can get the biggest Intentions behind users using the software application.
2. Intention is the main reason behind use of software application, and stakeholders can get more hold on it.
3. Some intention can have higher priority than other, and application maintenance and SLAs / OLAs can be decided based on priority of Intention.
4. Most used Intentions and most used links can get first position on screen. (Ordering and layout of Intention and links can be decided based on frequency of selection and use.)
5. Stakeholders can come to know the area of main focus (Instead of guessing the most used links, or calculating them based on logs or some third party site meters or site catalyst, stakeholders can get actual counts from users themselves.)
6. Stakeholders can get better view of variations of intentions with respect to age, geographic location, education of users etc.
7. In case of marketing / sales applications, this data can help a lot to the stakeholders as they can map a matrix of Intentions Vs Age of users, Intentions Vs Geographic Location of users etc.
8. Stakeholders can also get the duration wise variations in intentions. i.e. If some particular intention(s) are being selected at some particular time e.g. – Income Tax department will get maximum hits for 'Tax payment' at March end and 'Income Tax Returns' at September
9. Once 'Intention Load Matrix' is obtained, hardware resources can be mutualized to get optimum results and noisy[3] neighbour effect can be avoided or minimized.
10. Technical support team can derive the 'Number of complaints Vs Intention' matrix which can help stakeholders and technical team to focus at exact impact area.
11. If found that system is working fine, then it means that complaints are coming due to inappropriate or incomplete design or definition of that particular Intention.
12. As the Intent-Driven System grows old, it becomes more and more mature and the design itself will reduce the number of complaints over certain time period.
13. Historical data of IDD will guide its future transformation of system.

B. CONS

1. Stakeholders and technical team MUST keep on studying the matrix reports, otherwise the system will NOT achieve the maturity it is expected to reach.

V. Conclusion

"Karma should be known. The cause by which karma comes into play should be known."[4] IDD suggests architects to take into consideration the intentions of both ends, the stakeholders and users for better architecture and simplicity. Use of IDD certainly increases usability[5] of system as well as makes it more futuristic. It also provides important information for software transformation phase[6].

"Companies spending less than 13 percent of their ERP project costs on training are three times more likely to fall short of their business and project goals than organizations spending 17 percent or more"[7] In case of Intent-Driven Design, the cost of user training is much less as it also possesses the simplicity of common WebApp.

'Intent-Driven Design' can also be used to design the desktop based applications. The only constraint in that case should be to collect the usage record from individual user machines to generate the intentions matrices for further evolution. As the contribution of Cloud computing[8] will increase, even desktop applications will also be continuously connected to their respective servers which will make it easier to fetch the usage data and generate the matrices.

References

[1] https://en.wikipedia.org/wiki/Maintainability
[2] https://en.wikipedia.org/wiki/Modularity
[3] http://en.wikipedia.org/wiki/Cloud_computing_issues
[4] Nibbedhika Sutta by Gautam Buddha
[5] https://en.wikipedia.org/wiki/Usability
[6] https://en.wikipedia.org/wiki/Program_transformation
[7] http://www.trainingmag.com/end-user-training-afterthought-or-key-erp-success
[8] http://en.wikipedia.org/wiki/Cloud_computing
[9] Peter Bell, (2008). An Introduction to Intent Driven Design - http://peterbell.sys-con.com/node/311316/mobile
[10] INTENT-DRIVEN DESIGN : DESIGNING AND DEVELOPING SOFTWARE'S BASED ON INTENTIONS OF USERS. International Journal of Advanced Computer Technology (IJACT).(ISSN:2319-7900) - http://ijact.org/volume3issue6/IJ0360067.pdf

Saurabh Dhupkar received the B.E. degree in Computer Engineering from the University of Pune, City Pune, State Maharashtra, in 2008. Currently, He is working as Senior Consultant at Capgemini India Pvt. Ltd. His work area is more inclined in JAVA and Unix for application development and maintenance. He is beginner in the field of research and this is his second research paper.