Individual Research Paper

ENGN2225 Systems Engineering Design

System Attributes for the ANU Degree Planning System

Taeho Jung
u4997019@anu.edu.au
Tutorial Group: Wednesday 3pm

1. Abstract

This paper looks at the system attributes of the software degree planning system by the means of attributes cascade. System attributes are non-functional properties of the system that must be considered in any engineering projects in order to maintain a high quality solution. The paper starts by introducing the theoretical background of system attributes and goes on to apply the concept to the degree planning system for the Australian National University (ANU). The attributes cascade revealed the interrelationships between the design requirements and the sub-systems defined in the system interface. Also, it was possible to spot some gaps in the requirement and the functional block diagram developed in the previous stages of the project. The author suggests reconsiderations of these areas for the final report.

2. Project Background

The ANU has made a transition from a rigid set degree structure to a flexible degree program to allow students to explore diverse fields of academic disciplines. While the students benefit from this program by being able to pursue their different interests, they face a challenge in planning their degrees to satisfy requirements set by the colleges their degrees are regulated. Degrees such as the Bachelor of Engineering which students must fulfil all the requirements set by the Engineers Australia involves many compulsory courses and inadequate planning can easily extend their period of study at the university. This aspired our group to look at the degree planning system for the ANU. This paper looks at the system attributes of the software based solution and unfolds the relationship between the attributes and the subsystems found in the functional block diagram.

3. Theory Review

System attributes are qualities that the final product should have. They differ from the system requirements fundamentally; requirements are function-focused qualities while attributes are non-functional. For example, accessibility and availability are system attributes because they are about

how the system should *be*, not what it should *do* (Masood, 2013). Although system attributes are often not stated exclusively, they give directions to engineering projects and determine the quality of the solution (Microsoft, 2009). They 'may ultimately be the basis of acceptance or rejection by the customer' (Smith and Bahill, 2009). Consequently, it is important to have a clear understanding of what the attributes are for the system and to aim to produce the final solution that reflect such properties.

Despite the significance, system attributes are often not emphasised enough compared to functional qualities (Chung and Prado Leite, 2009). In software engineering, awareness of the importance of system attributes is gradually increasing. In fact, system attributes (or the non-functional requirements in software engineering language) have become a distinct research area of its own (Gross and Yu, 2001). Many academic papers have been written on non-functional requirements, notably by Chung (Associate Professor of Computer Science at The University of Texas at Dallas) as well as non-academic sources such as Microsoft (2009) and Scaled Agile Framework (Leffingwell, 2013) provide basic understanding of non-functional requirements. On contrast, there is few literature addressing this exclusively in the rest of the engineering fields. In non-software engineering projects, attributes are treated more like assumed knowledge and thus they are rather indirectly addressed. They are commonly dealt with during the design requirement stage where the attributes are translated to quantitative terms (Smith and Bahill, 2009). However, it is worth analysing and refining the attributes separately as this process sets a clear guideline and acts as a backbone throughout the project (Gross and Yu, 2001).

3.1 Real world example of system attributes application

An example will be looked at to illustrate the importance of system attributes. The chosen example comes from one of the real-world industrial applications of the system attributes theory presented by Doerr et al. (2005). The system at hand was a wireless control system for controlling and monitoring a manufacturing plant. Firstly, primary attributes were identified and defined through prioritisation questionnaire filled out by the plant staff. The most important primary attributes were reliability, efficiency and maintainability and they were then elaborated further by following the guideline set by the authors. Although exact details were not provided due to the non-disclosure reason, the authors stated that the attributes analysis revealed the architectural weakness in the company's initial planning and a major rework was necessary. This shows that system attributes analysis is actually useful and can reshape the design of the solution to achieve higher satisfaction by the end-user.

4. Application of theory to degree planning system

At the start of the project, customer and design requirements were established based on the student survey conducted by our group and the client meeting with the student services manager (CECS). Since then, a number of systems engineering tools has been applied and many possible solutions the group considered were eliminated, leading to a software based degree planning system as the most appropriate solution. Based on this, a system interface was created using a functional block diagram (FBD) and its sub-systems and their interrelations were identified in process. This paper now aims to combine the requirements and the sub-systems. To do so, system attributes will be derived from the design requirements and interacting sub-systems will be chosen for each attributes. Design requirements and FBD are provided in appendix.

As in the FFBD, the attributes can be divided in hierarchical order. Primary attributes, which are like top level functions in the FFBD, can be taken directly from the design requirements. The primary attributes can be broken down into smaller attributes (called secondary and tertiary attributes) by asking 'how' the higher level attributes can be achieved. Tertiary attributes are then related to a sub-system (or sub-systems) that has the direct relationship or dependency. This top-to-bottom breakdown of attributes is called the attributes cascade. From the cascade, it is possible to see how the design requirements connect to the sub-systems of the solution and observe implications of changes in both the requirements and the sub-systems.

5. Discussion

Table 1 shows the attributes cascade for the software based degree planning system for the ANU. Details of some terminologies that may be unclear in the cascade are explained below.

- Compliance: information provided by the system does not conflict with other information sources (Masood, 2013), e.g. Study@, Programs & Courses, brochures and other publications from the ANU.
- Monitorability: information provided is monitored or checked by staff on regular interval.
- Extensibility: ability for the administrator to add new information and extend the storage easily (Masood, 2013).
- Understandability: able to use the system with little training (Masood, 2013).
- Integrity: the system cannot be changed by unauthorised parties (Masood, 2013).
- Accountability: stores user specific data (Masood, 2013).

- Interoperability: ability of the system to operate successfully by communicating with other external systems (Microsoft, 2009). In this case, external systems include ISIS and WATTLE.

Table 1. Attributes Cascade

Primary attribute	Secondary attribute	Tertiary attribute	Related subsystems
A 1.0 Provides accurate information	A 1.1 Up to date	A 1.1.1 Monitorability	Database
	A 1.2 Compliance	A 1.2.1 Monitorability	Database
A 2.0 Updatable	A 2.1 Modifiability	A 2.1.1 Extensibility	Database
		A 2.1.2 Understandability	Interface, Database
		A 2.1.3 Integrity	Database
A 3.0 Provides degree planning framework	A 3.1 Interactive	A 3.1.1Real time response	Interface
		A 3.1.2 Fast Response	Server Module, Database, Memory
		A 3.1.3 Understandability	Interface
		A 3.1.4 Input assistance	Interface
	A 3.2 User specific	A 3.2.1 Accountability	Server Module, Database
		A 3.2.2 Interoperability	Database
A 4.0 Information clear and accessible	A 4.1 Open to students	A 4.1.1 Web based	Interface, Server Module
		A 4.1.2 Easy location	Interface
	A 4.2 Concise	A 4.1.3 Use of visual aids	Interface
A 5.0 Can evaluate graduation eligibility	A 5.1 Accuracy	A 5.1.1 Monitorability	Database
	A 5.2 Compliance	A 5.2.1 Monitorability	Database
A 6.0 Minimum cost of project	A 6.1 Low Maintenance	A 6.1.1 Stable & Reliable	Server Module
		A 6.1.2 Robust	Server Module
		A 6.1.3 Utilise existing personnel	Excluded
	A 6.2 Low start-up cost	A 6.2.1 Utilise existing infrastructure	Excluded
		A 6.2.2 Utilise existing personnel	Excluded

‘Database’ sub-system appears the most frequent in the attributes cascade and this suggests that it is the most crucial sub-system. On contrast, ‘memory’ sub-system only appears once. This suggests that it may be better to move this sub-system under another sub-system so that each sub-system deals with similar number of attributes (i.e. so that each sub-system is of similar size). Note that some attributes list multiple sub-systems and this shows interrelationships between them. In fact, all sub-systems are related in some ways as they all must work in conjunction with each other to satisfy primary attributes (especially ‘A.3.0 Provides degree planning framework’). This means that a small change in a part of the system can have amplified effect across the entire system. Knowing this flow gives better insight to the system and encourages the whole systems thinking.

A few more gaps in the previous stages of the project were identified through the attributes cascade. It seems that ‘A 5.0 Can evaluate graduation eligibility’ attribute was not the best choice for the design requirements as this is not as general as other attributes. Currently, it is a very functional description of what the final solution should do and it may be better to have a broader description. Since the cascade shows that its sub-attributes are almost identical to the sub-attributes of the first attribute ‘A 1.0 Provides accurate information’, merging the first and fifth design requirements may be another option to solve this issue. Also, the staff interaction with the software faded away as the project proceeded and the focus moved to the student interaction. Staff interaction is important in meeting the requirement of the system as the staff are responsible for providing up-to-date information. Currently, staff-system interaction is not emphasised at all in the FBD. It is recommended that the ‘interface’ sub-system include separate interfaces for student and staff so that the tertiary attribute of ‘understandability (A 2.1.2 and A 3.1.3)’ can be satisfied in both user level.

6. Conclusion

System attributes of the software degree planning system have been analysed in this paper through the use of attributes cascade. Application of system attribute theory proved the usefulness and importance of attribute analysis in creating a high quality solution. Through building cascade, it was able to detect some weaknesses in the current solution which are: (1) limited interaction of ‘memory’ sub-system, (2) vague description of attribute A 5.0 and (3) lack of staff interaction. The identified points will be analysed and adequate changes will be made based on the group’s decision. The updated requirements and the FBD as well as the attributes cascade developed in this paper will set the group a good basis for the verification and evaluation of the solution which is the next stage of the project.

7. Bibliography

CHUNG, L. & PRADO LEITE, J. 2009. On Non-Functional Requirements in Software Engineering. *In:* BORGIDA, A., CHAUDHRI, V., GIORGINI, P. & YU, E. (eds.) *Conceptual Modeling: Foundations and Applications.* Springer Berlin Heidelberg.

DOERR, J., KERKOW, D., KOENIG, T., OLSSON, T. & SUZUKI, T. Non-functional requirements in industry - three case studies adopting an experience-based NFR method. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, 29 Aug.-2 Sept. 2005 2005. 373-382.

GROSS, D. & YU, E. 2001. From non-functional requirements to design through patterns. *Requirements Engineering,* 6**,** 18-36.

LEFFINGWELL. 2013. *Nonfunctional Requirements* [Online]. Available: http://scaledagileframework.com/nonfunctional-requirements/ [Accessed 9 April 2014.

MASOOD, A. 2013. *System Quality Attributes* [Online]. Available: http://www.slideshare.net/adnanmasood/system-quality-attributes [Accessed 10 April 2014].

MICROSOFT. 2009. *Microsoft Application Architecture Guide* [Online]. Microsoft patterns & practices Developer Center. Available: http://msdn.microsoft.com/en-us/library/ee658094.aspx [Accessed 9 April 2014].

SMITH, E. D. & BAHILL, A. T. 2009. Attribute Substitution in Systems Engineering. *Wiley InterScience,* DOI 10.1002/sys.20138.

Appendix 1. Design requirements

Six design requirements were identified in week5: requirement engineering topic. They are:

1. Provides accurate information
2. Updatable
3. Provides degree planning framework
4. Information clear and accessible
5. Can evaluate graduation eligibility
6. Cost of project

Appendix 2. FBD

FBD created in week 7: sub-system integration topic is provided figure 1.

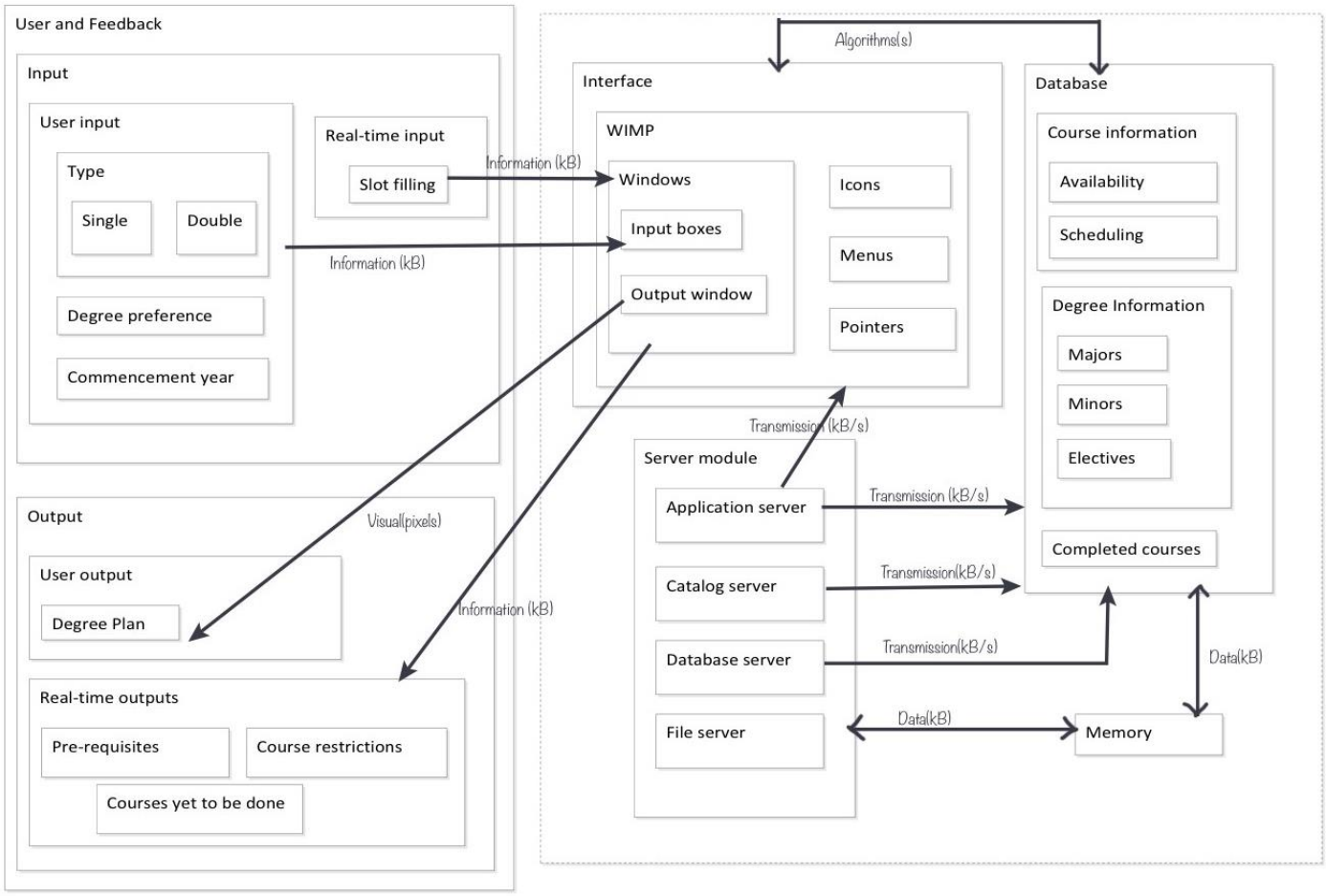

Figure 1. FBD of software degree planning system.

Peer Review Critique

I found peer reviewer 1's comments helpful. Here are some suggestions made by peer reviewer 1:

- Peer reviewer 1 suggested that I find some papers to back up my assumptions which were in the theory section of the draft. I tried to reduce those assumptions and included more back-up citations as per appropriate.
- Peer reviewer 1 also suggested that I include the findings from my research in conclusion and this was taken on board.
- Peer reviewer 1 found the case study section a bit misleading as it sounded as if the rest of the paper would talk about this. Thus, I changed 'case study' to 'real world example of system attributes application' and put it as a subsection of '3. Theory Review' to avoid any confusion.
- Peer reviewer 1 suggested that I mention the IDs in the body of the writing when referring to attributes in the cascade. It was done as suggested.
- Peer reviewer 1 questioned the quality of one of the references which was from a slide share site. I understand the point but I concluded that it was trustworthy as the author is a software architect and a doctoral candidate (provided on the slides) and references were given on the slides which were from trustworthy sources.

Peer reviewer 2 was less helpful because the only point the peer reviewer 2 made was to include more examples. In the aspect 1 comment, peer reviewer 2 suggested that I expand the case study further. This is opposite to the comment I got from peer reviewer 1. I decided to follow peer reviewer 1 because I feel a half page of a case study is more than enough and the case study is not much relevant to the actual project with the degree planning system. I think the peer reviewer also misinterpreted the case study as a part of the actual project like peer reviewer 1 because peer reviewer 2 recommended again in aspect 3 comment that I give more detailed example. Hopefully, this issue is now solved and the paper flows well.