Archinotes: A Tool for Assisting Software Architecture Courses

Juan Sebastián Urrego
Systems and Computer Engineering Department
Universidad de los Andes
Cra 1E No 19A-40, Bogota, Colombia
js.urrego110@uniandes.edu.co

Darío Correal
Systems and Computer Engineering Department
Universidad de los Andes
Cra 1E No 19A-40, Bogota, Colombia
dcorreal@uniandes.edu.co

Abstract

During the last years, software architecture has become an increasingly necessary subject in IT education. At Universidad de los Andes, a software architecture course is taught midway through the Systems and Computer Engineering degree and aims to create skills within the student to identify quality requirements as well as designing software architectures for middle-sized systems. However, based on statistics over the last 8 years of our Software Architecture course, we've noticed that correctly identifying quality requirements seems to be a recurring issue among students. One of the course's objectives comprehends the construction of an architecture document during the semester, which must be done in groups who tend to find it difficult to get together and work as a team. This paper present Archinotes, a collaboration tool to support the teaching of software architectures by facilitating collaboration workload and providing useful information on the difficulties and progress of the students to the course's teacher. The conclusions of the paper prove Archinotes effective in helping students learn critical concepts and artifacts during the software architecture course.

1. Introduction

At Universidad de los Andes, the Systems and Computer Engineering Department introduced the Software Architecture course in its bachelor's degree in 2004. This course is offered each semester and is taken by an average of 100 students each year, who are midway through their degree. Between 2004 and 2012, the course has been redesigned on three occasions. The latest definition seeks to sharpen quality requirements identification skills, as well as tactics and strategies relevant to the design of mid-sized information systems. The course focuses on the creation of high-level design skills and presents the processes and tools necessary to design software architectures for this kind of systems.

During the semester, students validate their architectures through the development of experiments. For each experiment, the students must develop a software which addresses a specific set of quality attributes. The goal of each experiment is to determine if the architectural patterns introduced in the resulting software, satisfy the requested quality requirements. At the end of the semester, the students are required to present a Software Architecture Document (SAD) that reflects the lessons learned during the experiments.

After several semesters imparting the course, we've found two big problems. The first one regarding students having difficulty with adequately expressing the quality requirements to develop.

978-1-4673-5140-9/13/$31.00 © 2013 IEEE CSEE&T 2013, San Francisco, CA, USA

This problem is particularly difficult to identify in each student individually and is usually reflected in the test results. The second problem refers to the difficulty students have coordinating their work during the semester, as they don't have a tool that facilitates collaboration without the need of meeting face-to-face.

This paper presents Archinotes, a tool to help teach software architecture courses. In this context, Archinotes allows teachers and their assistants to monitor each student's progress, as well as problems they may have on their project. This information is immediately available to the teacher and allows him to adjust the content of some of his classes to solve specific and recurring problems. Additionally, Archinotes allows students in a team to coordinate their activities and share information using methods that don't necessarily use face-to-face strategies. The paper presents a brief analysis on the results obtained to this date from pilot tests done on a few students, and compare them with the results from previous versions of the course.

This paper is organized as follows: Section 2 describes the structure and main characteristics of Archinotes. Section 3 shows Archinotes being used in an educational and collaborative environment. Section 4 presents partial results obtained to this date and Section 5 presents related work. Finally, Section 6 states conclusions and future work.

2. Archinotes

Archinotes is a platform that allows users to build architectures collaboratively with the help of mobile devices with a touch screen. Archinotes' goal is to facilitate the software architecture design and documentation process, as well as help students during the cognitive process. This help is done through methods such as stakeholders and quality attributes prioritization, architecture cost-benefit analysis, and architectural viewpoints design, among others. Additionally, Archinotes supports the construction of the users' SAD through external systems such as Facebook, Dropbox, Jira, and Version One.

Archinotes is divided in four components: (1) a mobile application for stakeholders, (2) a storage system, (3) the main processing server, and (4) a mobile application for students. Figure 1 presents Archinotes' components. The stakeholder mobile application allows stakeholders to input their interests, modify their information and observe general project progress. The storage system and the main processing server store and process all of the platform's project data and user information. Finally, the student mobile application allows the definition and prioritization of the quality requirements, stakeholders, business drivers and architectural viewpoints. In this paper, we focus only on the mobile app for students because it is the one that supports the creation and learning of software architectures.

2.1. Students' Component

Archinotes seeks to introduce the student to an intuitive, user-friendly environment. Particularly, regarding software architecture learning, we consider important for students to feel comfortable with the tools they are required to use, instead of confused them with complex interfaces that are difficult to understand. Taking into account the information display models defined in [14], we designed Archinotes to avoid using conventional graphical representations of information (tables, list, etc). To accomplish this, Archinotes main menu presents all available options by dividing them in two parts: the documentation that must be defined before the construction of a software architecture, and the architectural viewpoints. The first part's goal is to automate as much as possible the documentation needed previous to the construction of the architecture. The second

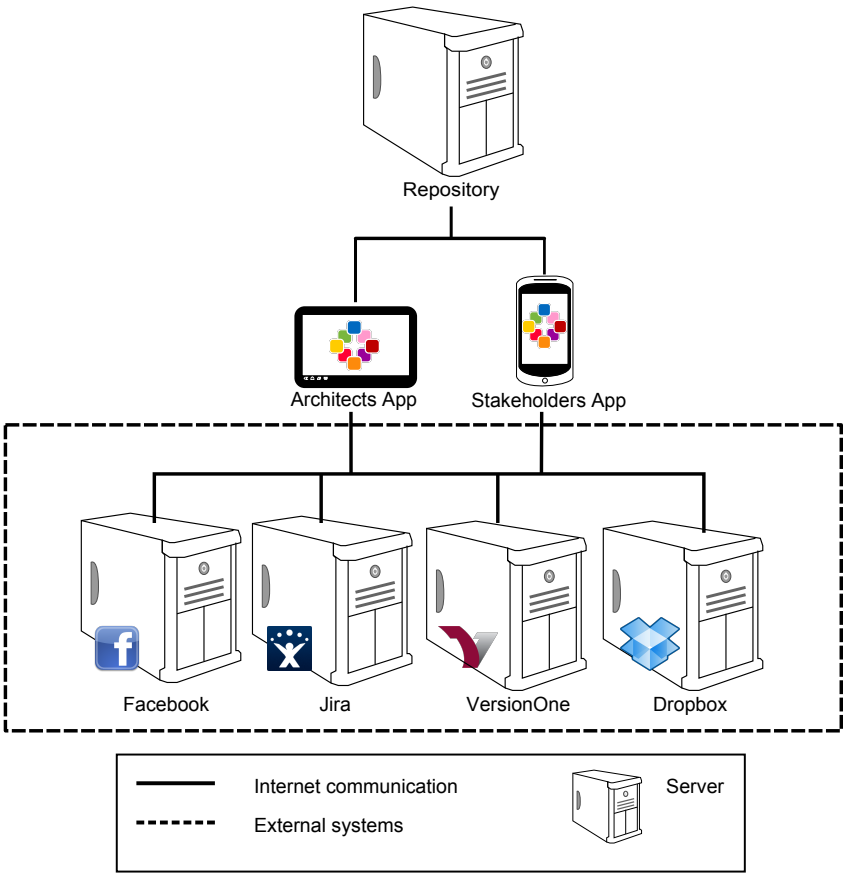


Figure 1. Architecture of the system

part, called Archinotes Modeler, presents the common viewpoints used during the construction of software architecture, as proposed by Rozanski in [11], as well as the models created from each one.

Archinotes functionalities are based on the concepts of color usage, images and figures. As shown in Figure 2, each button has a different color, according to their meaning within the application. Similarly, the usage of different figures throughout the application indicate the events generated when pressed (an action or the display of information), giving each figure their own semantics.

2.1.1. Color and Learning

According to Frank H. Mahnke in [9], colors are capable of triggering different stimuli that help the learning processes in people. Colors provide the ability to get the attention of the observer and increase the concentration of students over an artifact or concept during the learning process. Meanwhile, Leatrice Eiseman [2] states that each color and the different combinations of them provide definable messages for the observer, therefore selecting a certain color or scheme should not be done lightly. Taking this into account, we decided that a strong and striking color palette should be used to capture the observer's attention and help improve their concentration over the artifacts and concepts.

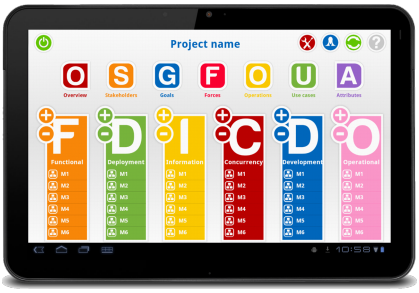


Figure 2. Archinotes main menu

Within the entire application, the color orange is assigned to the concept of stakeholder (except in the models section, where it is assigned to the concept of movement and functionality). This is done so that each time students sees the color orange, they will associate it with the concept of stakeholder. Additionally, where the concept is most important, the background will be of the same color (refer to Figure 3). Stone [13] states that color is of great help in mnemonic teaching, thus the use of saturated colors as background. This characteristic serves not only as a referencing method to define in which zone or stage of the process the student is currently at, but also generates association between concepts, artifacts and colors.

Figure 3. Add new stakeholder

As we previously stated, understanding quality scenarios is one of the students' greatest difficulties. From a predefined pool of quality attributes, students can create multiple quality scenarios for each one. For this, Archinotes presents a friendly graphic user interface where the elements of a quality scenario are presented by cards with meaningful icons and colors. The goal is for the student to associate specific colors and figures to each element, allowing him/her to quickly memorize concepts and avoid confusion between them. Figure 4a presents the Source of the quality scenario as an orange colored card with an icon similar to the one used in the stakeholders menu (recall Figure 3). The selection of this icon is to indicate that the Source of a quality scenario is usually a stakeholder, and since it can also be a system, the user may also add sources that aren't listed as stakeholders. Another clear example of association to colors and figures is the Environment element, which is a green card and its figure is a tree. Similarly, the Stimuli element is presented as a yellow card and its figure is a lightning bolt.

On the other hand, Archinotes doesn't just provide an easier and simpler way of developing quality scenarios. It also provides some predefined values and associations to other elements of the project, thus fastening the definition of software architectures. Recall the quality scenario Source

(a) New quality scenario (b) New source

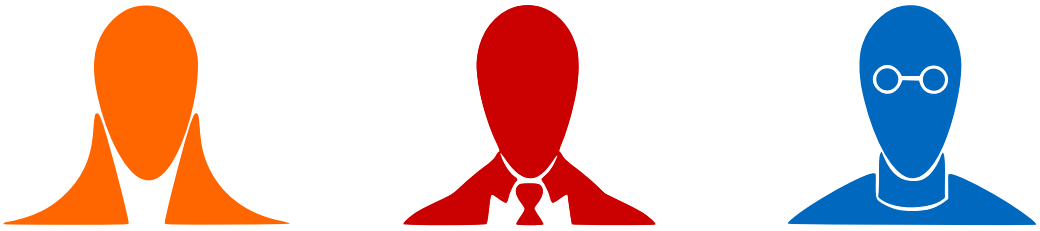
Figure 4. Quality Scenario Configuration

example: when students are working on their SAD, they usually have to write the name of the source or the stakeholder's ID over and over. In Archinotes, the student only needs to write part of the stakeholder's name in the source field and the platform will display the names of all the stakeholders that match the pattern introduced. In this case, the use of a list is believed to be the best way of presenting the results of the search (see Figure 4b).

Similarly, some quality scenario elements may use predefined pieces of data. For example, if the scenario is referring to performance, a possible Measure can be $seconds/transaction$. This property of Archinotes prevents the creation of erroneous information in the scenarios and presents the students with the typical data used in quality attributes. Finally, the predefined pool of quality attributes and the data used within their corresponding scenarios, can be extended as needed by the project administrator, in this case the teacher.

2.1.2. Images and Shapes

The shapes of the buttons and the images used by Archinotes tell the students the events that will be triggered when they drag or press them. Round buttons usually indicate movement or action, thus operations such as adding, eliminating, and modifying, among others, will be presented using this notation. On the other hand, square buttons (or cards) generate new displays when pressed, showing information about the referenced object (quality scenario cards are one example). Using recurring images throughout Archinotes helps the student understand, without any additional hint, the actions that a certain zone or stage of the application executes. Once again the stakeholders represent a good example (see Figure 5).



(a) User (b) Worker (c) Programmer

Figure 5. Stakeholder images

3. Supporting Collaboration Through Social Media

One of the main advantages of Archinotes is the possibility of designing and documenting architectures without the need of face-to-face meetings. As mentioned in Section 2.1, Archinotes allows building models based on the viewpoints proposed by Rozanski [11] through the Archinotes Modeler component. This component is inspired on the Lattanze model template [8], which belongs to the Architecture Centric Design Method (ACDM). Archinotes Modeler provides architects to design and coordinate despite geographical and temporal differences.

In 2012, according to Infographic Labs [4], Facebook had 845 million users of which approximately 50% logged on to their accounts for an average of 20 minutes. At the same time, 425 million of those users use Facebook mobile app. Taking into account that students attending the Software Architecture course usually have different schedules between them (they might not all take the same classes) and may not always be attentive to the project's progress, we decided to integrate Archinotes with Facebook, allowing students to share models in public or private groups, thus taking advantage of the benefits that this social network brings to teamwork. One of the greatest problems that students usually have is the lack of communication and, even though there are several means like email or even Archinotes, users aren't as attentive to it as they are with Facebook or other social networks.

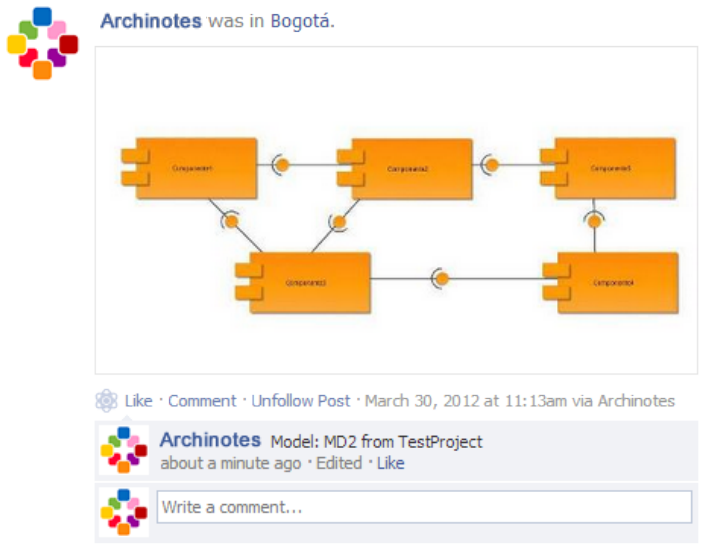


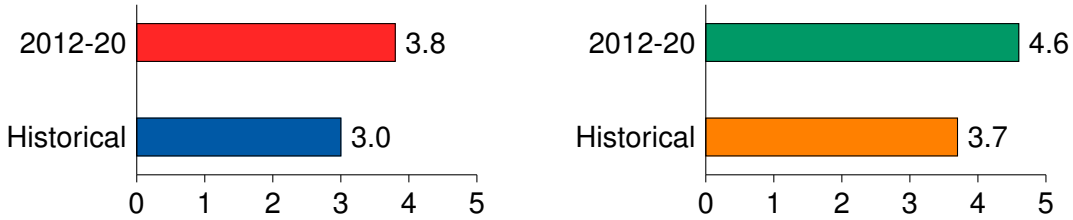
Figure 6. Sharing an architectural model in a private Facebook group

As an example, in 2012 we started a project with the Mechanical Engineering Department from Universidad de los Andes to design and build a Command and Control Center (C3) for the execution, real-time monitoring, and consequent analysis of amateur rockets. The project management software used was JIRA with Greenhopper, where students registered their progress and tasks using SCRUM as their work framework. After three weeks, we noticed that not all of them complied with their responsibilities, and they usually didn't report to JIRA. Our solution to this problem was to create a private Facebook group where everyone was constantly reminded of the milestones they had to register. After two weeks of using this group, 80% of all tasks were registered successfully in JIRA and the project's progress improved noticeably. For this reason, we decided to allow pub-

lishing architectural models and the students' progress in Facebook through Archinotes (see Figure 6). Currently, Archinotes Modeler supports components and classes diagrams from the UML standard. Furthermore, its implementation allows to extend the models that can be designed. In the future, Archinotes will provide the capability to design models based on Architecture Analysis & Design Language (AADL) and Service Oriented Modeling Framework (SOMF), among others.

4. Preliminary Results

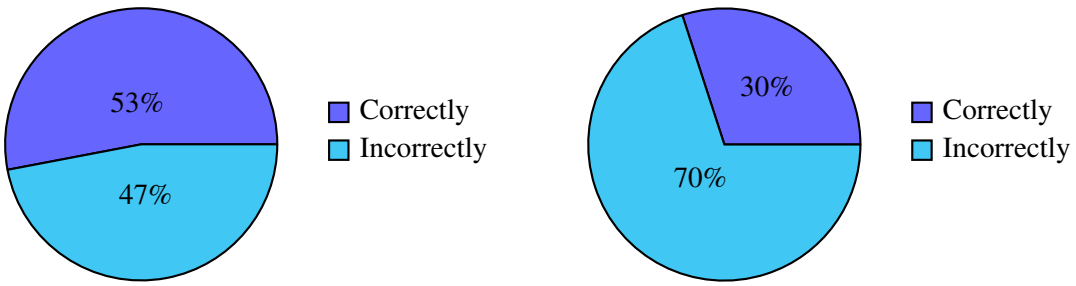
To validate Archinotes, we used the software architecture course of the second semester of 2012 (2012-20). To develop the different experiments, each group had access to Archinotes, and in case they didn't have the necessary electronic devices, the university could provide the equipment through the mobile devices lab. We used the grades obtained in two written midterms and the final SAD presentation as criteria to compare the results of using Archinotes. Based on these evaluations, we noticed that during the final SAD presentation, students showed a better understanding of the concepts and artifacts of software architecture with respect to previous semesters. At the same time, each group individually scored higher compared to the average grade of previous semesters. The results can be viewed in Figure 7.



(a) Exams (b) SAD

Figure 7. Average grade of students

As can be seen, the use of Archinotes improved the exam results in 16% and the SAD presentation in 18%. Meanwhile, specific points of the midterms were analyzed, particularly points where the students were usually confused and got the answer wrong. Figure 8a presents the percentage of students that correctly answered the questions regarding the differences between quality attributes and elements of quality scenarios. Usually by the end of the course, 70% of the students had doubts about concepts and artifacts, however by the end of 2012-20 course the percentage of students with erroneous concepts was far lower (just 47%). The results can be seen in Figure 8.



(a) 2012-20 (b) Historical

Figure 8. Percentage of questions answered correctly per semester

When asked about what factors helped them the most in the learning process and team coordination, most students agreed that sharing information on social networks (in our case, Facebook) considerably improved their cognitive process. Furthermore, a lot of them stated that thanks to the use of colors, especially in quality scenarios, they managed to memorize and improve their understanding of the concepts more efficiently.

5. Related Work

The literature presents different authors that have tried to improve the way we teach software architecture. Among the different methods proposed, we find Problem-Based Learning (PBL), which was introduced by The Community of Learners [3], and focuses especially on teamwork. This method states that students are presented with a problem, which must be solved in groups. Each group designs a software architecture as a solution for the problem, and then the different solutions are discussed in the course. The strength of this method relies in the discussion between the different groups about the viewpoint models created and trade-offs done over the architectures. In our Software Architecture and Design course, we constantly use this method and it works very well for some aspects. However, we've notice that this specific method doesn't help with individual learning. Usually, in many groups, one of the students is the spokesperson while there are others that don't get involved in the software architecture process, thus learning was focused only on certain students.

Additionally, our course usually has a formal presentation or poster at the end of the semester. This is done to allow students to observe all the possible architectures and assess them. This approach is very similar to the one proposed by [6] and applied at the Norwegian University of Science and Technology (NTNU). The problem with this approach is that you don't usually have enough time to observe every architecture in depth, leaving unresolved issues and doubts in some students.

[1] proposes a teaching method based on cases and the student's acquired experience through the observation of alternate architectures and corresponding debates over them. The objective of this method is to raise awareness among students regarding the possibility of producing different architectures for the same problem. All the previous approaches have in common that they don't involve any technological platforms or applications to teach software architecture.

At the same time, [7] designed two master degree courses that focus on two main aspects: the modeling of architectures, and importance of communicating with stakeholders. In contrast to our course, the authors aren't focused on software architecture as a process nor on aspects such as the prioritization and use of information tools.

[12] designed and implemented an application for Windows Mobile which its main purpose is to teach service oriented architectures (SOA). Unlike Archinotes, they focus only in SOAs and don't provide ways of generating documentation of the software architecture. Also, the platform on which the software was built is rather old and is no longer commercially available. In contrast, Archinotes was developed for Android and iOS, which are considered the current leaders in mobile applications.

[10] structures a way of teaching software architecture through 3D models. This approach doesn't consider all the steps in the process of developing a software architecture and focuses only in the models produced. Additionally, they don't use software to build these models.

Finally, [5] presents how color can affect a person in a computer-mediated learning environment. The main test colors are yellow and blue, which were used in the display's background to see how

the users reacted and learned. Similarly, Archinotes is intended to see how the students improve their learning of software architectures through colors and figures, as well as mobile devices.

6. Conclusions

The use of tools that help the mnemonic process improves learning considerably. In the case of software architecture, a better understanding of concepts was achieved through the use of colors and figures, which reflected in a reduction of errors found in the course's evaluations and SAD presentation.

At the same time, Archinotes as a learning tool offers the option of automating certain processes throughout the construction of the SAD. This seeks to eliminate repetitive tasks that are not very pleasant for students and are often prone to errors and confusion. Finally, social networks have proven to be a good tool in the learning process, since it maintains the person in constant interaction with the project and offers the chance to share and debate information.

In the near future, we plan to use Archinotes in a distributed environment so it may provide support for virtual education in the subject of software architecture. We also want the platform to allow data mining from information of the different projects, in order to foresee errors that students could make and propose possible solutions to them.

References

[1] R. de Boer, R. Farenhorst, and H. van Vliet. A community of learners approach to software architecture education. In *Software Engineering Education and Training, 2009. CSEET '09. 22nd Conference on*, pages 190 –197, feb. 2009.

[2] L. Eiseman. *Color - Messages & Meanings: A PANTONE Color Resource*. A Pantone color resource. F+W Media, 2006.

[3] M. Howard S. Barrows and B. Robyn M. Tamblyn. *Problem-Based Learning: An Approach to Medical Education*. Springer Series on Medical Education. Springer Publishing Company, 1980.

[4] Infographic. Facebook 2012, June 2012.

[5] R. Kumi, C. M. Conway, M. Limayem, and S. Goyal. Research article learning in color: How color and affect influence learning outcomes. *Professional Communication, IEEE Transactions on*, PP(99):1, 2012.

[6] P. Lago and H. van Vliet. Teaching a course on software architecture. *Software Engineering Education and Training, Conference on*, 0:35–42, 2005.

[7] P. Lago and H. V. Vliet. Teaching a course in software architecture. In *18th Conference on Software Engineering Education and Training*, pages 35–42. IEEE Computer Society Press, 2005.

[8] A. Lattanze. *Architecting Software Intensive Systems: A Practitioners Guide*. Taylor & Francis, 2008.

[9] F. Mahnke. *Color, Environment, and Human Response: An Interdisciplinary Understanding of Color and Its Use as a Beneficial Element in the Design of the Architectural Environment*. Interior Design Series. Wiley, 1996.

[10] C. Rodrigues. Visar3d: an approach to software architecture teaching based on virtual and augmented reality. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, volume 2, pages 351 –352, may 2010.

[11] N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011.

[12] M. Sherriff. Teaching web services and service-oriented architecture using mobile platforms. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages S2D–1 –S2D–6, oct. 2010.

[13] T. Stone, S. Adams, A. Morioka, and N. Morioka. *Color Design Workbook: A Real-World Guide to Using Color in Graphic Design*. Rockport Publishers, 2008.

[14] N. Yau. *Visualize This: The FlowingData Guide to Design, Visualization, and Statistics*. Wiley, 2011.