Lessons for Software Modeling from “Architecture 101” Movie

Kai-On Chow\textsuperscript{1} and Tak-Lam Wong\textsuperscript{2}

\textsuperscript{1} City University of Hong Kong
\textsuperscript{2} Hong Kong Institute of Education

\textsuperscript{1}cspchow@cityu.edu.hk, \textsuperscript{2}tlwong@ied.edu.hk

Abstract

In software modeling, it is difficult to properly arrange the modeling of system structure and behavior as the traversal between software models usually lacks a clear progression path. Taking an interdisciplinary approach, this paper tackles the problem by borrowing ideas from a successful movie “Architecture 101”. The commonalities between the movie and modeling are studied. The result is a proposal for multi-modeling construction. The benefits include more explicit guidance in software development. And the progression from model to code is made more productive.

\textbf{Keywords:} System view, abstraction level, fragment, balance, “Architecture 101”

1. Introduction

“Architecture 101” is a recent South Korean film [1, 2] with successful box office ratings [3]. Nowadays software modeling is an important technique in software development. The twos are far apart in terms of the disciplines each belongs. However, there are important commonalities that the two share. And it is hoped that the engineering of software artifacts can borrow useful ideas from the accomplished movie product, in the end improving the software engineering discipline.

This paper is an integration of art and science. It investigates ideas that can be applied into computer science from the filming industry. The similarities and differences between “Architecture 101” and software modeling are identified. The film is analyzed to give clues on how to structure the concurrent development of multiple software models.

Boyd has stated, the theory of modeling remains incomplete though software modeling is the heart of engineering [4]. Software engineering is awaiting progress. It is not clear how different models can be coordinated during the model construction process. This paper tackles the problem of a suitable use of the multiple system models in the software development process.

This paper is organized as follows. Section 2 discusses the key idea of multiplicity in software modeling. Section 3 describes the “Architecture 101” movie and summarizes its pleasing harmony characteristic. Sections 4 and 5 highlight the pleasing aesthetic from historical portraits, gives a comparison to software development and draw important lessons applicable to software modeling.

2. Multiplicity in Software Modeling

This section describes the key concepts in system views and abstraction levels, and highlights the consequences from multiplicity.
2.1. Multiple System Views

Modeling makes it easier to understand essentials of a system and is a central tool in model-driven software development. Abstraction is closely related to modeling as models are built by abstracting important system elements. Modeling has migrated from a single perspective to multiple ones. Multiple perspectives put software in better context. As such, multi-modeling prevails in current software development, as is shown in UML 1.x [5] and the later UML 2.x [6]. There are different proposals in terms of 2-model, 3-model and others. The more common one is a dual model, comprising structural model and behavioral model. The two models coexist but are opposite and relative to each other. During the development process, these models grow over time and become models with increasing details. However, multiple models shoulder the shortcoming of inconsistencies among system models. Further it is not easy to provide a clear guidance on when and how to construct these models. Although models are part of system, it is not sure how they should be integrated. There is also a lack of clear mapping between model elements.

2.2. Multiple Abstraction Levels

Software methods incorporate the techniques of decomposition and composition in organizing system parts into a vertical hierarchy of abstraction levels. This is common to both the structured and the object-oriented approaches. Abstraction leveling has become an important idea in tackling complexity in software development. Abstraction level is a relative term, with no absolute value. It is not clear what constitutes high and low levels in the absolute scale. These abstracted levels comprise high and low levels, forming a relative hierarchy. A major weakness in the abstraction process is that it is not clear when should abstraction be stopped. This is so because the nature of abstraction implies that the process should be complete when essential features are obtained. However, the task of separating essentials from non-essentials is itself a lengthy iterative process with continuous refinements. Other shortcomings include no relationships exists between the hierarchical levels. Inconsistencies exist between abstraction levels. In addition, inconsistencies may also exist between models and abstraction levels.

3. “Architecture 101” Movie with Pleasing Harmony

This section presents an analysis of the “Architecture 101” movie with a view to give hints to tackling the software modeling problem. The analysis is a personal interpretation of its author and is subjective by nature. The framework of analysis is a based on using cubism in painting art, which will be explained later in the section. This cubic analysis of the movie is done within the context of the avant-garde movement. In previous searches of literature about the movie, no mention of cubism or avant-garde has been found. Most of the listings shown in this section are not intended to be a comprehensive listing of all relevant entities in the movie, but more in giving illustrative examples.

3.1. Movie Description

“Architecture 101” is a romance film about architect and love [1]. It is released in March of 2012 in South Korea and has since received high success in box office rating with millions of admission [3] as well as favorable reviews [2]. The movie tells the
story of two university students, namely Lee Seung-min and Yang Seo-yeon, that first meet and fall in love in an Introduction to Architecture course (course code: Architecture 101) in Seoul and re-meet some 17 years later in the office of an architecture firm to design and rebuild a house in the southern Jeju Island. There are two pairs of actor/actress to star the two main characters for the two time periods, college study years in the past and the present house project. The film is set in a way so that the romance story progresses in two parallel paths of past and present, skillfully incorporating a traversal across casts, eras, seasons and places, and subtly transmitting messages at different levels of depth.

3.2. Dual paths through Fragmentation

The movie is explicitly structured with alternate displays of film fragments for the first love in the past and the hopeless fate for a second chance love in the present time period. In other words, it does not follows a sequential progression of story from past to present or vice versa. In order to achieve this, the romance story is decomposed into numerous fragments, constituting an employment of fragmentation in the movie. As such, part of movie quality will be related to how the love story is fragmented and how the fragments are organized so that film audience can read the romance story, extract messages and enjoy viewing the movie overall.

The film starts at the architecture office and ends in the rebuilt house, with both belonging to the present time path. The main story for the past includes first meet in a university setting, fall in love and eventually separate at the end of semester. The main story for the present portion includes re-meeting again to seek help in building house. The twos spend a lot of time together in Jeju during the house renovation. But eventually Yang stay to care for her dying father and Lee marries his fiancée and flies to US. Some examples of film fragments include meeting in the architecture office but Lee pretends not remembering Yang, eating in a restaurant with Lee’s fiancée participating and informing Yang of their coming marriage, first encounter with Yang in the “Architecture 101” course, first kiss in the railway station, coaching by good friend on how to get girls by Lee, sadly leaving the CD and player by Yang in the vacant house when failing to meet Lee in early snow, expressing opinion on life through spicy soup in a waterfront restaurant in Jeju, teaching student to play piano in the renovated house, surprise receipt by post of the CD and player returned by Lee who is already on a flight to US, and others. In sum, there are numerous fragments that form the movie story. They are organized in a skillful way so that eventually the romance story is properly told in the mind of the film writer and director. The audience views the movie story and read the messages by piecing together the different fragments. The dual paths complicate the presentation of the movie story and probably instill to the audience a sense of ambiguity.

The followings are observed with relation to the frequency of alternate display of fragments in both the past and present, and with the pattern of fragment length:

- At the beginning of the movie, there are more fragments shown on the past. The length of these fragments tends to be longer than those at the end of the movie.
- Towards the end of the movie, the switching between past and present are more frequent, and these fragments are shorter compared to those at the beginning of the movie.
- The developments of story are not synchronized in both past and present.
### 3.3. Similarities and Differences at Three Levels

A number of contrasts are observed. They can be grouped into three levels, representing increasing significance in meanings. Some examples in the movie are:

- Although Yang already has a withering interest in piano, she still plans a piano room in the new house. These are done to please her father in his remaining years. Yang places high priority on his father’s wellbeing, though money is important to her.
- In contrast, Lee keeps his promise from first love to help Yang to build a house, though he is stupid and idiot (in Yang’s words).
- Yang’s father does not realize that her daughter has less interest in piano. Similarly, Lee’s mother has no knowledge of how his son is affected by the low family status among university peers.
- For some 17 years after the first love, Yang still retains the thrown away house model. Similarly, Lee also keeps the CD and player collected from the vacant house.

At a higher level of personal value, the following reflections are deduced:

- The retention of the thrown away house model and left-behind CD and players indicates that both Lee and Yang are serious in this first love and are willing to keep these deep and lasting memories.
- In the eyes of Lee, Yang is a bad girl, implying that Lee has a different moral value on the matter. In contrast, Yang does not think that she is bad and Lee is regarded as “stupid, idiot, ruin everything in the love romance” and be responsible for the failure of the first love.
- Yang is willing to stay behind and care for her dying father in his remaining years in the beautifully renovated house. In contrast, Lee flies to USA with his newly married wife, leaving his mother in the old Seoul house. This reflects different orientations for the two persons in terms of personal value.

At an even higher level, the following observations on the wisdoms of life are obtained:

- There is no such thing as absolutely right or wrong. Even when Yang is bad in sex treatment, she also does good things in caring her father. Lee insists on sex within moral standard, he flies away, leaving his mother behind.
- To Yang life is harsh, just as Yang has expressed through spicy soup in the seaside restaurant in Jeju Island. One does not know the ingredients in the spicy soup, only that it is spicy.
- As a fact of life, both Lee and Yang should not (and cannot) marry together. Though they like each other and are serious in the first love and the second encounter, there are fundamental differences between them in terms of views and value.
4. Avant-garde/cubism with Pleasing Aesthetic

Originally used in the French army, avant-garde refers to works that are experimental or innovative, particularly in art and culture. Essentially, the term represents a pushing of boundaries beyond the norm. An example is pushing the movie theory boundary towards using the technique of fragmentation, beyond convention. Cubism is an avant-garde movement. Object in cubic works, such as a movie story, is broken up into fragments, analyzed and then re-assembled. Fragments are important parts of the movie, as shown by the frequent flashbacks as the movie story progresses. It is interpreted that the “Architecture 101” has adopted cubism into the movie production, though no substantiation or confirmation has been obtained. The cubic fragments are broadly classified into two groups, namely the first love in the past and the second chance love in the present. These two groups intermix in the movie as the story is unfolded. The overall organization of fragments conforms to a conscious, deliberate dissociation and recombination of fragments into a new artistic entity made sufficient by its movie structure. The “Architecture 101” has made an attempt to stamp out ambiguity and to enforce audience to read the movie through fragments. In summary, the following points are noted from the cubism perspective:

- The film does not choose to take a low-level straight-forward description of the love story.
- The alternate display of past and present fragments is a new concept in the spatial organization of the love story, creating cuttings between past and present.
- The inter-mixing and co-existence of fragments for the past and present symbolizes continuity in time, expressing complexity within the movie.
- Deeper reading of the film message by the viewing audience depends on how the fragments are pieced together, but not on visual experience and emotional feeling.
- The film does not present an explicit judgment on whom or what is right or wrong.

5. Lessons for Software Modeling

This section presents the main results in terms of the lessons applicable to software modeling. The three commonalities of view, level and fragments are identified.

5.1. Commonalities

With reference to the software modeling in Section 2, the following cornerstone concepts are re-iterated:

- Dual models co-exist in software development. Model development dictates the need to move back and forth between structure and behavior.
- Abstraction leveling is an important consideration in software development.
- Progression in software modeling needs to incorporate both switching models and deepening levels.
Summarizing the “Architecture 101” movie in Section 3, it can be observed that:

- There are multiple views within the movie story as represented by the college study years in the past and the present house project.
- The past and present sub-parts intersect randomly. This seems to create ambiguity in the story.
- The two pairs of actors and actresses represent a multitude of viewpoints that broadens a wider context and continuity in life.
- Messages are conveyed at multiple levels, namely the movie story, personal value and wisdom in life.
- Multiple fragments from the movie help to establish the story space and the movement in space for story progression.

Table 1 gives a comparison of the development process for the movie and modeling. It is obvious that the twos share similar development steps.

<table>
<thead>
<tr>
<th>Movie</th>
<th>Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progress to present movie story</td>
<td>Progress to build models</td>
</tr>
<tr>
<td>Start at present sub-part</td>
<td>Start at structural model</td>
</tr>
<tr>
<td>Flashback to past sub-part</td>
<td>Switch to behavioral model</td>
</tr>
<tr>
<td>Add details in movie story</td>
<td>Add details to models</td>
</tr>
<tr>
<td>Iterate back to ‘Flashback’</td>
<td>Iterate back to ‘Switch’</td>
</tr>
<tr>
<td>End at present sub-part</td>
<td>End at structural model</td>
</tr>
</tbody>
</table>

Table 2 shows the three commonalities between the movie and modeling.

<table>
<thead>
<tr>
<th>Movie</th>
<th>Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple sub-parts of story (past and present)</td>
<td>Multiple views of system (structure and behavior)</td>
</tr>
<tr>
<td>Multiple levels of movie messages</td>
<td>Multiple levels of abstracted details</td>
</tr>
<tr>
<td>Movie fragments</td>
<td>Model elements</td>
</tr>
</tbody>
</table>

5.2. Model Views and Levels

Meaningful modeling results will not be produced with the construction of a structural model alone and with no reference to the behavior side. In order to suitably develop the multiple models of structure and behavior, a number of switching between models during the modeling process are needed. This is similar to the alternate display of fragments in the movie. Compared to the movie, the following similarities are observed in modeling:

- For movie: “At the beginning of the movie, there are more fragments shown on the past. The length of these fragments also tends to be longer than those at the end of the movie.”
For modeling, similarly, it normally starts with the class diagram to build system structure. It also takes longer time in drawing the class diagram.

- For movie: “Towards the end of the movie, the switching between past and present are more frequent, and these fragments are shorter compared to those at the beginning of the movie.”

For modeling, similarly, the frequency of switching between structural and behavioral models tends to be higher towards the end of modeling.

- For movie: “The developments of story in both the past and present are not synchronized.”

For modeling, similarly, it usually starts and ends at structural model.

5.3. Model Fragments

Unlike view and level, there exists a bigger difference between fragments in the movie and that in modeling. In the movie, it is the fragments for the past and present that form the building blocks. As we have observed earlier, these fragments vary in display frequency and duration at various stages of the movie. Some movie fragment examples are:

<table>
<thead>
<tr>
<th>Table 3. Sample Movie Fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Casts</strong></td>
</tr>
<tr>
<td><strong>Places</strong></td>
</tr>
<tr>
<td><strong>Eras</strong></td>
</tr>
<tr>
<td><strong>Seasons</strong></td>
</tr>
<tr>
<td><strong>Others</strong></td>
</tr>
</tbody>
</table>

In modeling with UML, the three kinds of building blocks are things, relationships and diagrams [6]. Examples of structural and behavioral things are class, use case, interaction and state machine. Association and generalization are examples of relationship. Using the common diagram groups, a complete listing of model elements is shown below:

<table>
<thead>
<tr>
<th>Table 4. Sample Model Fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class diagram</strong></td>
</tr>
<tr>
<td><strong>Use case diagram</strong></td>
</tr>
<tr>
<td><strong>Sequence diagram</strong></td>
</tr>
<tr>
<td><strong>Activity diagram</strong></td>
</tr>
<tr>
<td><strong>State diagram</strong></td>
</tr>
</tbody>
</table>

A comparison of Tables 3 and 4 shows apparently few similarities between the movie and modeling. The fragments in the movie are clips which incorporate combination of casts, places, etc. The fragments in model building are units of information which may involve scale and granularity.
5.3. Balance between Models

One of the keys for the success of the movie is the balance between the past and present fragments with different levels of messages, which perfectly interweave to form the whole story. The past fragments are displayed at the right time to deliver a message to enhance the overall effect of the movie. Towards the end of the movie, the past and present fragments are switched more frequently, and the messages delivered are tightly coupled. In software modeling, diagrams cannot be worked out in a certain ordering. Very often, multiple diagrams are required to be drawn together with reference to each other. As the modeling process proceeds, the diagrams become more and more detailed and the abstraction level increases. Frequent reference between diagrams (or models) is needed in order to obtain a highly accurate and complete model of the system. As a result, strategic planning and structural organization are needed to strike a balance between structural model and behavior model in software modeling.

6. Discussion

This section gives a discussion in modeling criteria, fragmentation, and development order.

6.1. Multi-modeling Criteria

A set of criteria that can be used to refine modeling is derived from Tables 1 and 2. The similarity in development process and the common entities lend ideas to switching views and deepening levels. In terms of order importance in modeling, it should be frequency of switching views, sequencing of fragments, duration of each fragment, and beginning and ending of modeling.

Just as the movie does not take a straight-forward progression, multi-modeling needs to determine how often to switch from one view to another, what fragments to create, how long to use a fragment in one of the dual model, and finally what model to begin and end the process.

6.2. Fragmentation and Sequencing

Fragmentation is an important key to the success of the movie. But is it as important to modeling is still an open issue. How should fragments be defined in models? Should it be at basic level of primitive or at certain granularity? The answers to these questions may provide result to the balancing question in suitable modeling and leveling. Up to now, the common understanding is that thing and relationship at the basic level are the model building block, whereas diagram is a higher level building block. In sum, there is a need to re-think the nature of important model elements and the ideal organizations of model elements in facilitating progression from multiple models in design to the later stages of programming. We have seen this phenomenon before when the function-decomposition based dataflow diagram is “side-lined” in many object-oriented methods as the next wave of the object technology emerges. But the de-emphasized function property still exists in system, and system complexity still demands decomposition. The model elements of control flow, function, conditions, states, actions, events, and many others form the many facets of a system. They need to be reconciled in terms of which one, or their combination, will contribute to better software development results.

One of the normal sequences of diagram is to use the structural model as the first model to be worked on. Afterwards more detailed works start on sequence diagram to fill the interactions among classes through message passing. Methods defined in a class
are expanded to become an activity diagram. This process carries on. At end of modeling, the class diagram will be modified to obtain a finalized diagram.

6.3. Ideal State of Development Order

As Nogueira et al., [7] has stated, the edge of chaos may constitute an ideal state in software development. Just as it is difficult to determine the conditions of having too much or too less structure/behavior, software modeling should aim through a structured variably changing state. This is the same as the movie which has achieved success through varying flashbacks between past and present.

As Ambler and Jeffries have illuminated in the set of modeling principles defined for agile modeling [8], the essence of modeling is balance. This balance cannot be wholly dependent on a rigid set of rules and pragmatics, but should include some kind of flexibility. The lessons from arts, including the “Architecture 101” movie should make contributions along this path.

However, this non-rigidity needs to be suitably differentiated from the so called “systematic approach” given in the definition of software engineering, such as by IEEE [9]. In essence, both are needed and they are not mutually exclusive. This one shares certain degree of similarity with the following paraphrased statement by Pressman [10], “We need discipline, but also need adaptability and agility”. In the end, I think a more generic approach borrowed from other areas, such as with lessons from out-of-discipline areas in arts, may bring benefits.

7. Conclusion

In conclusion, this paper has presented an inter-disciplinary work. The “Architecture 101” movie and software modeling are studied and compared with a view to derive hints that can overcome the difficult challenges in suitably sequencing multiple models. View, level and fragment are determined as the key commonalities in the comparison. Among the three, fragmentation remains as the open issues, while view and level reinforces each other with mutual benefits in terms of switching frequency. The paper proposed an approach that inter-mixes the three commonalities and suggests a more practical process for software development with explicit guidance in switching models. As a result the progression from model to code can be made more productive. It is of the opinion that fragments take a more important role than view and level. Rather than the current diagram-driven approach, this leads to a re-thinking on the nature of model fragment that can facilitates model progression. There are other outstanding issues, such as deriving a detailed mechanism to facilitate a balance between suitable modeling with leveling. Future works include realistic applications for purpose of further validation and verification.

References


Authors

Kai-On Chow is an associate professor in the Department of Computer Science of the City University of Hong Kong. He has worked in the industry prior to joining academia. He obtained his Ph.D. from the Hong Kong Polytechnic University. His current research interests center around software modeling.

Tak-Lm Wong currently is a Lecturer in the Department of Mathematics and Information Technology of The Hong Kong Institute of Education. He received B.Eng., M.Phil., Ph.D. in Systems Engineering and Engineering Management (specialized in Information Systems) from the Chinese University of Hong Kong in 2001, 2003, 2007 respectively. His research interests lie in the areas of Web mining, data mining, information extraction, machine learning, and knowledge management.