Is the Derivation of a Model Easier to Understand than the Model Itself?

Janet Feigenspan
University of Magdeburg
feigensp@ovgu.de

Don Batory
University of Texas
batory@cs.utexas.edu

Taylor Riché
National Instruments
riche@cs.utexas.edu

***Abstract*—Software architectures can be presented by graphs with components as nodes and connectors as edges. These graphs, or models, typically encode expert domain knowledge, which makes them difficult to understand. Hence, instead of presenting a complete complex model, we can derive it from a simple, easy-to-understand model by a set of easy-to-understand transformations. In two controlled experiments, we evaluate whether a derivation of a model is easier to understand than the model itself.**

I. Introduction

Equations are used in physics to describe the movement of planets, predict the stress load on bridges, and to anticipate interaction of atomic particles. In graduate level physics, it is standard fare for students and researchers to rederive published equations from scratch, in order to understand them [2]. It is also generally easier to remember how an equation was derived, rather than to memorize the equation outright. The reason is simple: Most equations are specific to the nuances of a particular application, rather than a fundamental identity (e.g., $E = mc^2$). The laws presented in undergraduate physics texts are usually examples of the latter; the exercises at the end of each chapter are usually examples of the former.

In software engineering, we might expect the same. Complex architectural models (typically represented as graphs where nodes are components and edges are connectors) encode a substantial amount of expert domain knowledge [8]. Without such knowledge, it is difficult to understand a model and why it has the "shape" that it does.

A host of researchers in the last 15 years [8] have suggested another way to explain software architectures: Instead of presenting an architectural model as a fully-completed "spaghetti diagram," they start from an elementary model that is readily understood, and apply a series of semantics-preserving refinements and optimizations to transform that simple model into the complex model that represents the complete architecture. Each transformation, in isolation, can be easily grasped and represents a fundamental mapping that arises in architectural designs in that domain. These fundamental mappings, which equate a computational abstraction with one of its implementations, correspond to a "law" in that domain. The sequence in which these mappings are applied to a simple architecture to produce the final architecture corresponds to an application-specific equation in physics.

It seems plausible that the derivation of a model is easier to understand than the model itself. However, "understanding" is an internal cognitive process. Plausibility is not sufficient to argue that something is easier to understand; empirical investigations are necessary. In this paper, we present two experiments, in which we analyze whether the derivation of a model is easier to understand than the model itself. The results do not confirm this expectation.

II. Objective

Using the goal-question metric [1], we *analyze* the derivation of a model *for the purpose* of the evaluation *with respect to* memorization, comprehension, and modification *from the point of view* of the software developer/architect *in the context* of pipe & filter architectures [9].

We believe that the derivation of a model is easier to understand than the model itself. The reason lies in working memory capacity, which is limited to 7 ± 2 items [6]. This means we can handle about 7 items at once; if there are more, we need support (e.g., by writing information down). If we have a complex model with considerably more than 7 items, we cannot understand it all at once, but look only at parts of the model to comprehend it. Thus, when we present an architectural model piece by piece, subjects do not have to handle more information at a given time than they can. Hence, we defined the following research hypotheses:

RH1: The derivation of model is easier to memorize than the model itself,
RH2: The derivation of model is easier to comprehend than the model itself, and
RH3: The derivation of model is easier to modify than the model itself.

III. Pilot Study

The pilot study tested our experimental material and setting (e.g., to avoid ambiguous formulations of the tasks). The material and detailed results of the experiments is available as technical report [4].

A. Material

We used two systems: Gamma (a relational database machine) and Upright (a synchronous crash-fault-tolerance server). First, Gamma is a data base machine, which is known for its innovative parallelization of hash joins [3]. In Figure 1,

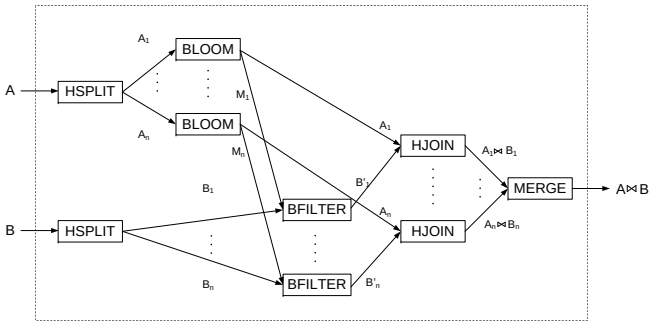


Fig. 1. Parallel hash join in Gamma.

we show how Gamma implements parallel hash joins. The relations to be joined are first split into substreams using a hash function (HSPLIT). For each tuple of the first relation, the join key is hashed and stored in a bitmap M (BLOOM). Then, for each tuple of the second relation, the join key is hashed and compared with the join keys stored in M. All tuples that have no entry of the hashed join key in M cannot be joined and are deleted from the second relation (BFILTER). Then, the remaining tuples of the second relation are joined with the tuples of the first relation (HJOIN). Finally, the joined substreams are merged and the result is returned. We decided to use the hash join in Gamma, because it is simple enough to understand in a limited amount of time, but not too simple to understand at first sight.

Second, we used Upright, a synchronous crash-fault-tolerance server [8]. A server processes client messages sequentially (Fig. 2(a)). Through server and client-message replication, a certain number of server crashes can be tolerated and still provide the image of a single server processing client messages (Fig. 2(b)). A client (C_j) sends a message to a routing box (Rt_j), which routes a copy of the message to all agreement nodes (A_1 to A_n). As part of the agreement protocol, each agreement node votes by broadcasting the message that it believes should be processed next to all quorum nodes (QA_1 to QA_k). When the quorum nodes have received a sufficient number of identical messages, that message is sent to each server replica (S_1 to S_k). Each server processes the same message, and sends its response to a quorum node (for a message from C_j, the quorum node would be QS_j) that "sits" in front of the receiving client. A quorum is taken, and a single response is returned to the client.

For both Gamma and Upright, we found a derivation of their models. For demonstration, we illustrate the derivation of Gamma in Fig. 3. We start with a hash join without optimization (Fig. 3(a)). Then, we introduce a bloom filter before the hash join, in which we delete tuples of B that cannot be joined with A (Fig. 3(b)). In the next steps, we parallelize each box by splitting the stream of tuples into substreams, processing each substream, and merging each substream (Fig. 3(c) – 3(e)). We put the parallelized boxes together (Fig. 3(f)) and by deleting unnecessary merge and split operations, we obtain the final architecture (cf. Fig. 1).

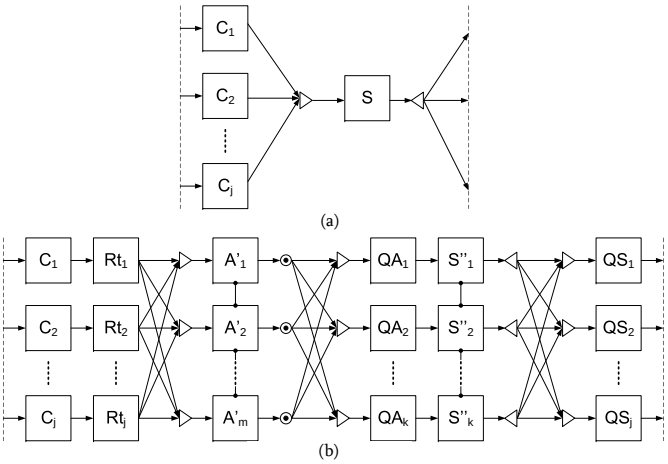


Fig. 2. Synchronous crash-fault-tolerance server. (a): before transformations; (b): after transformations.

For Upright, we started with a simple client-server abstraction of j clients $C_1 ... C_j$ sending messages to a single, state-free server (cf. Fig. 2(a)). By applying a sequence of semantics-preserving transformations, we derive the architecture of Upright (cf. Fig 2).

We created two sets of slides–one with the derivation, one with the complete model–with explanations of the boxes. One group of subjects received the set with the derivation, the other group the set with the complete model, both on paper.

We created three tasks to test each hypothesis. First, subjects should redraw the models of Gamma and Upright. Second, subjects should answer multiple-choice comprehension questions (five questions per model), for example:

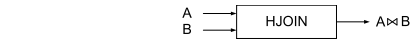
> Why are the A nodes required to communicate amongst themselves?
>
> a) To combine messages from the clients to reduce server load.
> b) The A nodes split the load coming from the clients.
> c) The A nodes run a decision making protocol that can tolerate if some number of them crash.

Third, subjects should modify the models of Gamma and Upright. For Gamma, they should delete the BLOOM box (requiring to also delete the BFILTER box). For Upright, they should add a server replica, such that the system can tolerate one more server crash. The tasks were identical for the derivation and the complete model.

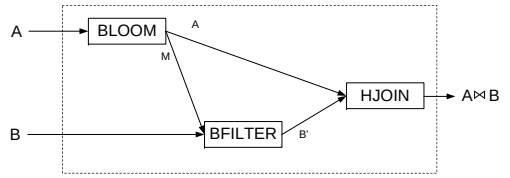
Finally, we asked our subjects to estimate the difficulty of the tasks and their motivation to solve them on a five-point Likert scale [5]. We also assessed the background with a questionnaire.

B. Subjects

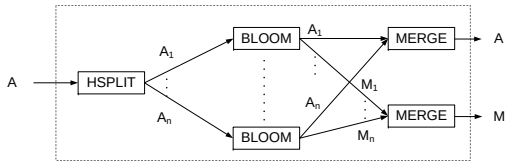
As subjects, we recruited four male PhD students from the University of Texas. Two were working in empirical software engineering, two in data bases. All volunteered and did not receive any compensation for their participation.



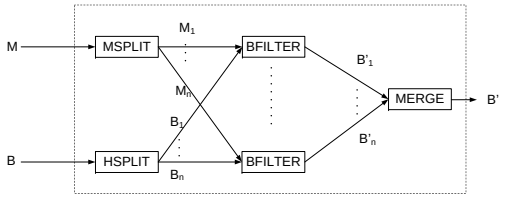
(a) Hash join without optimization.



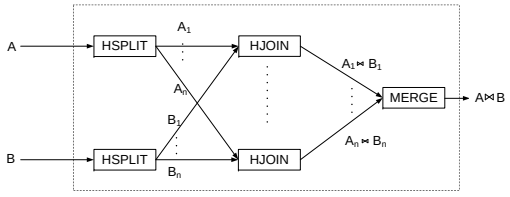
(b) Bloom filter before hash join.



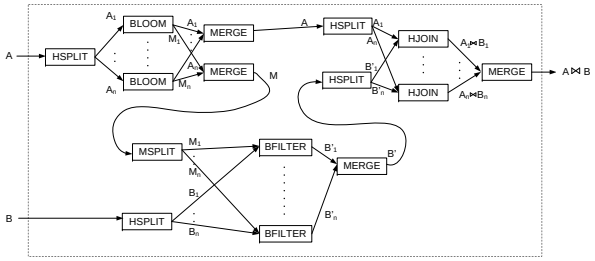
(c) Parallelization of BLOOM.



(d) Parallelization of BFILTER.



(e) Parallelization of HJOIN.



(f) Putting parallelized boxes together.

Fig. 3. Derivation of Gamma.

C. Execution

We conducted the pilot study in November 2011. Subjects completed the background questionnaire first. After an introduction, we handed out the slides for Gamma. When subjects finished reviewing the slides, we distributed the tasks one at a time. We recorded the time subjects needed to finish a task. Then, we gave subjects the questionnaire regarding motivation and difficulty. After a few minutes break, we repeated the same procedure with Upright. There are no deviations to report.

D. Results and Consequences

The pilot study yielded two important results. First, two models were too much. Although we included a short break, subjects were fatigued when working with the second model.

Second, we noticed (and subjects told us) that going through the models on their own is tedious. There were many slides, especially for the derivation, to absorb. Furthermore, subjects with the derivations felt rushed, since the other subjects had to wait for them before they could start the tasks.

As consequences, we henceforth used only one model. We selected Upright, because it is the more complex one and has more elements than 7 ± 2. Since the working memory capacity is exceeded, we assumed that the benefit of the derivation would be more evident (cf. Section II).

IV. Experimental Run 1

With our first experimental run, we evaluated our research hypotheses. The material was the same as for the pilot study, except that we had an expert present Upright.

Our subjects were undergraduate students from the University of Texas who were enrolled either in a database or software-engineering course. The mean age of subjects that worked with the complete model was 23, the mean age of subjects that worked with the derivation was 25.7. One female subject worked with the derivation. Both groups estimated their experience with crash-fault-tolerance servers as low (2 for the derivation, 1.5 for the complete model; both on a five-point scale). The same counts for their experience with modeling (3 for the derivation, 2.5 for the complete model). Subjects participated voluntarily and were rewarded with food and beverages. Subjects were aware that their performance in the experiment did not affect their grade and that they could leave any time.

We had two appointments for the experiment. In the first, we presented the derivation, in the second the complete model. Subjects could choose the appointment to their convenience, which lead to different group sizes: Eight subjects worked with the derivation, four with the complete model. We prepared a booklet with all tasks and questionnaires, which we distributed at the beginning of the experiment. After an introduction, an expert on Upright presented either the derivation or the complete model. Then, subjects solved the tasks. For each task, subjects had a time limit (based on the pilot study), after which they had to turn to the next task. The time limit was large enough so that no subject experienced time pressure.

One experimenter checked that subjects worked as planned. There was one deviation: For one subject in the derivation group, the first and third tasks were in reverse order. Since we cannot measure the performance for these tasks, we excluded this subject from the analysis.

Analysis and Interpretation

To analyze the first and third task, we counted the number of elements that did not belong to the model, were missing, or were in the wrong order. Hence, the larger the number, the more errors subjects made. For the second task, we counted the number of correctly solved comprehension questions. The

Group	Task	Median	Min	Max	U value	significant?
Derivation	1	4	2	4	7	no
Complete		3	2	5		
Derivation	2	4	2	6	10	no
Complete		4	4	5		
Derivation	3	0	0	1	19	no
Complete		0.5	0	1		

TABLE I
EXPERIMENT 1: OVERVIEW OF CORRECTNESS OF SOLUTIONS.

differences between groups are small or non-existent (see Table I). For the first task, the group with the derivation made one error more than the group with the complete model. For the second task, both showed the same performance. For the third task, the difference is smaller than one error. A Mann-Whitney-U [7] test revealed that the differences between groups are not significant for any of the tasks. Hence, we cannot accept our research hypotheses.

At first sight, this result means that the derivation does not provide a benefit, compared to presenting the complete model. However, we took a closer look at the model of Upright: There are groups of similar elements. We have a group of clients, routing boxes, agreement nodes, quorum nodes, and servers. Hence, we have six different groups of boxes. It is possible that subjects looked at the *group of boxes*, not the single boxes, which is called *chunking* [6]. Thus, the working memory capacity might not have been exceeded, which means that the model could have been too simple to show a benefit of derivation. Looking at the estimation of difficulty for each task, it was perceived as medium to easy, except for the first task (difficult). Looking at the correctness of the first task, subjects only made three to four errors. Thus, we think that the model was too easy to reveal a benefit of derivation. Hence, we conducted a follow-up experiment with a more complex model, which we explain next (we discuss threats to validity for both experiments in Section VII).

V. EXPERIMENTAL RUN 2

In the second run, we extended the server with a recovery feature, shown in Fig. 4. Via a backward loop, a server replica can now recover from a crash. To this end, it sends a message to the agreement nodes via quorum nodes asking for the correct timestamp. Furthermore, the server replicas communicate amongst each other to get the correct status.

With the recovery feature, we included two more comprehension questions, so we had seven questions. Other than that, the experimental material was the same as before.

We recruited different students from the same software-engineering course as before; six worked with the derivation, four with the complete model. The mean age was 24.3 for the derivation group, and 21.8 for the other group. One subject in each group was female. Subjects estimated their experience with crash-fault-tolerance servers and modeling as low (1 and 3 for the derivation, 1.5 and 2.5 for the complete model).

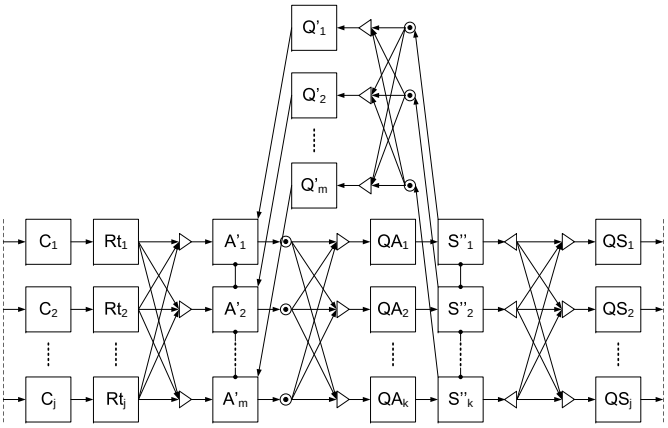


Fig. 4. Asynchronous crash-fault-tolerance server.

Group	Task	Median	Min	Max	U value	significant?
Derivation	1	2.5	1	4	10.5	no
Complete		2.5	1	3		
Derivation	2	5	2	6	11.5	no
Complete		5	2	7		
Derivation	3	3	1	4	8	no
Complete		3	1	3		

TABLE II
EXPERIMENT 2: OVERVIEW OF CORRECTNESS OF SOLUTIONS.

The experimental sessions were now held in parallel. Hence, an additional expert on crash-fault-tolerance servers explained the derivation. No deviations occurred.

Analysis and Interpretation

To evaluate whether a task was solved correctly, we used a four-point scale for the first and third task. A solution could either be completely correct (4), almost correct (3), correct to some extent (2), or completely wrong (1). An expert evaluated to which category a solution belonged. In Table II, we show an overview of the correctness. We can see that the medians for both groups are the same. Furthermore, the modification task seems to be easier than the memorization task. A Mann-Whitney-U test revealed no significant differences. Hence, we cannot accept our research hypotheses.

So, although we increased the complexity of the underlying model, we still did not observe a significant difference in the performance of subjects. The perceived difficulty of the tasks is comparable with that of the first experimental run (medium to easy difficulty). Hence, the model might still have been too simple to show a benefit of its derivation.

VI. PUTTING IT ALL TOGETHER

So far, we conducted two controlled experiments to evaluate our research hypotheses (i.e., that the derivation of a model is easier to memorize, comprehend, and modify

than the model itself). We could not accept our research hypotheses, for which we suspect three possible reasons: First, the models were too simple; second, understanding the derivation required too many cognitive resources; third, there is no benefit of derivation.

First, the model of the crash-fault-tolerance servers could be too simple. In the first experimental run, the elements of the model could have been grouped, such that the working memory capacity was not exceeded. We made the model more complex in the second experimental run. However, with grouping, the number of elements still lies in the upper bound of the working memory capacity. Replicating the experiment with a more complex model would reveal more insights into the relationship of the size of the model and the effects of derivation.

Second, subjects that worked with the derivation had to understand several transformations. It is possible that understanding the transformations required too many cognitive resources, such that the benefits of the derivation are erased. To test this hypothesis, we would have to conduct another experiment.

Third, it could also be possible that in our context (i.e., with students and our certain model), there simply is no benefit of using a derivation. Hence, whether we explain a model to students incrementally or all at once, might not matter.

The bottom line is that we need to conduct further experiments to gain better insights into the effects of derivation on memorizing, understanding, and modifying models. However, instead of keeping our results to ourselves, we like to share our experiences and initiate a discussion of our results. To gain deeper insights, we are currently planning an experiment in which one group of subjects should implement the derivation of Gamma, the other group the complete model. As dependent variable, we plan to measure development time and code quality.

VII. Threats to Validity

There are several threats to internal validity for both experiments. First, we used convenient sampling to create our samples. However, the background of subjects is similar for both groups, as we found with a questionnaire. Furthermore, we have different group sizes, because we could not assign subjects to the appointments. Nevertheless, we do not believe that it affected our result significantly, since both groups have a comparable background.

Second, we did not test the memory skill of subjects. However, memory skills can have a significant influence on our result, especially the performance for the first task. Unfortunately, there is no way to control this threat.

Third, we used multiple choice questions to measure comprehension. This could have made the tasks too easy, such that we could have measured how well subjects are able to rule out wrong answers, not how well subjects understood the model. However, we made sure that all possible answers sounded plausible and had about the same length and detail. Furthermore, none of the subjects noted that there were obvious wrong answers. Hence, we believe that we sufficiently controlled this threat.

External validity is limited by our sample and by our model. First, we only recruited students as subjects. Hence, our results can only be interpreted in the context of students and cannot be generalized to experts. Second, we used only one model of a particular domain with a certain size. With larger models of different domains we may observe different results. To reduce these limitations, further experiments with different subjects and models are necessary.

Last, our small sample threatens statistical conclusion validity. We controlled this threat by using a Mann-Whitney-U test with a table for U values especially developed for small sample sizes.

VIII. Conclusion

In physics and more generally in the mathematical sciences, understanding the derivation of an equation or model seems to be easier than understanding an equation or model itself. In two controlled experiments, we evaluated the validity of this statement. Our experiments found no evidence to support it.

Nevertheless, we believe that what is well known in physics and other mathematical sciences should hold in related domains as computer science. Hence, we still believe there is a benefit of derivation, although we have not found it yet. Thus, as a next step, we plan to let subjects implement an architecture and analyze whether implementing its derivation has a benefit compared to implementing it at once.

IX. Acknowledgements

Feigenspan's work is supported by BMBF project 01IM10002B. Thanks to Daniel Miranker for his support in recruiting subjects. Thanks to Chandrajit Bajaj and Dewayne Perry for fruitful discussions.

References

[1] V. R. Basili. Software modeling and measurement: The goal/question/metric paradigm. Technical Report CS-TR-2956 (UMIACS-TR-92-96),, 1992.

[2] R. Coker. Private Correspondence. Department of Physics, University of Texas at Austin, 2011.

[3] D. J. Dewitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H. I. Hsiao, and R. Rasmussen. The Gamma Database Machine Project. *IEEE Trans. Knowl. & Data Eng.*, 2(1):44–62, 1990.

[4] J. Feigenspan, D. Batory, and T. Riché. Material and Detailed Results of Experiment on Model Comprehension. Technical Report TR-12-01, University of Texas at Austin, Department for Computer Science, 2012.

[5] R. Likert. A Technique for the Measurement of Attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

[6] G. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review*, 63(2):81–97, 1956.

[7] N. Nachar. The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution. *Tutorials in Quantitative Methods for Psychology*, 4(1):13–20, 2008.

[8] T. Riche, D. Batory, R. Goncalves, and B. Marker. Architecture Design by Transformation. Technical Report TR-10-39, University of Texas at Austin, Department for Computer Science, 2010.

[9] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.