Mapache Dialogue Refinement Tools: a Preliminary User Study

16.02.2009 – 23.02.2009

Alexander Behring, Andreas Petter

TECHNISCHE UNIVERSITÄT DARMSTADT

26.05.2009

Technical Report No. TUD-CS-2009-019
Technische Universität Darmstadt

Telecooperation Report No. TR-10,
The Technical Reports Series of the TK Research Division, TU Darmstadt
ISSN 1864-0516
http://www.tk.informatik.tu-darmstadt.de/de/publications/

Table of Contents

1. Introduction

This report presents a user study to investigate the mechanic refinement concept of Mapache. Through this concept, properties are inherited via multiple UI variants ordered in a tree. Thus, modifications applied to more abstract UI variants automatically are applied to more concrete ones, as well. The concept tested in this work is part of the larger "Dialogue Refinement" concept. Further details can be found in (Behring, 2009).

1.1. Method

Quantification of the use of a development tool is not easy. Usability studies are complex to conduct, cf. (Olsen, 2007), because of the multitude of influence factors. These, e.g., could be the experience f the participants with a programming language and its concepts, general programming experience, experience regarding software architecture, experience regarding design of user interfaces, project background, or collected best practices. To even evaluate the concept behind a tool(s) and not the implementation of a tool itself, further factors should be taken into account, e.g, quality of the prototype, chosen design for the prototype and experience of the participants with similar designs.

Because of the reasons illustrated above, we constrain ourselves to a simple time measurement, which consequently can only give hints and no final results. Furthermore, observations and discussions with the participants yield additional information.

The participants came from a computer-science-related background. Their experience was elicited in a pre-study questionnaire. All participants had 3 or more years of programming experience.

1.2. Description of the Study

The study consisted of two tasks. For task one, participants had to replace a naming in the UI with a different version. The name to be replaced occurred on buttons and labels. Task two consisted of modifying tooltips for various buttons. These modifications were made using Mapache (the tool to be evaluated) and Netbeans[1] (the "control-tool").

Before the participants started with their tasks, they were asked to complete a pre-study questionnaire. Following it, a short introduction (approx. 10 minutes) to Mapache and Netbeans took place. Then the study tasks were completed by the participants, followed by the post-study questionnaire. Finally, a short interview with the participant concluded the study.

Experiments were selected in a cross-over design strategy: participants switched between starting with Mapache and starting with Netbeans as the tool. Using this approach, learning effects were identified. Both tasks were completed, before the tools were switched in order to complete the tasks with the other tool. In the following, groups are distinguished regarding whether they started with Netbeans or Mapache into **groups A and B**.

The next sections present the sheets used during the session, as well as the interview guide and sheet used. Then, in section 0, collected data is presented in a condensed form. The following section presents the interpreted results. For reference, the raw collected results are contained in the annex (section 11).

[1] http://www.netbeans.org/ - a programming environment, which also contains an editor for direct manipulation of graphical UIs

2. Interviewer Sheet

This section is describing the information the interviewer collects during the study.

2.1. Introduction

- Thanks & Welcome
- I make notes to capture feedback
- Results will be anonymized for publication

2.2. During Before Study Questionnaire

- Date:
- Participant name in case of questions (blinded for analysis)
- Group (circle choice)
  A (Mapache first) B (NetBeans first)
- Running number of participant

2.3. During Introduction of NetBeans and Mapache

- Impression of comprehension / skill level for NetBeans
- Impression of comprehension / skill level for Mapache

2.4. During the Study

- Start time

- When the participant corrected herself
- Errors made
- How much time is spend navigating to a control (for editing) compared to editing time (circle appropriate)

  little some more a lot

- How are elements most often edited in Netbeans

  inside the prototype gui (inline) through the properties view

- How are elements mostly selected for editing in Mapache

  by selecting them in the editor by using the interactive interpreter

- Finish time

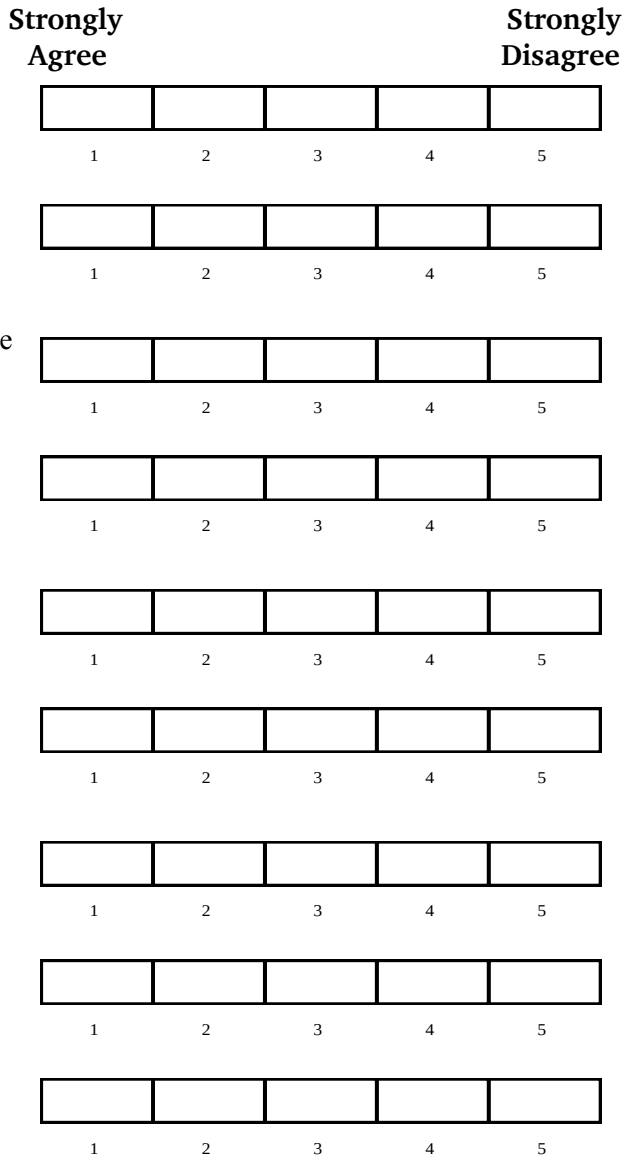
2.5. During After Study Questionnaire

Nothing special is recorded.

3. Questionnaire Before Study

The following questions are designed to help us judge your abilities used when completing the task of the study. Please fill out the following questionnaire by marking a box that best represents your opinion. In case of questions how to fill out the questionnaire, please do not hesitate to ask the person conducting the study. Thank you.

- Years of programming experience:
- Years of using Eclipse:
- Years of using NetBeans:



- Normally I program GUIs using:

- For programming, I normally use:

Thank you.

4. Introduction to NetBeans GUI Editor and Mapache Environment

A brief introduction to NetBeans GUI Builder and the Mapache Environment is given, as well as to the application to be modified. This section describes what topics are covered and how.

4.1. Application

- All UIs are very briefly looked at and their purpose is explained
- Three contexts: desktop, smartphone, subnotebook
- Naming conventions are explained

4.2. NetBeans – our baseline

- The project structure is explained
- The property view is explained
- Opening and editing an UI is explained
  - How is the JFrame selected for editing
- When in source view use the design button to return to the UI
- Use "F2" to start modifying a selected element in the UI
- All open editors are closed at the end of the introduction

4.3. Eclipse with Mapache – the candidate

- Dialogue Refinement is introduced as a form of inheritance
  - What are AUIBoxes and how are they presented in the Mapache UI
  - How is a JFrame and its children presented
  - The class label on elements
- The Refinement view is explain, incl. observing an element's refinement properties
- UI Property view is explained
- How to start editing is explained
  - Use the **interactive** interpreter
  - Opening a model and interpreting it
  - Switch on editing mode for the interpreter
- All open editors are closed at the end of the introduction

… please ask the person conducting the survey in case of questions.

4.4. For the tasks…

- Don't worry about capitalization and singular vs. plural
- Read tasks before time-taking is started
- … please ask the person conducting the survey in case of questions.
- Use triple-click to select a whole line
- Copy/Paste – how does it work on the mac

The later two subsections are inversed for group B.

5. User Study Tasks and Form of Conduction

The tasks that are completed in context of the study are describes in this section. Every participant uses Mapache and NetBeans – one after the other – to perform the required tasks (within-subject design). Of all participants, two groups are formed to create a Cross-Over study design. **Group A** first uses Mapache for completion of their tasks, whereas **group B** first employs the NetBeans GUI builder.

The use case is a guide application that allows users to retrieve information about objects in the computer science facility of the TU Darmstadt. The following devices are to be supported:

- Desktop PC with >> 640*480 px
- Subnotebook with 640*480 px
- PDA/Smartphone with 240*320 px

5.1. Task 1: replace "Object" by "Exhibit"

All UIs were designed with the same texts. Now, in all UIs, replace the text "object" by "exhibit" please.

Reminder: with Mapache, use the "Interpreter Action" to show an UI; the selection in the interpreter and the editor are linked.

The elements listed below are to be edited. All elements visible in the UIs with "object" in them are in the list. Additionally the frame titles are to be modified. I.e. all occurrences of the word "object" in the UI are to be replaced.

- Affected elements
  - Continue Tour
    - Previous Object
      - Title field
      - Label "previous object"
    - Continue To (Title field)
    - "How to get to the next object"
  - Main
    - Frame title
    - Title Label
    - See object button
    - Search object button
  - Object (don't change this)[2]
    - Frame title
    - Title field
    - Area field
    - Previous button
    - See object button
  - Object Seen
    - Frame title
    - Title label
  - Query
    - Frame title
    - Title label
  - Query result

[2] This list omits the description field in the Object form – the field is not included into the evaluation

        - Frame title
        - Title label
        - Button show object
    - Start a tour
        - n/a

- Success-criteria
    - All elements listed above are converted
- Should-do-it-way
    - Mapache
        - Look at UIs through interactive editor
        - Use refinement, apply in top level interactors only
    - NetBeans
        - Iterate over all UIs and convert fields

5.2. Task 2: Create Tool Tips

Please create tool tips for the elements below. The tooltips are already put into an external editor, because we don't want to measure your typing skills. So please copy and paste.

- Object
    - "I see a different exhibit" button → "Use this function to get help identifying an object that you are standing in front of"
    - "Back to main menu" → "This button returns you to the main menu. If you are on a tour, the tour is aborted."
    - "Continue tour" → "Continues to the next exhibit. Upon activation, a guide is shown where the next exhibit is."
    - "Previous exhibit" → "Returns to the previous exhibit."
- Query results
    - Table → "Use the table to identify the exhibit you are interested in. Select it and press the 'show selected exhibit' button."
- Continue Tour
    - "Continue" → "Press this button in order to display information on the next exhibit"

- Affected elements
    - See task description above
- Success-criteria
    - All elements listed above were added the correct tooltip
- Should-do-it-way
    - Mapache
        - Open a model
        - Use refinement, apply in top level interactors only
    - NetBeans
        - Iterate over all UIs and set tooltip

6. Questionnaire After Study

Thank you for completing the study tasks. In order to evaluate the usability of the concepts under study, we would like to ask you to answer the questions below. As in the first questionnaire, it is done by marking a box that best represents your opinion. In case of questions how to fill out the questionnaire, please do not hesitate to ask the person conducting the study. Thank you.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
2. I found the system unnecessarily complex	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
3. I thought the system was easy to use	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
4. I think that I would need the support of a technical person to be able to use this system	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
5. I found the various functions in this system were well integrated	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
6. I thought there was too much inconsistency in this system	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
7. I would imagine that most people would learn to use this system very quickly	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
8. I found the system very cumbersome to use	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
9. I felt very confident using the system	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
10. I needed to learn a lot of things before I could get going with this system	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5

Thank you.

7. Interview Guide

Following the study and questionnaires, an interview will be conducted. These topics are suggestions of what to discuss with the study participants.

- What language to use German or English?

- Refinement View
    - How important was it to complete your tasks?
    - Impression?
    - Did it and how did it help you orient?
    - Was it possible to locate interactors and refinements in it?
    -
- Your overall impression
    - How do you rate the potential of this approach?
    - Please compare other approaches or products to this approach…

Thanks!

After the interview
- The workspaces of the participant is saved and archived with the participants number
- The workspaces of NetBeans and Mapache are restored to their default state

8. Collected Feedback of the Participants

This section documents the feedback that was gathered from the participants in the concluding interview.

8.1. Participant 1

- Not clear whether UIs for mobile devices can be developed using this approach, as it is very Java based (and Java mostly not on mobiles)
- "Usability very good": easy to use, not very complex, integration is very well; maybe Eclipse-knowledge is beneficial
- It is great to see the abstract models (root UI models) – and not just have some formal representation of them. Compared to the SotA, this approach is very graspable and straightforward for the user. You don't have to tune abstract properties in order to produce the correct concrete UI out of an abstract UI, as in Vaquita.
- Synchronization between different parts very good (even though it didn't work all the time ;)
- Main editor (GMF based) is not so great, because it does not serve a definitve purpose
    - It could be improved by allowing filtering or grouping of elements based on the current task
    - For example, type "label" in a search field and all elements that carry one show up – or even their label property pops up right besides them
    - Or type a string like "object": all elements that contain this string in a property value show up / are filtered
- Refinement View was not needed to solve tasks
- Preview is very great
- XCode Interface Builder is a great editor, but Mapache could supersede if editing is available at runtime
- Suggestions
    - Replace all elements of this type by this type
    - Replace all elements of this type in this UI … (e.g., in a smaller UI)

8.1.1. Observation Notes

Selected notes from the observation.

- User initially used double click in NetBeans to edit labels, thus often opening the code view.
- For NetBeans task 2, strategy to leave related windows open for concurrent modifications was used.
- Label property seems to be hard to find, when starting search from interpreter GUI (after selecting an element there).

8.2. Participant 2

- Massive problems with the usability of the setup occurred:
    - 2 screens are used: the right one was not aligned at the bottom but at the top, thus the mouse was often searched → this only affected Mapache, as NetBeans only used one screen
    - Synchronisation problem was a problem quite often
    - → time measured not representative
- Concept was found to be very good and easy to use (instead of the prototype implementation)
    - Products are tested better…

- Directly editing inside the UI is wished (easier, because of shorter ways[3])
  - With Netbeans, less mouse movements were necessary
  - … when the interpreter is outside eclipse, this costs movements, too…
- The need to switch on the editing mode for the interpreter separately is cumbersome
- The use of Netbeans to solve the tasks was monotonic – the tested concepts with Mapache are very good if you have multiple UIs
- "Reminds me of early refactoring tools" (the Mapache tools)

8.2.1. Observation Notes

Selected notes from the observation.

- At first started Swing Interpreter (non interactive one) in Mapache - lost 30 seconds for it in task1
- Searched 30 seconds for frame interactor in Mapache.
- Various times tried a double click to open sub-section in Mapache property view
- Two screens are used with Mapache, but the right one was not aligned at the bottom with the left one – rather at the top. Thus the mouse was often searched for. This only affected Mapache, as NetBeans only used one screen.
- Very thoroughly in his actions.

8.3. Participant 3

- It was great to not have to click like 3 times to reach your goal when you use Mapache (rem.: there were 3 UI versions to be edited one after the other in NetBeans)
- Couple of problems with the prototype state of Mapache
- Mapache is intuitive like NetBeans
- Editing directly inside the interpreter would be very cool
- Mapache is useful

8.3.1. Observation Notes

Selected notes from the observation.

- Task 2 was started wrong by setting text property first instead of setting the tooltip property: took about 1,5 minutes of rework.
- Often clicked on Mapache subproperty line in expectation to edit it.

8.4. Participant 4

- It was good seeing the whole frame – as it would be when the app is running – in Mapache. Netbeans instead left out the title bar.
- Visual Studio has an interesting facility to script changes to UI elements; it's very heavy (learning), but for huge projects very useful
- Mapache saves you time, but needs some practice
- Netbeans usage easy, because the type of the IDE is known through other IDEs
- How about integrating a snippet navigator / possibility to insert code snippets?

[3] this is similar to the remark of participant one, who suggested a grouping / filtering capability. The whole point besides both suggestions is that the interaction would become faster – shorter paths to travel with the mouse and less elements to check / have in view.

8.4.1. Observation Notes

Selected notes from the observation.

- Very confident NetBeans user - the experience with other IDEs pays off.
- Often clicked on GMF editor compartments in Mapache, which needed time to correct
- Like participant two, often moved elements out of the AUIBox in the editor, which had to be moved back.
- Had trouble targeting the mouse.

8.5. Participant 5

- Practical, but some issues with the prototype state
  - 2nd task had synchronization issue between interpreter and property sheet
- Container (GMF editor) for GUIs with many elements complex
- Concept useful for hierarchical systems

8.5.1. Observation Notes

Selected notes from the observation.

None.

8.6. Participant 6

- NetBeans is more well arranged than Mapache
- Cf. Visual Studio which has a combo box for selection the localization (Remrk: the participant was comparing the localization possibilities – which allows to set properties of elements depending on localization – with the possibility in Mapache to set properties depending on the refinement)
- Mapache is beta….

8.6.1. Observation Notes

Selected notes from the observation.

- Thoroughness is reduced when using NetBeans in comparison to Mapache use, checking finished items is not done on item level (as when using Mapache) anymore, but on window level.
- Problems using and positioning the mouse
- The need for opening the subproperty to edit is superfluous.
- For task two, every item was checked after completion in Mapache.

8.7. Participant 7

- One mistake is very time consuming – costs 5-10 seconds
- Opening the correct tab and the property item is cumbersome
- Relevant properties are not immediately visible – as they are in Netbeans

8.7.1. Observation Notes

Selected notes from the observation.

- Forgot a great deal of the changes in NetBeans for task one - after reminding, rest was completed.
- Not seeing the whole UI in NB was a problem - some parts were forgotten.
- After missing a lot of the elements in task one, now things are dbl-checked in task two.
- To open "UI Properties" tab in Mapache's property view is a hurdle - sometimes this took a while.

9. Results

This section presents the measured, quantitative results. Starting from participant 5, the following bugs were fixed in Mapache:

- Synchronization between interpreter and editor works more reliable
- Interactive Interpreter now is in mode "editing" when it is started
- Interpreter MetaUI now is always on top when UI itself is shown

This is respected in the analysis: data marked as *series one* contains participants 1 through 4, whereas *series two* contains participants 5 through 7.

Series one participants conducted their study with Mapache released on 16th of February 2009, series two participants used the release of the 23rd of February 2009.

The overall learning is that the concepts of Mapache do seem to have potential, but because of usability issues and other experimental setup-problems, no significant quantitative results could be obtained. This actually comes as no surprise, because, as Olsen pointed out in (Olsen, 2007), usability experiments are probe to errors when evaluating UI Systems Research. But on the other hand, using usability-test-settings can yield valuable insight for the researcher, such as preliminary input about what aspects of the concepts need to be looked into more deeply.

9.1. Learning Effect

Task one delivered a surprising result: great variance. We attribute this to the setup that seems to have been non-optimal. We even had to discover learning effects the experiment suffered from.

	Time in minutes:seconds	First Environment	Second Environment
Series One	**Average**	13:43	08:52
	RMS Deviation	03:18	01:20
	Learning effect based on average		04:51
Series Two	**Average**	13:43	08:52
	RMS Deviation	03:18	01:20
	Learning effect based on average		02:49

Table 1: Learning effect had an impact on **task one** completion times for the both series of participants.

Table 1 shows an estimate of the learning effect the experiment suffered from. The numbers in the cells are times spend completing task one for both series. The distinction is in this case not made between Mapache and Netbeans, but rather whether it was the first time completing task one or the second. A similar picture can be drawn for task two. But here, the numbers are not as clear as for task one, cf. Table 2. This can either be attributed to the nature of the task (task two was so much easier to learn), or – as we think more likely – to that the users became accustomed to the tools.

	Time in minutes:seconds	First Environment	Second Environment
Series One	**Average**	05:15	04:17
	RMS Deviation	01:27	01:27
	Learning effect based on average		00:58
Series Two	**Average**	04:32	03:51
	RMS Deviation	00:24	01:41
	Learning effect based on average		00:40

Table 2: Learning effect for **task two**. Both series seem to be affected, but much less than in task one.

Such a learning effect is a great problem for the experimental setup; and one reason, why the numbers presented for the task measurements must be read with great care. Comparing to Olsen's arguments in (Olsen, 2007), this is no surprise. It just is **the confirmation that learning effects are a problem in such complex experimental settings**.

Another issue could be errors – when producing an error in such complex tasks, it is costly to redo/repair it. Taking up to 10 seconds for a correction in task two, this can be as much as 5% of the overall test duration: thus increasing variance greatly. **Identifying how many errors were made and corrected helps improving usability** (cf. Section 9.3).

9.2. Task Execution Times

Task execution times were measured to investigate the effect of the mechanic refinement in our modelling concept, cf. (Behring, 2009). The numbers presented is the speedup using the Mapache tools. They were derived by normalizing the time measured for the respective task to the total time needed for all tasks in both environments (two with Netbeans and two with Mapache). This way, we can account for user's common pace. From the normalized Netbeans time share for a task, the normalized Mapache time share for the task was substracted, yielding the results in Table 3.

Difference of time shares normalized to total time needed		**Speedup Task 1**	**Speedup Task 2**
Series One		11%	8%
		-24%	2%
		9%	8%
		-15%	0%
	Average	-5%	4%
	RMS Deviation	17%	4%
Series Two		-4%	4%
		-4%	0%
		23%	11%
	Average	5%	5%
	RMS Deviation	16%	5%

Table 3: the measured and normalized task execution times for both tasks and series of participants.

Thus, we tested out hypothesis:

Hypothesis: Using Mapache results in faster task execution times.

As we discovered serious learning effects (cf. section 9.1) for task 1, we only performed statistical analysis on task 2 data. Furthermore, the results of series two, task two obviously cannot be taken as significant on a better level than approximately 10% and thus were not analysed in detail. For series one, task two, we performed a Kolmogorov-Smirnov-Test to check for normal distribution, which was accepted on a level of $a=0.05$ ($0.29 <$ critical value 0.62). The subsequent **T-Test accepted our hypothesis that using Mapache results in faster task execution times on a 6% significance level** ($t=2.27 > 2.156$ – the critical value).

9.3. Error Rates

Naturally, users make mistakes. Especially when modifying multiple UIs in a monotonic fashion (as some study participants noted). Thus, we checked:

Hypothesis: Using Mapache results in a smaller error rate.

Error rate is in this respect defined as: number of errors per edited UI. Consequently, for measuring, the number of errors was counted and then divided by the number of UIs that had to be edited (21 for NetBeans and 9 for Mapache).

Measured results were:

Participant No	Group	NetBeans	Mapache
1	B	57%	56%
2	A	38%	0%
3	B	5%	0%
4	A	29%	11%
5	A	0%	0%
6	A	48%	11%
7	B	0%	0%
Average		20%	4%
RMS Deviation		21%	6%

Table 4: Error rates measured for editing the UIs. Group A started with Mapache, group B with NetBeans.

The numbers presented in Table 4 were tested on significance to the hypothesis. Participant one was removed, as it seems to be an outlier (error rates with both environments were extraordinary high). On the remaining data, a Kolmogorov-Smirnov-Test accepted the hypothesis that the data is normally distributed ($0.24 < 0.521$ – the critical value for $n=6$, $a=0.05$). A subsequent **T-Test accepted our hypothesis that using Mapache results in smaller error rates** ($t=2.25 > 2.02$ – the critical value for $n=6$) on a 5% significance level.

This is not surprising, because editing every single UI one after the other when using NetBeans causes oscitancy. Mapache helps to reduce oscitancy when editing multiple UIs. Comments made by the participants are in line with these numbers. Two participants noted that the tasks are monotonic and annoying when using Netbeans – as subtasks had to be repeated quite a lot.

9.4. Summary of results

The important results were:

- Task one suffered a great learning effect in contrast to task two.
- The use of Mapache is found to speed up task two by 4% on average with a significance of 6%.
- The use of Mapache is found to reduce error rate

9.5. Mapache Lessons Learned

9.5.1. Experimental setup and Usability

The feedback given by some of the participants hints that usability issues were a key problem when trying to obtain quantitative results from the experiment. For further studies, learnings regarding the experimental setup were made. Especially, **the hardware setup should be tested with a variety of people** in advance to avoid handling problems with dual monitor setups (i.e. ideally only use one bigger one) and the mouse (cf. participants 2 and 4).

Depending on what should be measured, editors should be arranged "on even terms" that includes having similar mouse pointer ways between the edited UI and the properties view (the setup used here had much longer ways for Mapache). Furthermore, the areas to be clicked with the mouse during the tests (e.g., the property lines to edit) should be of equal size. This could be accomplished by, e.g., setting font sizes for the editors, such that **there ideally is no difference for the tasks regarding to Fitt's Law**.

Unnecessary components should not be "in the user's way" so that she does not have to make a decision that is not needed for measurements. For example to show two different interpreters for choice but only one of the two is the correct one to edit UIs (cf. participant 2).

9.5.2. Editors (Eclipse-Plugins and Interpreter)

The **Mapache editor was not optimized with respect to Fitt's Law** (cf. previous section). It cost double as many clicks to edit an item in Mapache as it cost to edit it in NetBeans. There were two reasons: i) no in-place editing in the WYSIWYG editor was possible, and ii) the properties view had an additional tab to open and a tree item to open in order to get to the editable value of a property.

Both issues not only caused delays, but also confusion among participants (cf. participants 2, 3, 7). They tried to edit properties by double clicking on the top-level tree item were in vain – expanding it first was an unnecessary step. Also, properties were searched, because the wrong tab opens initially (cf. participant 7).

The **life interpreter was well accepted among the participants**. The possibility to see the fully UI when editing, and that UI editing could even be possible at application runtime fascinated the participants (1, 4). The integration of the life interpreter with the other editing tools was well-accepted (almost all participants used the life interpreter to select the element to edit).

In contrast, the **GMF[4] based editor** was only used for workarounds of the selection bug (fixed for series two). Its **visualization is too complex and unhandy**, as well as overflowing when more elements are in the UI. Ideas for improvement included grouping and filtering abilities, as well as searching for element properties and values.

9.5.3. Dialogue Refinement

Feedback and gathered results regarding the Dialogue Refinement concept were not able to support it quantitatively, but qualitative results show great acceptance and potential (participants 1, 2, 4). The deep integration of WYSWIG-based editors with the concept of abstract models was found to be a good idea (participant 1).

[4] Eclipse Graphical Modeling Framework, http://www.eclipse.org/gmf/

The need for a concept like Dialogue Refinement was illustrated by the strategy employed by some participants in NetBeans (leaving all related UIs open and make changed to elements concurrently in all UIs). Furthermore, **error rates, as presented in section 9.3, hint that the quality and consistency of the UIs can be increased by avoiding tedious tasks and employing elaborate concept such as Dialogue Refinement**, which could also be integrated into NetBeans.

The System Usablity Scale questionnaire (section 2.5) revealed that the users felt fairly confident when using Mapache (average SUS score of 75, rms deviation of 13). For future studies, a Likert-scale based direct comparison between Mapache and its related work could yield interesting feedback, as well.

10. References

Behring, A. (2009). Rapidly Modifying Multiple User Interfaces of one Application. *ICSOFT 2009* (S. 344-347). Sofia: INSTICC Press.

Olsen, D. R. (2007). Evaluating User Interface Systems Research. *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology* (pp. 251-258). ACM.

11. Annex: Raw Collected Results

				Observations								Collected Data				
				Navigation time		NetBeans Editing		Mapache Selection		Skills		Study Before			Likert-Scale	
UserNo	Date	Time	Group	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2	NetBeans	Eclipse	Progr. Experience	Years using Eclipse	Years using NetBeans	Native Speaker	Hardly speak English
1	17.02.2009	11:00	B	4	4	0	1	0,5	0,5	2	3	9	5	0,5	1	5
2	18.02.2009	14:50	A	4	4	0	1	0,5	0,5	1	1	19	0,5	0	4	5
3	18.02.2009	16:05	B	3,5	4	0	1	0,5	0,5	1	1,5	5	5	1	1	5
4	19.02.2009	15:10	A	4	4	0	1	0,5	0,5	1,5	0,5	10	3	0,5	1	3
5	23.02.2009	14:00	A	4	4	1	1	0	0	1,5	1	3	3	0,5	1	5
6	23.02.2009	15:00	A	4	4	1	1	0	0			12	2	0	2	5
7	23.02.2009	20:00	B	4	4	0	1	0	0	1	1	3	0	0	3	5

Navigation time:

1 - little
2 - some
3 - more
4 - a lot

NetBeans Editing:

0 - using the GUI preview
1 - using the property view
… 0 to 1 possible

Mapache Selection:

0 - using the GUI preview
1 - using the GMF editor
… 0 to 1 possible

Skills

0 - poor
5 - god-like

										Study After			
UserNo	**New Tools?**	**Love standard**	**Inheri-tance**	**Like shallow inherit**	**Often create GUIs**	**NetBeans a wife**	**WYSWIG daily**	**GUI Tools**	**Programing tools**	**Guess use freq**	**Unnecessary complex**	**easy to use**	**Support needed**
1	2	3	3	3	2	5	1	Xcode + Interface builder; Eclipse, Visual Editor	Xcode; Eclipse (+plugins like Aptana)	4	1	5	1
2	2	4	3	3	3	5	2	Interface Builder	Objective-C (Xcode)	5	3	2	1
3	4	2	2	2	4	5	5	build JSPs	Eclipse	3	2	4	1
4	2	1	1	2	1	4	3	C++ Builder X (2005), Java Eclipse DIE, C# Visual Studio 2008	Zen Studio (PHP), Visual Studio (often)	5	2	4	2
5	3	2	3	2	3	4	4	Eclipse	Eclipse	3	2	4	1
6	2	3	1	4	3	5	3	Visual Studio, Windows Forms for .Net, Delphi 5, Notepad, Visual Studio Code Editor	Visual Studio, Eclipse, Delphi 5 for legacy code	2	4	2	2
7	5	3	3	3	5	5	5	none - don't do it…	C, VBA in Excel	3	2	4	1

							Times measured (Minutes) NetBeans Times		Mapache Times		Comments Usage of
UserNo	**well integrated**	**too inconsistent**	**learn quick**	**cumbersome**	**Confidence**	**Learn a lot using**	**Task 1**	**Task 2**	**Task 1**	**Task 2**	**NetBeans**
1	5	1	5	1	5	1	11:27	04:38	08:24	02:31	user initially used double click to edit labels thus often opening the code view; for task 2, strategie to leave related windows open for concurrent modifications was used
2	4	1	4	3	3	1	10:00	04:22	19:36	03:33	employed a different strategie than participant one for task 2: one UI after the other was added tooltips, setting the same tooltip in all Uis was not done in one step
3	4	3	4	2	4	1	12:49	08:31	09:52	04:12	early started to use copy/paste to easy reoccuring tasks; checked his actions after completion of an UI; for task2, same strategy as participant 2 was emploed (one UI after the other); task 2 was started wrongly by setting text property first instead of setting the tooltip property: took about 1,5 minutes of rework; phone call in task 2 Netbeans: 15 seconds
4	4	4	5	1	4	1	07:10	06:05	12:00	06:04	very confident user - the experience with other IDEs pays off
5	4	2	4	3	3	2	07:49	05:20	08:47	04:23	
6	3	1	3	3	2	1	08:52	Stop watch failure	10:00	04:13	opened all three platform Uis for one UI at the same time; one UI was not saved (approx 15 secs for rework); thouroughness is reduced in comparison to Mapache use, checking finished items is not done on item level but on window level; task2: learned tripleclick to select whole line
7	3	1	5	2	4	2	13:33	04:59	07:11	02:02	Short acclimatization phase then productive very quickly; NB bug: when edigin a button label inside the GUI and dbl-clicking on other UI to load when finishing, the edit is discarded; forgot a great deal of the changes - after reminding, rest was completet; not seeing the whole UI in NB was a problem - some parts were forgotten;after missing a lot of the elements in task 1, now things are dbl-checked in task 2

UserNo	Mapache	both
1	hard to find label property when starting search from interpreter GUI; artefacts of selection frames in interactive interpreter seem to have been disturbing; compartments of GMF editor figures sometimes disturbing (because then the prop view did not show the interactor); sync of selection between interpreter and editor did not always work	after some time (50% through netbeans task 1) copy and past was used; used SVN to track which Uis edited; prestructured the provided tooltips with blank lines to separate Uis
2	at first, started Swing interpreter (non interactive) - lost 30 seconds in task1; searched 30 secinds for frame interactor; various times tried a dbl clik to open sub-section in property view; after 15 mintes, a strategie against buggy synchronisation of selection was brought up and employed: select interactor in interpreter, reselect it in GMF editor; edit its properties; activate editing often was not pressed directly after starting the interpreter, it cost some time; creating a strategy took much longer for participant 2 than for 1 and 3 (e.g., no copy and paste was used for task 1); during task 2 the usage strategie was much better adapted to Mapache and the prototypical implementation; 2 screens are used: the right one was not aligned at the bottom but at the top, thus the mouse was often searched - this only affected Mapache, as NetBeans only used one screen	some instructions in task 1 were ambigous/unclear: "section label" (occured in Mapache); very thoroughly in his actions
3	often clicked on subproperty line in expectation to edit it; after g.5 minutes strategy for sync bug (click in itnerpreter, click in GMF editor, edit) was employed	
4	Often clicked on compartments, which needed time to correct; non-working synchronization was probelamtic at first; left all interpreters and Uis open; at the beginnin trouble with sync-bug, "how do I get from WYSWIG to property", starting the interactive interpreter with activate editing button; often (like SP2) moved elements out of the AUIBox in the editor, which had to be moved back	had trouble moving the mouse, especially with Mapache, as it involves 4 repositionings and clicks for editing instead of 2 clicks with 1 repositioning in Netbeans; tooltips in Netbeans were significantly slower, as they involved more mouse movements - cf. other A-group participants; commented about his large background in GUI programming (mostly Visual Studio)
5	The user is much more fluent with the selection bug not being there anymore; hmm - in task 2, it was there again - task2 was completed with GMF editor only :(; multiple open interpreters could confuse the user; user is expressing his fear the task will be cumbersome and boring with Netbeans	
6	every completed item was checked on the task description; probems using and positioning the mouse; the need for opening the subProperty to edit is is superfluous; for task 2, every item was checked after completion, as well; selection for the query result UI defect (cost approx. 30 secs)	Fitts-Law… the area (a property item) that needs to be hit in Mapache/Eclipse is much smaller that the one in NetBeans
7	To open "UI Properties"! tab in property view is a hurdle - sometimws this took a while	

	Times Corrected					Errors														
	NetBeans		Mapache			Mapache									NetBeans					
UserNo	**Task 1**	**Task 2**	**Task 1**	**Task 2**	**SUS Points**	**Continue Tour**	**CT: Tool-tips**	**Main**	**Object**	**Obj: Tooltips**	**Object Seen**	**Query**	**Query Result**	**QR: Tool-tips**	**Continue Tour-Desk**	**Tool-tips**	**CT-Smart**	**Tool-tips**	**CT-Subnote**	**Tool-tips**	**Main**
1	11:27	04:38	08:24	02:31	97,5	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	1
2	10:00	04:22	18:36	03:33	72,5	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
3	12:49	06:46	09:52	04:12	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	07:10	06:05	12:00	06:04	80	0	0	0	0	1	0	0	0	0	2	0	0	0	1	0	0
5	07:49	05:20	08:47	04:23	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	08:52	04:12	10:00	04:13	52,5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
7	13:33	04:59	07:11	02:02	77,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Time corrections
according
to notes

															Overall			
UserNo	**Object-Deskt**	**Tool-tips**	**Obj-SmartP**	**Tool-tips**	**Obj-Subnoteb**	**Tool-tips**	**ObjSeen**	**Quer-Res-Deskt**	**Tool-tips**	**QR-SmartP**	**Tool-tips**	**QR-Sub-noteb**	**Tool-tips**	**Query**	**Net-beans Errs**	**Mapache Errs**	**NB Errs Norm**	**Map Err Norm**
1	1	0	1	0	1	0	1	2	0	2	0	2	0	1	12	5	57%	56%
2	0	0	0	0	0	0	1	0	0	1	0	1	1	0	8	0	38%	0%
3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	5%	0%
4	1	1	0	0	0	0	1	0	0	0	0	0	0	0	6	1	29%	11%
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0%	0%
6	1	0	2	0	1	0	0	2	0	1	0	1	0	1	10	1	48%	11%
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0%	0%

Normalized Errs

Normalization: divide no Errs by Uis that had to be modified (a more dteailed level would be the same for Mapache and Netbeans)