INF5500 - Project assignment:
When is Ruby more efficient than statically typed languages for developing solutions in consulting organizations?

Eivind Barstad Waaler
eivindwa@student.matnat.uio.no

December 14, 2008

Contents

1 Introduction

1.1 Problem formulation

When is Ruby more efficient than statically typed languages for developing solutions in consulting organizations?

1.2 Definitions

Ruby is a dynamically typed programming language, not requiring compilation of the source code. The type checking is performed dynamically at run-time. It is fully object oriented and runs on a variety of platforms and different runtimes. With statically typed languages we mean programming languages where type checking is performed at compile-time. Examples of such languages are C++, Java and C#. This report will mainly focus on comparing Ruby with Java.

By *when* I mean for what kind of tasks/problems/projects and with *more efficient* being a measure of the total development time taken from "complete" requirements to a production ready system, such that a *more efficient* language will solve the task in less time than whatever language it is compared with. A *consulting organization* is in this context seen as an organization that performs custom development projects with teams of people, projects being limited in terms of time and money. *Bekk Consulting AS* (BEKK) is an example of a consulting organization, and is used as case in this study.

1.3 Motivation

The Ruby programming language has been increasingly popular [2] and marketed as the "next big thing" in some enterprise environments the last couple of years. Environments where statically typed languages like Java and C# have been the dominant languages of choice for many years. I feel that much of the information given about Ruby is presented in a very biased way, so this report aims at looking at evidence and evaluating some of the Ruby languages claimed advantages.

BEKK is an enterprise consulting company located in Oslo and Trondheim. They have for several years mainly focused on delivering solutions developed in either Java/J2EE or C#/.NET. They are currently trying out Ruby in a few projects, and will be used as main example study in this report. The author or this report has been an employee of BEKK for the last 8 years, and has currently started on a 2-year master program at UiO.

2 Method

2.1 Theory search

I started searching in the ISI Web of Knowledge. As Ruby is still a "new" language in many academic communities I started out with a very simple query searching for Ruby in topic and title:

Topic=(ruby) OR Title=(ruby)

This yields 5,651 results, but mostly articles about lasers and other uses of the gem ruby. Also many of the results are actually older than the Ruby language itself (released in 1995 [2]). I therefore refined the query by using publish year (from 2000 to present) and only selecting results from computer science related topics:

Topic=(ruby) OR Title=(ruby) AND Year Published=(2000-2008) Refined by: Subject Areas=(COMPUTER SCIENCE, SOFTWARE ENGINEERING OR COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE OR COMPUTER SCIENCE, THEORY METHODS OR COMPUTER SCIENCE, INFORMATION SYSTEMS OR COMPUTER SCIENCE, HARDWARE ARCHITECTURE)

This search gave me 71 results, most of which have something to do with the Ruby programming language. Looking through the results I see that they consist mostly of tutorials or other introductory articles about Ruby and Ruby on Rails. No articles were found actually showing any empiric comparisons of Ruby and other programming languages.

I then tried the Google Scholar. This yields a lot more results than the ISI Web of Knowledge. The search phrase I ended up using here was the following:

ruby AND rails AND java

This search yields 1,110 results. I browsed through the pages of results looking at the titles and clicking on the ones that looked interesting. There are quite a few results comparisons between Ruby and other languages, but mainly blog-entries or other quasi-scientific sources. I ended up choosing one reference from the search results, a master thesis from the University of Göteborg [3].

I also asked on an internal discussion board in BEKK about tips for relevant references showing actual experiences with Ruby. From this I ended up with two references, a master thesis from NTNU [1] and a book [4].

2.2 Interviews

Since BEKK already have used Ruby on several projects I decided to interview participants on those projects. The interviews were conducted by email. The questions were asked in Norwegian and are given in Appendix A. I divided the questions in 6 blocks:

- Background - Identify juniors and seniors and their prior knowledge.
- Project - Information about the project(s) using Ruby.
- Lessons Learned - Identify actual Ruby experiences.
- Methodology - Was there any difference in methods used?
- Future recommendations - What are the developers technologies of choice?
- Other - Allowing them to add more information in general.

The email questions were sent to 19 people having experience with Ruby, out of which 13 replied. One of the replies was excluded as he did not really have any experience and simply replied stating this. The full interview transcripts are given in Appendix B. An interesting observation is that one of the interviewed developers is actually the author of one of the master thesis I use as reference [1], who started working for BEKK after finishing his master degree.

3 Evaluation

3.1 Book: From Java to Ruby

3.1.1 Author

Bruce Tate, an independent consultant and international speaker from Austin, Texas. He has more than ten years experience working with Java and J2EE, and has written several books on the topic. The last years he has been an active user of Ruby, and has held several talks and written books and articles about Ruby and Rails.

3.1.2 Content

The book describes methods and migration strategies when moving from using Java to evaluating and possibly using Ruby as a programming language. It has chapters describing the technologies briefly, experiences building pilot Ruby projects, integration with other technologies/systems and evaluating risks involved.

The book also presents various forms of evidence to provide selling points for Ruby. This is the material I've focused on writing this report.

3.1.3 Main claim/result

The main claims of the book when it comes to comparing Ruby and Java are the following:

- Java has become too complex with too many frameworks to choose from.
- Java is not productive enough for much application development.
- Java is an infrastructure language, and not particularly well suited to write applications.
- Project risk increases with time and complexity, both factors handled bad by Java.
- Deployment is more immature with Ruby than Java.
- Because Java is more widely adopted, Ruby requires more training and planning to use in an organization.

3.1.4 Evidence presented

Many of the claims are made based on the authors experience. Bruce Tate has been involved in many Java/J2EE projects, and was a very early adopter of the Ruby and Ruby on Rails technologies. He has had experience using Ruby on several projects.

Another part of evidence presented in the book is interviews with people having experience with Ruby. I counted 8 different interviews used in various places in the text.

3.1.5 Evaluation

The purpose of this book is not really to perform a comparison between Ruby and Java. It is meant as a guide to follow for managers who wish to start using Ruby on their projects when their former experience is mostly about using Java. I chose it as a reference partly because it gives some arguments why you should use Ruby, which I can hopefully use to answer "when", and secondly simply because it was one of the few sources found dealing with experiences of both Ruby and Java.

The author claims to be unbiased when it comes to choosing Ruby or Java. However, some quick research about him does reveal that he is a very active international speaker promoting the use of Ruby and Rails. That combined with being the author of a book promoting Ruby means he makes a living from people using Ruby, and as such might be biased towards his conclusions about Ruby. I also feel that some of the arguments he is presenting in the book show that he has a clear preference for using Ruby. His long experience with Java is of course also very interesting, and I find it fascinating that he seems to have gotten such a strong preference for Ruby in such a "short" time. This fact in itself might be one of the strongest selling points for Ruby presented in the book.

The negative claims made about Java in the book seem a bit exaggerated. I have a feeling these claims are made to make Ruby seem like a better alternative. There are a number of unsupported claims missing any empirical evidence. For example he claims that Java is a language mainly suited for infrastructure tasks because of its inheritance/legacy from the C/C++ languages, and further more that this makes it less suited to write applications. Just a quick look in pretty much any organization would probably reveal successful usage of Java as an application language, proving him wrong in this. Instead of becoming a selling argument for Ruby, I feel that this increases my scepticism.

The interviews in the book are very interesting. Some, like the interview with Martin Fowler (pages 23-25 [4]), gives an overview of his thoughts on the shortcomings of Java. There are other interviews showing some experiences using Ruby in projects. However, these projects are not randomly chosen and it is quite clear that they were included to help sell Ruby. In a way it is quite like the way big companies provide reference projects/clients. Successful stories are specifically chosen, leaving any information about negative or unsuccessful experiences out.

I would not emphasize on this book to provide evidence for using Ruby over Java. However, as stated above, this is not the goal of the book. I think the book will be very handy when I get to the design of my own suggested study on the topic. It gives many different ideas on how one could compare the two technologies and evaluate the performance and risks involved with both.

3.2 Master thesis: Polyglot Programming

3.2.1 Author

Hans-Christian Fjeldberg, consultant working for BEKK. At the time of writing the report he was a master student at NTNU (Trondheim, Norway). It is worth noting that the thesis was written using projects at BEKK as examples, and with Carl Christian Christensen (BEKK) as co-supervisor.

3.2.2 Content

A master thesis researching polyglot programming (*Programming in more than one language within the same context, where the context is either within one team, or several teams where the integration between the resulting applications require knowledge of the languages involved* [1]). The goal of the thesis is stated as *discover how polyglot programming can be and is used in businesses.*

The report first uses a literature study to come up with some claims (perceived advantages/disadvantages), and then tries to validate these claims using case-studies and interviews. It also has a section describing the platforms involved, being .NET, J2EE/JVM, and the programming languages which can be run on these platforms. There is a strong focus on Ruby as a programming language, both for the examples given and in the case-studies presented. That is my main reason for using this report as a reference; it being more relevant than the title/overall description suggests.

3.2.3 Main claim/result

The following claims were identified in the report:

- Polyglot programming offers increased productivity.
- Polyglot programming might offer easier maintenance.
- The use of polyglot programming helps motivate and change developers' perspective.
- Frameworks written in "new" (dynamic) languages offer a more natural solution to the problems solved, and lack of previous knowledge/familiarity with language not being a problem.
- Lack of tool support is a disadvantage with polyglot programming.

Since the "new" language used in the study was Ruby, I have read these claims substituting "polyglot programming" with "programming in Ruby".

3.2.4 Evidence presented

A literature study is used to describe polyglot programming and identify claimed advantages/disadvantages. Some interviews with field experts are also used to verify/help identify claims. The identified points are then evaluated using a case-study as evidence.

There are three case-studies being used. Two covering web development and one covering testing. The data from the case-studies was collected by interviewing consultants and customers involved in the projects. The transcripts from the interviews are not included in the text because of confidentiality.

3.2.5 Evaluation

This master thesis is well written, with a very thorough use of references and examples. All terms used are explained clearly with links to references or interviews. I feel that the author was not biased in any particular direction. He talks about perceived advantages and disadvantages in a sober and concise manner. However, just the fact that the author did the master thesis with BEKK as co-supervisor, and started working at BEKK after completing the work, does give reasons to suspect that he might be biased towards making the projects/case-studies seem more successful. This is difficult to evaluate as I am also an employee of BEKK, and thus in the same biased position, but still worth mentioning.

The main issue for my use of this report is that it does not directly discuss or compare Ruby with other alternatives. The focus is on using many different programming languages together, not on comparison between them. I still feel that the advantages and disadvantages are relevant for my problem, as the languages in question are Ruby and Java. In addition to the fact that the case-studies actually are performed on projects using Ruby in BEKK, which of course make it highly relevant however the problems were formulated.

The cases presented for the web development part are very interesting. An observation I made that would be worth studying further is that both projects seem to have made adjustments to the default way of using Rails. The first project has made adjustments to the ActiveRecord mechanism in Rails since the database was not directly available, and the second project has made special database-views to fit in with the Rails naming conventions. As I am trying to answer "when" Ruby should be used it would be very interesting to study how many of these custom changes can be done to the Rails framework before you start loosing some of the claimed added productivity. Unfortunately the report does not discuss this at all.

One thing that strikes me reading the case-study and discussion of the report is that it is a bit unclear where the link between the claimed advantages/disadvantages and the case is. This could be because the transcripts from the interviews are confidential and not given, but parts of it should have been used as quotes or similar to actually show how the conclusions are drawn. It is mostly statements or conclusions that seem a bit vague on the evidence. Especially the part about testing at *Statens Vegvesen* seems vague. The whole case-study is summarized in 3 paragraphs not really showing anything but an overview of what was done. Here I would have liked to see some better link with the actual experiences, and not just

what was done.

Another note about the case-study is the sources of information used. As the author himself actually writes in the "research process" section of the report, using multiple sources is important when gathering data from the case-study. Documentation, interviews, observations and participation is mentioned as possible sources of data. Yet, in the results and analysis parts I can not find any reference to other data than the interviews. This would make the results depending on the validity of the interview. I would have liked to see some cross-checking of the results found by comparing other data sources. A typical problem with interviews is project member bias, a developer might defend bad choices to make them look better or even try to make a project failure look like a success. This kind of bias would be much more difficult to discover if the interviews are the only source of data. The questions used for the interviews are not given, so it is difficult to evaluate how subject these are to developer opinions.

From my perspective of "when" to use Ruby in BEKK I find the case-studies interesting, as they were conducted at BEKK. My own interviews have been conducted with many of the same people I presume, and I will try to find links between the two sources to draw out some concrete experience.

3.3 Master thesis: Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering

3.3.1 Author

Antonios Protopsaltou, computer software professional in Greece. At the time of writing the referenced report he was a master student at the University of Göteborg, having 3 years experience as a web developer and 1 year developing J2EE applications.

3.3.2 Content

A master thesis presenting results from a study comparing Ruby on Rails modeling and code generation features against J2EE environment and AndroMDA. The main research questions are *How does Ruby on Rails support Model Driven Development and other development quality attributes in comparison to other platforms like J2EE and AndroMDA? How efficient are the existing reverse engineering tools for Ruby on Rails?*

The report first introduces the problems and methodology chosen. It then gives a background on the various technologies mentioned before it shows the results from the case-study performed. In the end the results are analyzed and a conclusion given.

3.3.3 Main claim/result

The following claims were identified in the report:

- Ruby on Rails provides a faster development environment than J2EE or AndroMDA.
- A Rails developer writes less code than a J2EE developer to implement the same functionality.
- Textual database-modelling is faster than a visual environment.
- Reverse engineering a database to create an ER-diagram is faster with Ruby on Rails than with J2EE.
- Switching between different database schemas is faster and simpler in Ruby on Rails than with J2EE.

3.3.4 Evidence presented

The evidence presented is a case-study performed by the author as part of his master thesis. It consisted of running three different scenarios on three different platforms, the goal being to find differences between the platforms.

The scenarios were (1) to create a picture diagram of the database, (2) add a new table to the database and create code to access it using relationship mapping with an existing table and (3) to change the name of a column in the database and update code to work with the new name.

The three platforms were (1) a J2EE environment using web services and enterprise java beans, developed using the NetBeans IDE, (2) a Ruby on Rails environment developed using Notepad++ and (3) an AndroMDA environment developed using MagicDraw UML editor and Eclipse Java IDE.

For the J2EE environment the scenarios were executed by a junior Java developer (with 6 months experience). For both the Ruby on Rails environment and the AndroMDA environment the scenarios were executed by the author himself.

3.3.5 Evaluation

The first impression of this report was not the best. First of all the front page for the report has wrong title and year. Seems like the title has changed, and the front page not been updated or something like that. Secondly the language used in the report is not the best, giving an impression of being a work of haste. Seeing that the author is not natively English-speaking might explain this of course.

Looking at the case-study performed in this report there were a few things that struck me. Firstly it is done studying a first year student in a controlled environment, and partly by the author doing the work himself. The way it was conducted makes it look much more like an experiment than a case-study, being more an assignment given by the author than a study of actual project work. A second thing I noticed is that when describing the third scenario with the student the author writes "so then *we* decided to go for the.." (page 25 [3]), clearly indicating that the author himself was actually influencing how things were

executed. I think this approach would have been much more convincing if there had been more participants conducting the experiment. With a large random selection of students performing both the J2EE and the Ruby on Rails scenarios there would be more data to collect and compare. The way it was done here we have one inexperienced developer doing a Java assignment, being compared to a much more experienced developer doing a Ruby on Rails assignment. How would the results have been if we switched the persons around, letting the inexperienced developer work with Ruby on Rails? The dependency on the participants pre-skills and knowledge is in this case so big, that it is difficult to trust the results.

When it comes to the three scenarios run to compare the platforms they seem to be chosen very specifically to support the Ruby on Rails paradigm. Firstly they are extremely narrow compared to the way developers actually work when developing software. Generating "CRUD" (Create Read Update Delete) type code, and making small database changes might not be where most of the time is spent. Maybe a scenario implementing a specific piece of functionality could have been executed as well, in order to be able to compare more the "regular" way of working with the different platforms. Secondly I feel that the scenarios might have been chosen under the influence of what is possible with Ruby on Rails. Could it be that the author chose the scenarios after looking at Rails and its features?

In the description of the case-study scenarios there is a sentence saying that the configuration files in the J2EE environment should be updated manually "without using NetBeans wizards" (page 22 [3]). The other references I have, together with the interviews, clearly suggest that tool-support is one of advantage J2EE has over Ruby on Rails. Thus, removing this "advantage" from the J2EE part of the case-study seems an odd thing to do. Personally I would have preferred it if the comparison was done with the working environment as close to the "real world" as possible.

The only claim found in the report that I don't question the validity with is the claim "A Rails developer writes less code than a J2EE developer to implement the same functionality". In the scenarios used in this report it is quite clear that Ruby on Rails solves the task using less code than the J2EE solution (1 loc vs. 120 loc - Analysis page 27 [3]). Again, this would be more interesting studying a real functionality example, and not just basic CRUD-functionality. So the claim seems valid for the case presented in the report, but the evidence might be questionable when trying to generalize to a wider meaning. For my use a further study into this claim is highly relevant, as less code written might be a direct reason for increased productivity and minimizing the development time needed.

All in all I don't think the evidence presented in this report will be very valuable for my evaluation. The tools seem to have been compared with scenarios adjusted to fit better to Ruby on Rails, and with one of the major perceived advantages of J2EE (better tool support) removed from the case. The claim I was most interested in, stating that a Rails developer will spend less time implementing the same functionality as a J2EE developer, is a very interesting one. However, I don't feel that the evidence presented actually proves this for a general case. He has shown that Rails is faster than J2EE in a very controlled and specific case, with two specific developers with different experience, but I can't really see how this can be generalized to a more wider meaning (as suggested in the claim).

3.4 Interviews

3.4.1 Results

12 participants answered the emailed questionnaire/interview. They are all developers or project managers on projects that are or have been using Ruby in BEKK. It seems most projects that have been run in BEKK using Ruby have been quite successful. I have identified 4 different projects implemented using Ruby on Rails, as well as a number of projects using Ruby for scripting/testing or other minor tasks.

4 of the developers have less than 5 years experience, while 8 have 5 years or more. All participants have more experience using Java/J2EE or C#/.NET than they have using Ruby, except for the 2 participants who came straight from college and thus did not have any experience at all. 9 of the 12 persons interviewed specifically mentions increased developer motivation in one way or another as a positive thing about using Ruby.

Below is a list of factors that seem to be requiring different focus when working with Ruby compared to Java/C#:

- Training/experience - There are fewer trained developers available for developing/-maintaining Ruby projects, meaning more effort must be made to ensure the project is manned sufficiently.
- Testing - Ruby seems to require a lot more focus on automated tests, mainly because it is not compiled. There is a strong TDD (Test Driven Development) culture amount the Ruby developers.
- Tool support/refactoring - Refactoring is more difficult in Ruby as there are no tools available doing the job in an automated fashion.
- Disciplined developers - Because of the lack of compilation developers must ensure that conventions are followed and tests written.
- Remoting/integration - Applications requiring big amounts of integration must do proper prototyping to ensure that these will work before proceeding with Ruby.
- Short iterations/quick feedback - Since code is written fast and thus functionality quickly finished, a short feedback loop is needed to make sure developers don't go off the requirements.
- API knowledge - Since there are no tools offering code completion the developers need to learn and lookup more in the API when using Ruby.

This list does not tell us "when" Ruby should be used or not, but it does provide important points to think about when considering Ruby. If these factors are controlled and planned the risk of using Ruby seems to be lower.

3.4.2 Evaluation

The most interesting observation in the interview answers was the high impact Ruby has on developer motivation. This was also one of the claims found in the first master thesis [1], making it an observation made by multiple observers (but from similar data). As one person put it "to be exploring Ruby is like being a kid in a candy store" (interview transcript, appendix B.10 - translated from Norwegian). The impact motivated developers will have on any project is in my opinion important. However, the motivation-factor will also make it very difficult to actually identify what gains are due to the technology itself. What happens in a year or two when the developer is not that motivated by the new technology anymore? Will the Ruby productivity gain still be the same?

The motivation and "drive" towards Ruby is also a very interesting observation compared to the amount of experience the developers have. As noted in the evaluation of the book [4] I find it fascinating that people having so many years of experience using J2EE/.NET embrace this new technology in such a way. In my opinion this is a good signal that the Ruby technology offers something to the developers that they have not seen before.

When is comes to evaluating the interview process there are several points worth noting. First of all this form for interview is almost more like a survey than an interview, as it was only based on a questionnaire. Follow-up questions are difficult to make when the process is done via email. The quality of the interviews would probably be higher if they had been conducted face to face. It is also interesting to note that the master thesis on polyglot programming [1] uses much the same process (email interviews with participants on Ruby projects in BEKK), so it is important not to make the same evidence count twice.

Another point is the order of the questions asked. Advantages are asked about before disadvantages, maybe a better approach would be to give half the interviewed developers the questions in a different order? The questions were also, as one participant noted, not very consistent in asking about the language Ruby and the technology platform J2EE/.NET. It should have been made much more clear if we want to study the programming languages Ruby, Java, C# or the technical platforms of Ruby on Rails, J2EE and .NET. The way it was conducted I tend to mix everything together.

A last point to note about the interview process is the selection of recipients of the questionnaire. The recipients were specifically chosen from a list of people known to have experience with Ruby in BEKK. Is it possible that BEKK, in choosing project participants for the Ruby projects, actually selected the developers known to have a preference for the Ruby technology? If so, the answers used in this report are mostly given by people having a preference for Ruby and as such probably having a bias towards wanting Ruby to look good. Much of the same critique as was given to the book [4] in section 3.1 might be valid also here. This is a difficult area, as interviewing people not having experience with Ruby would not be that interesting.

4 Synthesis

I decided on this research question quite early, mostly because of my own motivation and interest in the area. After doing a few literature searches it soon became clear that this is not a widely studied area yet, which can be seen in the somewhat "exotic" choice of referenced articles/books. In a course focusing on empiric evidence it would probably be smarter of me to find a question with more high-quality studies available. I still decided to move on with the question and see what I could find out.

The book [4] does not provide much evidence for when to choose Ruby as a programming language. Due to the authors experience, and the experience of the people interviewed in the book, the claims about increased productivity when using Ruby are absolutely worth looking more closely at. The interesting issue for me is "when" it offers this increased productivity, and that was not very well answered in the book. It does suggest that using the Rails framework with Ruby is one of the major factors when looking at productivity, something that seems to be confirmed in both the first master thesis [1] and the interviews (appendix B). The book gives some interesting ideas to how an organization could evaluate Ruby. As such I am glad I read the book and it surely provided me with good inputs when designing my own research study.

The master thesis about polyglot programming [1] does also point to increased productivity when using Ruby on Rails for developing web-applications. This is also verified by my own interviews from many of the same sources. It also suggests that using Ruby for testing does provide some useful frameworks not found in Java, but the case-study looking at that seems a bit low on evidence because of very little information given.

The second master thesis comparing Ruby on Rails with J2EE/AndroMDA environments [3] also suggests that the productivity is increased using Ruby compared with Java in a web-development scenario. The scenarios used seem quite biased though, and calling the evidence a case-study is also questionable as it was done on a small controlled student-project and not studying a real development project. I do not feel it is possible to generalize anything on the cases presented in this study.

My own interviews did turn up with some interesting information. Using Ruby seems to motivate the developers in a different fashion than Java/J2EE does, and list of important differences between Ruby and Java development was discovered. It also seems clear that the Rails framework offers some very interesting features when it comes to productivity, as long as very complex integration problems are not introduced.

My conclusion is that Ruby seems very efficient when used in combination with specific frameworks developed to solve specific tasks, and the project to be solved does not differ too far from these tasks. As an example Ruby seems to have a positive effect on the efficiency in developing web applications with much database integration, and little other integration, using the Rails framework. Especially when taking into consideration the list of factors found in the interview results (section 3.4.1), making sure these are covered. With applications having a large degree of integration, typically web portals or similar, more research needs to be conducted before answering the question. Ruby also seems to provide interesting capabilities when it comes to creating suites of automated tests and general scripting.

5 Suggested Research Study

As this is a complex topic with many possibly related variables (frameworks, culture, people know-how, motivation, requirements, organizational differences etc.) I would like to design a **case-study** to research it further. Separating the use of Ruby from the context of the project it is used in would be very difficult, and thus making the design of an experiment hard. A case-study is also useful in verifying a theory with reality, and in this case that would be a good fit (ie. "Is Ruby as productive as much theory and blog-hype claims?").

My case-study will be a descriptive single case, or possibly multiple case if more than one project can be found to study. My plan for conducting the case-study is detailed out in the following list:

1. **Clearly define research questions** - I would like to focus on the following research questions:
   - For what kind of tasks is Ruby a good choice of language?
   - What is the effect on developer motivation when using Ruby?
   - What is the rate of errors and how early in the project are these errors discovered?
   - What is the importance of expert knowledge/"best practice" when using Ruby?
   - What is the importance of automated testing and TDD?
   - How well must the given problem fit with the intended solution areas of the framework(s) used before loosing the productivity gain?
2. **Define the unit of analysis** - The unit of analysis will be the project (or projects if multiple cases are used).
3. **Prepare data collection** - Since a case-study typically generates a lot of data it is very important to create a study database to keep track of all data collected. A chain of evidence should be maintained, typically using references actively to visualize all conclusions made. Multiple sources of evidence should be collected:
   - Documentation - All available documentation should be collected. This includes contracts, requirements and test documents.
   - Interviews - Extensive use of interviews to collect data. I will not give the full list of questions here, but typically something like I did for this report (appendix A). Triangulation is important and can be achieved by rotating who is interviewed, covering all roles in the project (developer, manager, customer) and by whom the interview is conducted.
   - Observations - Being able to observe how the developers work, how meetings are conducted and how communication between client, project management and team members works is important.
   - Source code samples - Samples of code could be useful to illustrate examples or for later comparisons with other projects, and even other technologies.

- Issue tracking system - Data collected from issue tracking system would be valuable in order to evaluate rate of errors and other data related to bugs (time to fix, time from development to bug report etc.).

4. **Prepare how to analyze the data collected** - When analyzing the data it is important to regularly verify relevance to the research questions. All data must be looked at and will be equally important; if inconsistencies between various sources are found these must be investigated further. It is also important to be aware of both researcher and participant bias, as both might be influenced by each other, and project members might "defend" choices made or bad results discovered.

5. **Define criteria for interpreting the findings** - It is also important to define how the findings should be interpreted. All generalizations should be carefully checked, and evidence checked for correlation with other evidence. In a case-study many factors, or a combination of factors, might be the cause of a specific finding or result.

This plan will be used to create a case-study protocol giving an overview of the study, a list of procedures to follow, the research questions and a report guide. The case-study protocol should ideally be tested on a pilot project to check the validity (ie. test the interview questions, see if gathered artifacts are sufficient ++).

The project(s) chosen to follow in the case-study would ideally be a project implementing parts of the functionality in both Ruby and another language. The book used in this report suggests a "race scenario" (pages 71-73 [4]) describing this approach. A prototype or other kind of pilot project could be built by two parallel teams, one using Ruby and the other using .NET or J2EE. Performing the case-study on these projects would potentially reveal much data of interest when trying to build up some more empiric evidence comparing the technologies.

References

[1] Hans-Christian Fjeldberg, *Polyglot Programming* (2008). Available from: http://theuntitledblog.com/wp-content/uploads/2008/08/polyglot_programming-a_business_perspective.pdf

[2] Ruby webpage. Downloaded 03.12.08 from: http://www.ruby-lang.org/en/about/

[3] Antonios Protopsaltou, *Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering* (2007). Available from: http://hdl.handle.net/2077/4647

[4] Bruce Tate, *From Java to Ruby - Things Every Manager Should Know*. The Pragmatic Programmers, 1st Edition, 2006. ISBN: 0-9766940-9-3

A Appendix - Email questionnaire

BEKK Ruby Survey
==========
1. Din bakgrunn

1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

1.2 Hvor mange års erfaring har du med Ruby utvikling?

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)

2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i BEKK:

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

3. Ruby "Lessons Learned"

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

4. Metode/fremgangsmåte

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

5. Anbefaling/fremtid

5.1 Hvilken type prosjekter bør vi bruke Ruby i BEKK?

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

6. Andre kommentarer/innspill

B Appendix - Email interview transcripts

B.1 Thor Marius Henrichsen

1. Din bakgrunn
————————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

Ca. 3 års erfaring med J2EE

1.2 Hvor mange års erfaring har du med Ruby utvikling?

Ca, 1 år

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
————————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

- Reimplementering av nytt web løsning for lån/leasing. Systemet var en B2B webløsning som kalkulerte månedlige avdrag og tok imot lån- og leasingsøknader. Scoring og dokumentproduksjon ble gjort i eksterne tjenester. Løsningen fungerte også som en kommunikasjonskanal mellom leverandør og brukere i den forstand at de fikk tilbakemelding på søknader og mulighet til å endre disse. Innebefattet også brukeradministrasjon.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

- Internasjonalt finansieringsselskap (SGFinans)

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

- MRI 1.8.6 utvikling og test (pga. kort startup)
- JRuby 1.1.4 i produksjonsmiljø (pga deployment og eksisterende infrastruktur)
- Rails 2.1
- RSpec
- Cucumber
- RubyFIT
- Watir
- Glassfish app. server
- MS SQL Server 2000
- Netbeans 6.1

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

- Ca. 24 månedsverk vil jeg tippe. Antall utviklere varierte fra 2 - 6 i løpet av en 9 mnd periode.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

- Kunden er kjempefornøyd. Undersøkelser foretatt av kunden viser at de fleste brukerene er mye mer fornøyd med den nye løsningen enn den gamle.

3. Ruby "Lessons Learned"

———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

- Raskere utvikling
- Lettere å jobbe ordentlig testdrevet
- Lettere å få til hyppige releasser
- Lettere å gjøre gjennomgripende endringer i applikasjonen pga. funksjonalitet som monekypatching
- Mye mindre kode. Den opprinnelige løsningen hadde 100.000 LOC. Ny løsning hadde rundt 3.500 LOC.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

- Deployment, monitorering, unicode support, trådhåndtering, minnehåndtering er bedre håndterit i Java enn Ruby. Dette løser forsåvidt JRuby :-)
- Færre begrensninger i språk og lovlige konstruksjoner gir større muligherter. Dette kan misbrukes slik at koden blir uforståelig.
- Det er et mye større behov for fullstendig testdekning, dette kan senke utvikingshastigheten

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

- Vanskelig å disiplinere seg til ikke å ta snarveier.
- Opplæring av nye ressurser.
- Fokus på test kan føre til overtesting (tar tid å kjøre)

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

- Krever mer disiplin enn f.eks. Java fordi mulighetene til å skrive uforståelig kode er større.
- Krever kunnskap om hvordan man bør skrive tester som lar seg vedlikeholde, hvis ikke vil fort tide det tar å vedlikeholde tester overstige utvikling av produksjonsskode.
- Krever i større grad at man jobber sammen fysisk.

4. Metode/fremgangsmåte

———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

- Språket i seg selv stiller ingen krav til valg av metode, men ruby og smidig går godt sammen av to årsaker:
1. Testing er lett og gøy
2. Produktiviteten man oppnår gjør at det er lett å levere hyppig

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

- Andre prosjekter har vært smidige på lissom, dette prosjektet var gjennomført smidig. RoR var mye av årsaken til dette.
- Produktiviteten var mye høyere i begynnelsen av prosjektet enn i andre prosjekter.
- Dette prosjektet var virkelig testdrevet.

5. Anbefaling/fremtid

———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

- I eksisterende Java prosjekter bør ruby testrammeverk brukes i sammen med JRuby for å effektivisere testing.
- I databasedrevne web applikasjoner gir Rails økt produktivitet sammenlignet med Java.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

- Ruby on Rails. Fordi det lar meg uttrykke intensjonen med programmet jeg skriver uten å måtte hensynta allverdens boilerplate stuff for å tilfredsstille kompilator/rammeverk etc.

B.2 Trond Arve Wasskog

1. Din bakgrunn
———————

1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

13

1.2 Hvor mange års erfaring har du med Ruby utvikling?

0

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————

2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Saksbehandling og registrering.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Stor offentlig etat.

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rspec, Watir

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

10-20 utviklere over 2 år

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Automatisert funksjonell testing fungerte over forventning. Krever fortløpende forvaltning og forbedring.

3. Ruby "Lessons Learned"
———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Lett å skrive og forvalte, også for testressurser.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Ingen i denne sammenhengen.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Bleeding edge => bugs

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Vanlig kompetansebygging og avlæring av gamle vaner med statisk typing.

4. Metode/fremgangsmåte
———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Ingen

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

N/A

5. Anbefaling/fremtid
———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Avhenig av kunde og modenhet.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Cobol on nails J

B.3 Fredrik Hansen

1. Din bakgrunn
———————

1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

7 År

1.2 Hvor mange års erfaring har du med Ruby utvikling?

0 År

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————

2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Omskriving av ett B2B Verktøy for innsending av leasingsøknader.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Finanskunde

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Her er jeg ikke helt oppdatert, men Fit, JRuby, Rails, RSpec, CuCumber, Fit,

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

Seks utviklere, men Fire helttidsstillinger(2 jobbet 50%). Budsjettet var noe uklart. Tror kunden regnet med 3 mnd utvikling noe som var basert på løse antydninger.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Prosjektet var en suksess fra forretnignensperspektiv og innenfor forvalterne. Det er blitt gjort en unersøkelse i etterkant med 250 brukere og i generelle trekk sier den at løsningen er bedre enn de andre på markedet. Den er bedre enn den gamle løsningen. Ingen sier løsningen er blitt dårligere.

Fra forvaltningen er vi særs fornøyd med at løsningen er gjennomsyret av tester og at løsnigen nå er både er utrolig mye mindre i form av kodelinjer. (100 000 til 13000ish).

IT avdelingen er ikke helt fornøyd da løsningen ble noe dyrere enn de hadde antatt, men fornøyd med leveransern som sådan

3. Ruby "Lessons Learned"
———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

- Antall kodelinjer er jo noe vi har fokusert på.
- Raskere utvikling.
- Utviklerne like å jobbe med ruby. Ga de noe tilbake.
- Må ikke deployes for å kjøres.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

- Dynamiske språk krever flere tester for å sikre at koden ikke knekker på måter som en compilator ville oppdaget.
- Verktøystøtten er begrenset.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

- Vi sleit med en del bugs, spesiellt mot active records og MSSQL.
- Dette krevde at vi patchet en del av rammeverkene. TIl fordel ble disse raskt lagt til i de offisielle versjonene.
- Vi sliter også litt med å få Glassfish stabilt.
- Refactoring har jeg også inntrykk av at avogtil krevde litt mer enn vanlig. med dårlig verktøystøtte skal man holde tunga rett i munnen ved "Find and Replace" osv

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

- Det krever jo noe mer av utvikleren da han får lite hjelp fra IDE.
- Krever også at man tar seg tid til å skrive tester for å unngå feil i systemene.

4. Metode/fremgangsmåte
———————
4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

TDD er essensielt

Ellers tror jeg ikke den stiller noen spesielle krav fremfor noen andre. Vi liker jo smidig alle i bekk, men føler ikke at jeg kan si at det er ett must.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Vi møtte mange av de samme problemene som deployments til produksjon, oppsett av utviklingsmaskinger osv.

Jeg har ikke jobbet Testdervet før og føler at dette var nytt og nyttig.

5. Anbefaling/fremtid
———————
5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Vi bør vurdere å bruke det i prosjekter av små til mellomstore løsninger som skal ha en ny datamodell i bakgrunn.

Vi brukte mye tid å tilpasse datamodellen til ruby on rails konvensjon for å kunne utnytte RoR egenskapene best.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Noe som utviklerene på prosjektet brenner for. Dette var den store styrken til Ruby. Utviklerene digga det. Og da får man til det meste.

B.4 Frode Nerbråten

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

3.5 år med J2EE

1.2 Hvor mange års erfaring har du med Ruby utvikling?

1.5 år

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

For det meste interne og kompetanserelaterte prosjekter og noe bruk av Ruby til scripting og automatisering på j2ee prosjekt hos kunde.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Internprosjekt i faggruppe

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rails, Capistrano, soap4r, Rspec (litt)

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

Lite, i hovedsak meg selv og 1-2 andre (i faggruppesammenheng)

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Vanskelig å komme med noe konkret her

3. Ruby "Lessons Learned"
———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Større uttrykkskraft. Man gjør mer med mindre kode.

Lettlest kode.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Dårligere IDE-støtte. Større refactoring er vanskeligere uten å brekke noe. Tester er viktig.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Ting tar lengre tid når man ikke har så lang erfaring med et språk (gjelder forsåvidt alle språk).

Dynamiske språk tillater at man kan gjøre feil som kan være vanskelige å forstå.

Man får ingen forvarsler på feil man har innført som kompilatoren ville funnet i f.eks. Java.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

At de er disiplinert i forhold til testing av kode. Når kodebasen vokser, og man ikke lenger kan ha detaljoversikt over alle konsekvenser, er man nødt til å ha tester å støtte seg på.

4. Metode/fremgangsmåte
———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Smidig utvikling er selvsagt en fordel, men det er det uansett valg av språk.

Med høy uttrykkskraft (som i Ruby), kan man levere mer funksjonalitet på samme tid og dermed kjøre korte iterasjoner og få mer kontinuerlig tilbakemelding.

Testdreven utvikling er meget viktig.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

5. Anbefaling/fremtid
———————
5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Særlig til webutvikling og prosjekter som ikke har ekstreme krav til prosesseringshastighet.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Jeg kunne tenkte meg å jobbe med Ruby og gjerne Ruby on Rails

B.5 Øystein Ellingbø

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

java: 1 år fulltid + brukt under studier (5 år)

.net: 1 år under studier

1.2 Hvor mange års erfaring har du med Ruby utvikling?

1 år

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Ekstranettløsning for behandling av leasingsøknader. Løsningen erstatter en gammel javaløsning. Søknader som sendes inn via ekstranettet skal behandles videre i en gammel javaapplikasjon. Integrasjon er på databasenivå.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

SG Finans (evt "stort leasingselskap" hvis det skal brukes eksternt)

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rails, Rspec, Cucumer, Fit

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

ca 4 personer i 6 mnd

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Tid/budsjett: Vi brukte lengere tid enn først antatt. Grunnen til dette var todelt. Vi hadde

estimert tid for å gjøre en mye større oppgave og estimatet for det vi endte med å skulle gjøre ble gitt på sparket på et møte. Det var ment som en mageføelse, men ble fort det som var gjeldende. Omfanget var også større enn vi antok, da avhengigheten til et javasystem og ms sql database var mer kompliserte enn antattt.

Resultat: Kunden er veldig fornøyd med resultatet og de får mye positive tilbakemeldinger fra brukerene. Det har vært forståelse for at estimatet på tid var feil og de har synes de har fått mye funksjonalitet fort og at vi har rettet bugs og gjort småfiks veldig fort. Prosjektet går over i forvaltning nå på fredag og det er bare et par små kosmetiske "feil" igjen.

3. Ruby "Lessons Learned"

———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

- Mindre kode. Prosjektet var en omskriving av eksisterende javasystem. Javasystemet var 50000 linjer kode duplisert og tilpasset for å brukes i to land, dvs ca 100 000 loc (uten noe tester). Ny løsning er i underkant av 4000 loc, med 11000 loc test.
- Veldig lite konfigurasjon og xml-helvete.
- Raskt å jobbe med
- Gøy :)

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

- verktøystøtte (IDE, autocompletion, refactoring)
- er mer avhengig av gode tester (potensielt en fordel hvis det fører til at man lager bedre tester)

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

- databasedrivere for ms sql. ms sql er ikke den hotteste databasen å bruke i rubymiljøet og er derfor ikke så godt testet.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

- Gode tester er mer viktig i ruby enn i java, fordi kompilator, ide og språket i seg selv fanger opp mer av småfeil. Siden det er dårlig med refactoringsstøtte i ruby er man også avhengig av å føle seg sikker på testene sine for å tørre å gjøre refactoring.

4. Metode/fremgangsmåte

———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

- TDD/BDD er et must.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

- Mye større fokus på tester. Stor oppmerksomhet hos kunden på hvilket programmeringsspråk vi bruker (alle testbrukere (som opprettes av forretningen) heter ***ruby, tror ikke det hadde brukt ***java om vi hadde skrevet i det)

5. Anbefaling/fremtid

———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Ruby. Jeg liker ruby mye bedre enn java. Syntaksen er, etter min mening, mer naturlig. Jeg får litt vondt i magen når jeg må gjøre noe i java nå, fordi det er så tungvint i forhold til ruby.

B.6 Stefan Landrø

1. Din bakgrunn
———————

1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

10 år

1.2 Hvor mange års erfaring har du med Ruby utvikling?

2 år

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————

2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Integrasjon mot Buypass sikkerhets og id løsning fra en rails applikasjon som skulle brukes som teknologidemonstrator.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Buypass. Leverandør av elektroniske id og betalingsløsninger.

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

JRuby for å få til integrasjon mot eksisterende buypass sikkerhets api. Tomcat i produksjon. Rails.

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

3-4 deltagere i 4 måneder.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Som forventet. Veldig utydelig målsetning i utgangspunktet - var veldig teknologidrevet.

3. Ruby "Lessons Learned"
———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Fryktelig lett å lære seg - vi gikk fra 0 til meget produktive på 4 uker. Komnpakt kode.

Ypperlig testrammeverk. Overlegen produktivitet.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Deployment, ytelse, monitorering. Utbredelse i markedet. Kompetanse.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Remoting med unntak av REST er et issue. Elendig støtte for WS for eksempel, men kan løses med JRuby.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Man må skrive TDD.

4. Metode/fremgangsmåte
———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Må bruke TDD / BDD.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Det var utrolig moro. Man føler en voldsom befrielse i forhold til økt produktivitet.

5. Anbefaling/fremtid
———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Best egnet til mindre prosjekter (inntil 5 personer) da det er veldig krevende i forhold til testing etc. Viktig å holde jernkontroll.

Rails er suverent.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Gjerne rails. Det er gøy å føle seg produktiv igjen.

B.7 Ole Christian Rynning

1. Din bakgrunn
———————

1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

4 (proffesjonelt) - 8 år inkl. studier

1.2 Hvor mange års erfaring har du med Ruby utvikling?

3.5 (proffesjonelt) - 5 år inkl. studier

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)

————————

- Oslo Politiforening
+ Medlemsregister og administrasjon
+ Hyttebooking
+ Medlemstilbud
+ Rails, testunit
- Warren Wicklund økonomitorget
+ Prototype i Ruby, implementasjon i Java
+ Rails
- Øya-festivalen, Slottsfjell-festivalen, BY:LARM, Alarm, og mange andre i musikkbransjen
+ Frilansutvikler for Manual Design AS
+ Backstage CMS, et bransje-CMS
+ Rails
- BY:LARM
+ Skrev seminar og funksjonærutvidelsesmodul for Backstage.
+ Ruby + ERB
- Nordlys (bistand)
+ Bistod Invenia i utvikling av http://kart.nordlys.no/eiendom - en google maps mashup for annonser
+ Rails + RSpec + google maps
- En rekke internprosjekter for Primetime (kontrollpanel, kunderegister, automatisering, monitorering, etc)
+ I primetime automatiserte jeg drift av en rekke servere basert på programvare utviklet i Ruby
+ Ruby, SNMP, Rails, etc
- Juristopia v1
+ Topic maps implementasjon under/over wiki
+ Rails, RSpec, ...
- Juristopia v2
+ CouchRest, Sinatra, Flex, egen topic maps implementasjon (ikke ferdig)

2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

NAV Rettskildene - Import, scrubbing, XML-ifisering og konvertering av flere tusen dokumenter i ulike format på NAV
SG Finans - teknisk innsalg (sammen med Thor Marius)
Tillsammans - Internt

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Offentlig, finans, internt, musikk, media

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rails, RSpec, Cucumber, Test::Unit, Merb, Sinatra, en hel rekke gems og egenutviklede .

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

1-3 utviklere på alle

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Over forventning, fornøyde kunder. Et feilet prosjekt delvis fordi jeg sluttet i Primetime, delvis fordi kunden var lite responsiv.

3. Ruby "Lessons Learned"
———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Dynamisk, naturlig, lite bloat, mindre fragmentering, hyppigere utviklingshastighet, mindre duplisering, dyktigere utviklere**, lett å angripe alt fra biblioteker og rammeverk til nye kodebaser.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

IDE-støtte har vært ymse. En del merkelige feil med testverktøy og appservere. Noen biblioteker ganske premature.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Å få godkjent bruken...

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Jeg tror personlig man må ha litt mer forståelse for programmering. Man må ha litt hackers-and-painters-mentalitet og være litt kreativ.

4. Metode/fremgangsmåte
———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Man må definitivt kjøre korte iterasjoner - utviklingshastighet er mye større og man kan raskt jobbe seg unna målet om man ikke får interasksjon med kunder.

Testing er særdeles nødvendig når man samarbeider med andre. Test-dreven utvikling er et must når man jobber med mindre erfarne utviklere.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Langt mer interessant teknologi, får litt utløp for kreativiteten, klarer å holde fokus og effektivitet oppe mye lengre.

5. Anbefaling/fremtid
———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Alle

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Ruby, distribuert arkitektur og databaser, enorme skaleringsbehov. Mest fordi Java er uinspirerende, bloated, tregt, gammeldags og lages av utrolig mange dårlige kodere.

B.8 Kristoffer Dyrkorn

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

10 år Java, 5 av disse J2EE.

1.2 Hvor mange års erfaring har du med Ruby utvikling?

1 måned

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Prototype for søk (parsing og indeksering av weblogger)

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Internt (hobbyprosjekt/kom1000)

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Ingen

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

Kun kom1000-tid, 1 deltager.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

På forventning. Selve kodingen gikk bra, men litt vanskelig å integrere mot 3djeparts biblioteker (mangelfull støtte for Ruby som integrasjonsspråk, Java var langt bedre støttet og gjorde at jeg etter hvert gikk over til JRuby)

3. Ruby "Lessons Learned"
———————
3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Kort, effektiv kode. Scriptingmiljø gjør write-deploy-run-fix - syklusen LANGT kortere.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Kort kode - mye nytt syntaksmessig og strukturmessig i forhold til Java (block/proc/yield, etc). Ytelse.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Hvordan unngå å skrive Javakode i Ruby (utnytte de nye mulighetene som språket gir).

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Genuin interesse for å utnytte språket på dets egne premisser

4. Metode/fremgangsmåte
———————
4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Har lite erfaringer - men pga dynamisk typing vil TDD være en stor fordel

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Har for lite erfaring

5. Anbefaling/fremtid
———————
5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Små webløsninger (les: Rails), og som hjelpeverktøy i diverse scriptsammenhenger (JRuby bør da brukes for å unngå installasjonsdiskusjoner og problemer/diskusjon med å innføre en ny runtime.)

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Gjerne Ruby. Eller Java. Har ikke sterke meninger om dette - fordelen med Java er at jeg kjenner det miljøet såpass mye bedre at jeg er mer effektiv der. Og så har jeg en følelse av at det er mer gjennomtestet og utprøvd enn Ruby.

6. Andre kommentarer/innspill
———————

Go Ruby, men don't believe (all) the hype.

B.9 Eivind Uggedal

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

Omtrent 0 proffessionell erfaring

1.2 Hvor mange års erfaring har du med Ruby utvikling?

3,5 års erfaring på hobby/open source nivå

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

cv-base

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Internprosjekt, Bekk

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Jruby, Rails, Rspec, Cucumber, Webrat, Prawn

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

2 tekkere og en deltids GSP-er, samt noe design. Løsningen ble levert på 4 uker.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

Etter tilbakemeldinger fra kunde tror jeg prosjektet oppnådde resultater over forventning

3. Ruby "Lessons Learned"
————————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Smidighet, endringsdyktighet, mindre kode, lettere å teste, lettere å lese hvis konvesjoner følges og ikke for mye metaprogramering brukes (Rails har mye magi vs Merb sin kode som er lettere å forstå og endre siden den er mer ekpslisit)

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Mange vil si mangel på compile-time typesjekking. Skriver man de riktige testene har jeg ikke hatt problemer med dette.
Dårligere verktøystøtte med mindre autocomplete, refactoring muligheter, osv. Siden Ruby er mye mer dynamisk vil nok aldri slike verktøy bli like bra for språk som Java. Har vel heller ikke lidt mye på grunn av dette siden Ruby er et temmelig konsist språk og med vim og ctags kommer man langt.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Importering av annen kode er grisete i Ruby. Man importerer kun filer, og disse kan inneholde alt mellom himmel og jord. Python sin løsning ved at man importerer på modulnivå klikker bedre for meg (from somemodule.utils import get_object_or_404). Rubys dynamiske natur kan også være smertefult til tider. Bare ved å importere Rails har plutselig Object en million metoder. Navnkolisjon er temmelig vanlig.

I tillegg har jeg opplevd problemer med å utvikle/deploye Ruby på plattformer som ikke er POSIX kompatible. JRuby løser dette til en viss grad.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Man må opparbeide et sett med konvesjoner som alle på et team burde følge. Siden Ruby slekter på Perls filisofi om at en ting kan gjøres på mange måter må tungen holdes rett i munnen. Må igjen trekke frem Python som et moteksempel med sin filisofi om at " There should be one – and preferably only one – obvious way to do it".

4. Metode/fremgangsmåte
————————
4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

TDD/BDD blir viktigere.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Har ikke nok erfaring til å svare på dette.

5. Anbefaling/fremtid
————————
5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

Internprosjekter med korte leveransefrister.
Testing og automatisering av alle typer prosjekter.
Vi burde være varsome med å bruke Rails og dets sterke konvesjoner i prosjekter som krever mye integrasjon med legacy systemer.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Jeg har lyst å jobbe med prosjekter hvor Ruby/Rails har blitt forespeilt som en god kandidat men hvor man vurderer andre plattformer og rammeverk som muligens løser problemet bedre som for eksempel (J)Ruby/Merb, (J)Python/Django.

B.10 Nils-Helge Garli

1. Din bakgrunn
————————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

> Ca 8 år

1.2 Hvor mange års erfaring har du med Ruby utvikling?

> 1 år

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
————————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

> Nettbasert løsning for leasing- og lånesøknader.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

> Finanskunde.

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

> JRuby, Rails, RSpec, Watir, GlassFish +++

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

> ca 5 fulltidsressurser til sammen.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

> Over forventning. Kunden er meget fornøyd og mengden kode ble betydelig redusert!

3. Ruby "Lessons Learned"
————————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

> Mer "gøy"! Å utforske Ruby er nesten som å være en unge i en godteributikk. Man får gjort mer med mindre kode. Jeg tror det å programmere i Ruby har gjort meg til en bedre og mer kritisk programmerer.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

> Pga den dynamiske naturen så er det en del feil som kan oppstå som hadde blitt fanget compile time i Java. Det blir fortsatt sett på som et "leke-programmeringsspråk" av de fleste av våre kunder. Samtidig er det foreløpig mye mindre kompetanse å oppdrive, selv om de fleste Java/.Net-programmerer ganske fort vil kunne lære seg Ruby.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

> Pga Rubys "mangler" ifht til Java gjør at man må være mer disiplinert. Man kan rett og slett ikke hoppe over eller skrive dårlige tester pga det "smeller senere" enn i Java.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

> Først og fremst disiplin og objektorientert forståelse.

4. Metode/fremgangsmåte
————————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

> Testing, testing og testing. I så måte passer det best sammen med en smidig prosess.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

> The "Fun-Factor", og at avstanden mellom prototype og ferdig produkt er så liten siden prototypen er "et steg på veien" til den ferdige funksjonaliteten, noe som igjen gjør det veldig enkelt å hele tiden ha noe å demonstrere underveis.

5. Anbefaling/fremtid
————————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

> Det er fryktelig vanskelig å si, men Ruby bør definitivt være et forslag til løsning i de fleste tilbudene våre, iallefall der det er snakk om skreddersøm.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

> Jeg jobber gjerne mer med Ruby, men det viktigste er at prosjektet i seg selv er spennende og at man får spennende utfordringer. Og at man ikke minst som utvikler har et effektivt utviklingsmiljø.

6. Andre kommentarer/innspill
———————

> Jeg tror fortsatt vår største utfordring er å få kundene til å forstå at dette ikke bare er "hobby-teknologi" men en seriøs utfordrer til den vanlige "enterprise"-teknologien man bruker. Nå har vi et par Ruby-prosjekter, og jeg synes vi må få mer blest og reklame ut av disse.

B.11 Hans-Christian Fjeldberg

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

0. Kom rett fra universitet. Ble undervist i vanlig Java, men ikke i EE. Brukte Hibernate og Spring på sommerjobben, men det var jo bare 2 måneder.

1.2 Hvor mange års erfaring har du med Ruby utvikling?

0. Kom rett fra universitet. Har sittet på cv-base prosjektet som varte i en måned. Ble undervist i Ruby i New Zealand, og har brukt det som midt foretrukne språk siden (2006). Men har altså bare brukt det 1 måned i profesjonell sammenheng.

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

Jobbet på en intern cv-base webapplikasjon, for registrering og generering av cver.

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Internprosjekt.

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rails, RSPec, Cucumber, JRuby, Prawn, MSSQL, JQuery.

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

Vi var totalt 4.5. Prosjektleder, 2 programmerere, 1 GSP og en designer ved behov.

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

De tilbakemeldingene vi har fått har vært veldig positive, spesielt var kunden positiv til hastigheten vi oppnådde under prosjektet.

3. Ruby "Lessons Learned"

———————

3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Selv om dette var et nytt prosjekt som ikke baserte seg på gammel kode, hadde vi miljøet oppe å kjøre i løpet av 3 dager. Til sammenligning har vi brukt mellom 3-4 uker på å få opp miljøet på det prosjektet jeg sitter på nå, som er et J2EE prosjekt. Selv om vi støtte på problemer, hadde vi sikkert kommet til å støte på de samme, tilsvarende eller verre problemer ved bruk av et Java eller .NET.

Ruby sine API føler jeg også er mye mer naturlig å lese enn mange av Java sine, noe som gjør behovet for autocomplete mye mindre.

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Manglende støtte for refactoring, som er det eneste jeg savner fra en IDE.

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

Kravet til økt behov for tester er en stor utfordring, spesielt antall tester som må endres ved en refactoring.

Siden _alt_ er lov i Ruby, er det også nødvendig for utviklerne selv å sette retningslinjer for hvordan språket skal brukes. Herunder kommer f.eks. Monkey Patching, kodestandard, bruk av one liners, eksplisitt include og extend. Dette har du selvfølgelig kun kontroll på i din egen kode, hvordan rammeverk gjør dette har du ikke kontroll over. En liten risiko er at Ruby biter seg selv i halen fordi alt er lov, noe som kan gjøre programmerere utilpass.

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

Fordi du ikke har like gode IDEer som i Java/.NET, føler ihvertfall jeg at det kreves mer kunnskap om språket og rammeverkene for å bruke Ruby. Istedenfor å trykke på CTRL+Space, er utvikleren nødt til å slå opp i APIen for å finne ut hvilke metoder som finnes.

4. Metode/fremgangsmåte

———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

Føler ikke språket stiller krav til bruk av metode. Metoden kommer ann på prosjektet, teamet, oppdragsgiver, osv. Deretter velges språk og teknologi. (IMHO)

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

Selv om min erfaring er begrenset, så vil jeg si at farten vi utviklet ny funksjonalitet med i cv-base var betraktelig mye større ved hjelp av Ruby enn det er nå med J2EE. Det hjelper selvfølgelig ikke på nåværende prosjekt at min hjerne tenker i Ruby, og alt jeg gjør i Java sammenligner jeg med hvor lett det hadde vært å gjøre i Ruby :)

5. Anbefaling/fremtid

———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

For å få innpass i bedrifter kan det lønne seg å begynne med små oppgaver, som f.eks. testing, scripting osv. Små utvidelser kan lages i Rails eller Merb og deployes som war ved hjelp av JRuby. Deretter kan hele webløsninger erstattes av Rails og Merb (Merb trenger litt mer dokumentasjon bare, så er det klart til å taes i bruk).

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

Ruby. Som nevnt over så tenker jeg som sagt i Ruby for tiden, og sammenlikner alt jeg gjør med hvordan jeg ville ha gjort det i Ruby.

6. Andre kommentarer/innspill
———————

Diskuterer som sagt noen av disse problemstillingene i min egen hovedoppgave, som jeg linket til på forumet.

B.12 Aslak Hellesøy

1. Din bakgrunn
———————
1.1 Hvor mange års erfaring har du med J2EE/.NET utvikling?

6

1.2 Hvor mange års erfaring har du med Ruby utvikling?

6

2. Om prosjektet (hvis du har jobbet på flere - velg ett eller beskriv alle i punktlister)
———————
2.1 Beskriv kort hva slags prosjekt(er) du har jobbet med Ruby i Bekk:

- BuyPass - pilotprosjekt for betalingsløsning. Bruk av JRuby for integrasjon mot legacy system.
- SG Finans - Ny selbetjeningsløsning for lån- og leasingsøknader
- BEKK - Emprest - intern tjeneste for ansatt- og prosjektinformasjon
- BEKK - CV base - internt system for å generere og søke i CVer

2.2 Hva slags oppdragsgiver var det (internprosjekt, finanskunde osv.)?

Se over

2.3 Hvilke tekniske rammeverk/løsninger ble benyttet (Rails, RSpec, JRuby osv.)?

Rails, RSpec, JRuby, Cucumber, ZenTest, pluss et titalls Ruby on Rails plugins for autentisering, tagging, pdf generering etc. Si fra hvis du ønsker en full liste.

2.4 Hva slags omfang hadde prosjektet (antall prosjektdeltagere, budsjett)?

- BuyPass - 3 fulltid og 1 deltid over ca 3 måneder. Jon F har budsjettet
- SG Finans - 3 fulltid og 3 deltid over ca 6 måneder. Jon F eller Fredrik Hansen har budsjettet
- CV base - 2 fulltid ca 1 måned

2.5 Hvilket resultat oppnådde prosjektet (over/under forventning - beskriv kort)?

- BuyPass - under forventning. Eksisterende infrastruktur og kompetanse egnet seg dårlig for Ruby on Rails.
- SG Finans - over forventning. De fleste av prosjektdeltakerne var uerfarne og ble raskt produktive. Kunden er svært fornøyd med leveransen. Originalt system (J2EE) hadde 100.000 LOC produksjonskode og 0 LOC tester. Nytt Rails system hadde 4.000 LOC produksjonskode og 10.000 LOC tester.

3. Ruby "Lessons Learned"
———————
3.1 Hvilke fordeler har Ruby sammenlignet med J2EE/.NET?

Det virker litt rart å sammenlikne et språk med en applikasjonsplattform... Mente du Ruby on Rails sammenlignet med J2EE/.NET? evt Ruby sammenlignet med Java/C#?

Jeg prøver å svare likevel.

- Hurtig utvikling
- Morsom utvikling -> mer motiverte utviklere

3.2 Hvilke ulemper har Ruby sammenlignet med J2EE/.NET?

Det virker litt rart å sammenlikne et språk med en applikasjonsplattform. Antar det skulle vært ett av følgende:

3.2.1 Hvilke ulemper har Ruby on Rails sammenlignet med J2EE/.NET?

- Tilgang på kompetente utviklere
- Skepsis i miljøer som har låst seg til en plattform

3.2.2 Hvilke ulemper har Ruby sammenlignet med Java/C#?

- Tilgang på kompetente utviklere
- Skepsis i miljøer som har låst seg til en plattform
- Mangelfull støtte for refactoring og code completion

3.3 Hva er de største utfordringene du har opplevd med bruk av Ruby?

- Noen biblioteker har dårlig dokumentasjon (dette gjelder vel alle plattformer)
- Bruk av Windows som utviklings- og produksjonsplattform
- Plattformuavhengighet - det kan være utfordrende å få enkelte biblioteker til å kjøre på ulike kombinasjoner av MRI/JRuby og OS

3.4 Hva slags krav stiller bruken av Ruby til utviklerne på prosjektet?

- At de ikke har angst for å bruke kommandolinja

- At de praktiserer TDD/BDD. Skriver testene -først-
- At de er åpne for at man kan være mer produktiv selv uten IDE refactoring og code completion
- At de leser relevante blogger
- At de kjenner prosessen for å bidra tilbake til open source miljøet
- At de (helst) har en Mac eller Linux utviklingsmaksin

4. Metode/fremgangsmåte

———————

4.1 Hvilke krav stiller bruken av Ruby til valg av metode?

- Ingen. Det er et programmeringsspråk (?)
- TDD/BDD er likevel anbefalt for å heve kvaliteten på koden.

4.2 Hva er forskjellene fra andre prosjekter du har jobbet på?

5. Anbefaling/fremtid

———————

5.1 Hvilken type prosjekter bør vi bruke Ruby i Bekk?

- Der leveransen innebærer programmering. Det bør vurderes i ethvert tilfelle.
- Rails passer oftest bra til prosjekter der webgrensesnitt og database er sentrale komponenter.

5.2 Hvilken teknologi vil du helst jobbe med i ditt neste prosjekt? Hvorfor?

- En sunn balanse av "bevist" teknologi som jeg kan og ikke kan, fordi jeg har det gøy når jeg lærer. Og jeg leverer best kvalitet når jeg har det gøy.
- Eksempel på "sunn" combo: Flex, Erlang, CouchDB, REST, SOLR, Rails, Java :-)