Pervasive Fun

Dr. Benno Luthiger[*]
ETH Zurich
IT Services

PD Dr. Carola Jungwirth
University of Zurich
Chair of Strategic Management and Business Policy

January 2007

Published in: First Monday, 12, 1 January 2007,
http://www.firstmonday.org/issues/issue12_1/luthiger/.

Abstract

The goal of the study on *Fun and Software Development (FASD)* is to precisely assess the importance that fun has as motivation for software developers to engage in open source projects. A survey carried out both under open source developers and programmers working in Swiss software companies yielded that the fun motive accounts for about 27% to 33% of open source developers' motivation.

Fun is a pervasive feature of software development, not only for open source programmers but in the area of commercial software development too: Open source developers that are paid for their work are observed to be very motivated and prepared for future effort, especially if they enjoy their development time. Furthermore, the fun that the programmers experience functions as a good proxy for their productivity. Therefore, employers that want to enhance the programmers' productivity can safely invest in an environment of fun for developers in their company.

1 Introduction

Open source software is available for anyone to use without legal ramifications or entanglements, because it is legally permitted to be downloaded and used without having any financial obligations to the original provider. The open source movement again and again has successfully produced software of extraordinary quality since the concept of open source development came into existence. In fact, over

[*]Mail: benno.luthiger@id.ethz.ch

time, the emergence of the open source movement has steadily increased its dynamic and now poses a serious challenge to the sellers of commercial software. How is the phenomenon to be explained? Did the *homo oeconomicus* gone mad when he entered the software area?

To answer these questions, we have to ask for the motivations of the actors in the open source area. The scientific research dealing with the open source phenomenon has identified a variety of such motivations and has been able to give a reasonable theoretical justification for them (e.g. Hars and Ou (2001), Lerner and Tirole (2002), Osterloh, Rota, and Kuster (2002), Franck and Jungwirth (2003), Lakhani and Wolf (2003), Hertel, Niedner, and Herrmann (2003)). What's missing so far, concerning the research on open source developer's motivations, is a quantification of the different motivations. We're able to understand the motivations of the contributors to open source projects in theory, but we don't know how important these motivations are in practice. Possibly we have a misleading impression of the open source developers because we assume motivations that are theoretically elegant but insignificant in praxis. Besides the statements about the existence of motivations to engage in open source, we have to come to statements about their relevance in order to get a realistic impression of the actors in the open source area.

This paper aims to take a first step to fill this gap. In the FASD study (*Fun and Software Development*, see Luthiger (2006)) the fun motive is the exclusive concentration. Our ambition is to explain the share of the open source developers' engagement that can be attributed to fun.

Therefore, the main research question of the FASD study is: *How much of the open source developer's engagement can be explained by a model in which fun and spare time enter as independent variables?* In other words, we are trying to determine, assess, and quantify the importance and impact of fun as a motivating force for open source developers.

The assumption behind this research question is that software development is done because it is fun. Open source developers, therefore, program in their spare time, because they consume "fun" with this activity and the developed open source software is a by-product of this activity. This explanation sounds reasonable but is not enough to justify the existence of open source software. The fact that programming is fun may explain that there are persons doing this activity in the first place. However, we have to take into consideration that one can earn money by developing software. If we are dealing with rational software developers, the possibility to earn money is without doubt an additional utility. Consequently, we don't expect anyone, at least no rational actor, to program in his or her spare time anymore, because having fun and earning money is *mutatis mutandis* better then only having fun.

Therefore, if we want to explain the existence of open source software by the fun motive, we have to postulate that having fun while programming is somewhat substitutive to earning money with software development. The open source development has to offer better opportunities to enjoy programming then working in the commercial software area. In the FASD study, we want to verify this hypothesis by asking our second research question: *Do software developers in open source projects enjoy more fun than programmers working under commercial conditions?*

This question can be verified by finding significant differences concerning the experience of fun when comparing open source and commercial software developers; however, an additional question then arises: *Are the characteristics of an open source project, which a software developer is participating in his spare time, constitutive for the difference concerning the experience of fun?*

In the next sections we will answer these three research questions based on the data gathered in the FASD study. The paper is organized as follows. In the following section (section 2) we briefly describe the design of the FASD study and the sample characteristics. We then report the quantitative findings about the importance of fun (section 3). In section 4 we give details about the data gathered in the control group of commercial software developers. In the next section, we compare the experience of fun of both the sample of software developers and the sample of programmers working under commercial conditions (section 5). We conclude the paper with a discussion of the findings of the FASD study (section 6).

2 The FASD study: design and realization

To measure the importance of fun to explain the open source developer's engagement, we need a model that links fun as independent variable with engagement as a dependent variable. With such a model we can adopt the same methodology Hertel et al. (2003) used to determine the main motivational factors for team development in an open source project. Thus, instead of asking how important the fun motive is compared with the other possible motivations for an engagement, the variance of engagements among different open source developers is considered. By inquiring into the accuracy of our model (containing fun as independent variable) in explaining the developers' engagement, the quantitative estimate can be identified.

The model that is being sought can be understood as production function establishing a functional connection between input and output factors. The output factor in this case is the engagement in an open source project whereas the input is the fun the developers have while programming. A customary production function has a concave shape, indicating that an additional unit of the input factor only adds to a sublinear increase of the output. A possibility to model such a decreasing marginal utility is a quadratic function with negative coefficient in the quadratic term.

In addition to "fun" as input factor, it seems reasonable to add the availability of spare time as a second input factor. The phenomenon we want to explain is the free and voluntary engagement of an open source developer. Obviously, such an engagement occurs in their spare time. If an individual developer does not have any spare time, he simply does not have the possibility to engage, independent of the fun he might have while programming. Again, the marginal utility of an additional unit of spare time shall be decreasing, thus, a model is suggested with negative coefficient in the quadratic term. With these considerations, the following production function is arrived at:

(1) $E = c + a_1 * F - a_2 * F^2 + b_1 * T - b_2 * T^2$

where E: voluntary, unpaid engagement
F: fun
T: spare time
$a_1, a_2, b_1, b_2 > 0$

An alternative production function fulfilling the requirement of decreasing marginal utility can be obtained by a logarithmic model:

(2) $E = c + a * \ln F + b * \ln T$

where E: voluntary, unpaid engagement
F: fun
T: spare time
$a, b > 0$

Having the production function, the question arises on how to operationalise the variables in the model. In the FASD study, we used three different possibilities to operationalise the open source developer's engagement. First, we asked the respondents about their readiness for future engagement in open source projects. Such a question has been proposed in the empirical study by Hertel et al. (2003). Second, we ask the open source developers about how much of their spare time they spend for open source projects. As third possibility we tried to determine the engagement by the number of contributed patches and lines of code. Again, this question was influenced by the study by Hertel et al.

To determine the fun while programming we rely on the flow concept developed by Csikszentmihalyi (in Csikszentmihalyi (1975)). Flow is a special form of fun. The flow experience is characterized by the following elements:

- *Concentrating and focusing*: a high degree of concentration on a limited field of attention (a person engaged in the activity will have the opportunity to focus and to delve deeply into it).
- *A loss of the feeling of self-consciousness*: the merging of action and awareness.
- *Distorted sense of time*: the subjective experience of time is altered.
- *Control and a high level of absorption*: a sense of personal control over the situation or activity.
- *Clear goals and immediate feedback*: expectations and rules are discernable and successes and failures in the course of the activity are apparent, so that behavior can be adjusted as needed.

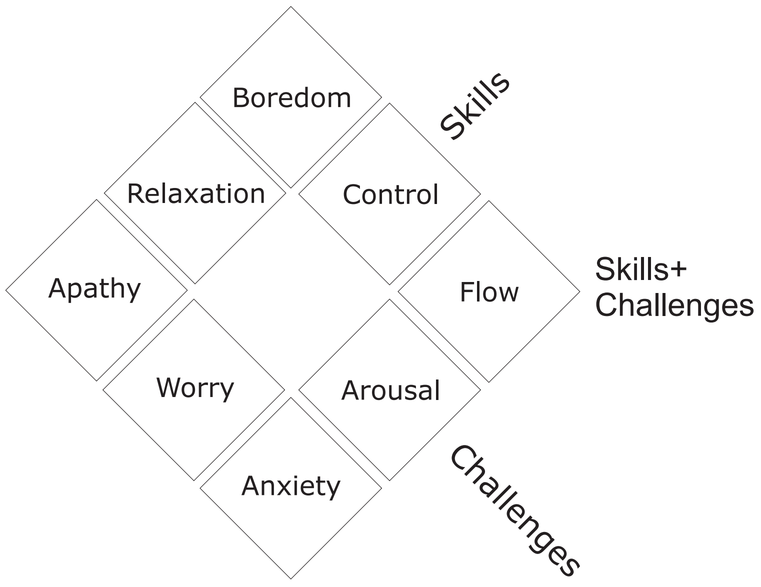

Figure 1: Flow channel segmentation model (from Novak and Hoffman, 1997, p. 11)

- *Flow of actions*: each steps leads fluently to the next as if the events are lead by an inner logic.

For that flow can happen, there are some prerequisites that govern the situation:

- *Attention focusing*: the attention has to be focused on a limited field of stimulus.
- *Balance between ability level and challenge*: The perceived requirements have to be in balance with the person's ability level, whereas both requirements and the person's abilities have to be over average (in the actor's view).

The last point has been expanded to the *flow channel segmentation model* by the flow research (see Figure 1). This model states that the flow experience is the counterpart of apathy and beyond boredom and anxiety (see Csikszentmihalyi (1975)). According to this model, a person falls into apathy if both challenges and abilities are low. If the challenges increase, apathy goes over to worry and anxiety. Having the challenges fixed, apathy may change to relaxation and boredom with increasing skills. Boredom changes to control with raising challenges and eventually goes over to flow. With constant challenges, anxiety is replaced by arousal and goes over to flow with increasing abilities.

The flow concept seems to be specifically prolific in the context of computer work in general and programming in particular. Since Csikszentmihalyi presented his seminal concept the first time, an abundance of studies appeared that measured flow (e.g. Rheinberg, Vollmeyer, and Engeser (2002), Montgomery, Sharafi, and Hedman (2004), Chen (2002), Novak, Hoffman, and Yung (1998)). In the FASD study, we were heavily influenced by the flow questionnaire Remy (2002) developed to measure the flow experience of computer activities.

To calculate the respondents' spare time we asked some additional questions.

In order to achieve the aims of the FASD study, a questionnaire for an online survey was developed. There were two versions of this questionnaire, one addressing open source developers, the other addressing programmers working in commercial software firms. The questionnaires consisted of 53 questions. The first part of the questionnaire was identical for both versions. The purpose of this part was to measure the flow software developers experience during their work.

In the following part, we asked the open source developers about their readiness for future activities in open source projects, about how many patches and modules they have developed so far, and how much of their working hours and spare time respectively they spend on developing open source software. With these questions, the criterion variables were established, i.e. they function as a measure of the developers' commitment.

In a further part of the questionnaire, the open source developers were asked about the reasons why they initially joined an open source project or why they started one themselves. In the concluding part of the questionnaire, we gathered demographic data about the respondents and tried to elicit information about the opportunity cost they have when they work for open source projects in their spare time, e.g. by asking them how much spare time and how many hobbies they have.

In the questionnaire for the developers in commercial software firms, we asked them about their willingness to work overtime and about how many checkins they did in the past few days in order to get an impression of their commitment. In a further part of the survey, these developers were asked about their relation to their employer, i.e. how proud they are of their employer and how well they can develop personally and professionally at their workplace. We further asked them how frequently they feel deadlines, about the project visions behind the software projects they work in and about the project managers' formal authority and professional competence. Again, gathering demographic data concluded this questionnaire.

The questionnaire we used was a standardised questionnaire that contained no open questions. About 80% of the questions were formulated as statements; the respondents then indicated their agreement with the statements from "completely unimportant" to "very important" and "is never the case" to "is always the case" respectively on a six point Likert scale.

By directly comparing the answering behaviour of open source developers and programmers working for commercial firms, it is possible to test the hypothesis that the two groups differ significantly concerning the experience of flow. And by asking the commercial developers about deadlines, project visions and formal authority - all characteristic elements of the commercial software development model which distinguish it from the open source model - the factors responsible for a potential difference between them can be identified.

The open source version of the questionnaire was launched on May 3, 2004. We sent a mail to the mailing lists of the various projects hosted by SourceForge, GNU/Savannah and BerliOS. This questionnaire was open during 53 days until June 25, 2004 and was filled in by 1330 respondents. For the second questionnaire, six software companies were identified in Switzerland ready to collaborate with the FASD study. The questionnaire for the developers working for these companies was open from September 20 until November 10, 2004. 114 software developers filled in this version of the questionnaire.

3 The importance of fun for open source developers

With the data gathered from the open source developers on the one hand and the model 1 on the other hand we are able to calculate the importance of fun as motivational factor to explain the open source phenomenon. However, we have to prepare the data gathered first.

3.1 Engagement and spare time

With the questions *40* to *42*[1] of our questionnaire, we were able to calculate the use of time for open source (Table 1). The contributors spend an average of 12.56 hours per week on OSS activities (median: 8 hours per week). This value is 1.5 hours below the value found out by Lakhani and Wolf (2003, p. 10). 7.31 hours thereof are spent during spare time (58% of the time, median: 4.8 hours) and 5.22 hours during working hours (median: 1.5 hours). This statistical analysis corroborates the impression that the commitment for open source projects mainly happens in the spare time. Yet, the share of development of open source projects during working time amounts significant level of 42%. However, we have to consider that the professional open source developers may be underrepresented in the FASD study. Professional open source projects can afford their own project infrastructure and, therefore, are not dependent on the open source platforms addressed in our study.

Combining the answers of the questions *40* to *42*, the number of spare time the respondents have was able to be calculated.[2] On average, the respondents have 26.7 hours spare time per week (Table 1). The value for the spare time will be used as input for the second independent variable in the model.

Table 2 shows the frequency distribution of the engagement for open source the developers' spend during their spare time. From this table we can infer that more then one third of the developers are paid for the majority of their time developing open source software. For the remaining two thirds of the programmers, the open source activity takes place mainly in their spare time. Lakhani and Wolf found

[1]40: "Please estimate the time you spend for the development of open source software (average hours per week).", 41: "Of the total time spent for the development of open source software, how much in percent is part of your spare time?", 42: "How much (on average, in percent) of your spare time do you spend on activities concerning open source projects?"

[2]I calculated spare time as follows: *spare time* $= q40 * q41/q42$.

Table 1: Time spent for open source (hours per week)

	Mean Value		Standard Deviation	Valid Number
Time spent in general	12.56	(100%)	14.18	1228
Time spent in spare time	7.31	(58%)	8.23	1215
Time spent during working hours	5.22	(42%)	10.82	1215
Hours per week spare time	26.70		24.39	1210

Source: Benno Luthiger, FASD Study, University of Zurich

out a share of 40% paid open source developers (Lakhani and Wolf, 2003, p. 9). However, they did not specify how much of the open source engagement of these developers is spent during working hours.

The frequency distribution (see Figure 2) shows two peaks at the corner points. This distribution suggests a differentiation in *professionals* and *open source hackers*. A professional is designated as an open source developer producing less then 10% of his open source engagement in his spare time. Such a programmer engages virtually exclusively during working hours. In contrast to this kind of developer, a programmer whose open source engagement takes place for more then 90% in the spare time, matches the image we have of an *open source hacker*. The share of professionals in our sample amounts to around 12% whereas the open source hackers contribute with about 40% to our sample.

Do these two types differ concerning their engagement for open source? The statistical analysis shows that indeed the professionals spend with 14.6 hours per week a significantly higher amount then the hackers with 9.7 hours per week (see Table 3).

With the questions 29 to 31[3] we queried different aspects of the readiness for future engagement. A factor analysis showed that these three items can be reduced to one underlying factor.[4] Therefore, we will work with an index "readiness for future effort" built of the values of these three items in the further evaluation.

How is the professionals' readiness compared with the hackers? Not very surprisingly, in two of the three items and also in the whole index, the hackers show significantly higher values for the readiness then the professionals (see Table 4). This shows that the hackers are more enthusiastic about their open source activities then their professional counterparts.

The time spent for open source represents the actual and the readiness for fu-

[3]29: "I'm looking forward to further development activities for open source software.", 30: "I'm prepared to increase my future commitment in the development of open source software.", 31: "With one more hour in the day, I would program open source software."

[4]The computed factor can explain 67% of the variance of the original items. The value of Cronbach's α is 0.74.

Table 2: Share of time spent in spare time

		Number	Percent	Valid Percent	Cumulated Percent
Valid	0%-10%	153	11.5%	11.9%	11.9%
	11%-20%	83	6.2%	6.5%	18.4%
	21%-30%	71	5.3%	5.5%	23.9%
	31%-40%	48	3.6%	3.7%	27.6%
	41%-50%	74	5.6%	5.8%	33.4%
	51%-60%	67	5.0%	5.2%	38.6%
	61%-70%	49	3.7%	3.8%	42.4%
	71%-80%	107	8.0%	8.3%	50.7%
	81%-90%	114	8.6%	8.9%	59.6%
	91%-100%	518	38.9%	40.4%	100.0%
	Total	1284	96.5%	100%	
Missing		46	3.5%		
Total		1330	100%		

Source: Benno Luthiger, FASD Study, University of Zurich

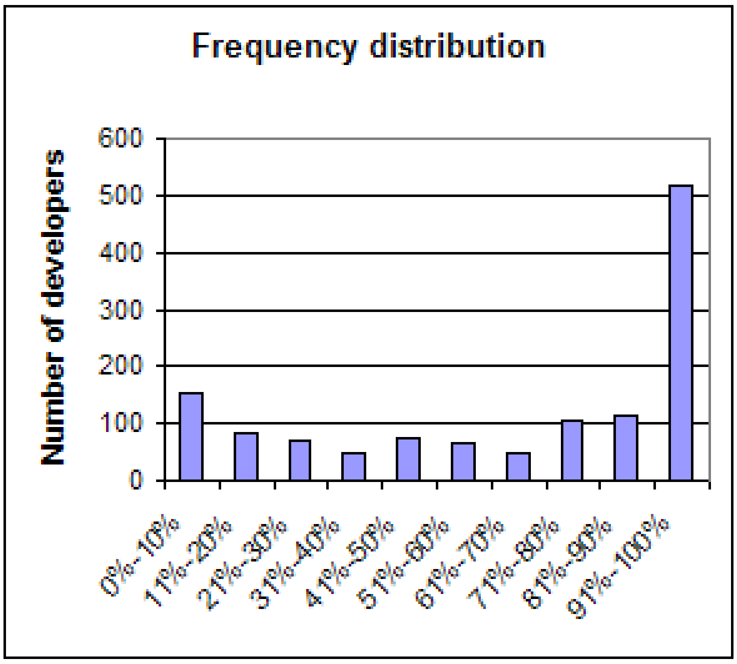

Figure 2: Share of time spent in spare time

Table 3: Time spent per programmer type

	Mean Value	Standard Deviation	Valid Number
Professional	14.60	24.17	131
Hacker	9.69	9.54	496

Source: Benno Luthiger, FASD Study, University of Zurich

Table 4: Readiness per programmer type

	Value		
	Mean	Std.error	Number
29: I'm looking forward to further development activities for open source software.			
Professional	4.66	0.10	145
Hacker	5.08	0.04	503
Total	4.98***	0.04	648
30: I'm prepared to increase my future commitment in the development of open source software.			
Professional	4.27	0.11	145
Hacker	4.41	0.05	499
Total	4.38	0.05	644
31: With one more hour in the day, I would program open source software.			
Professional	3.60	0.13	146
Hacker	4.23	0.06	487
Total	4.08***	0.05	633
Index "Readiness for future effort"			
Professional	4.16	0.10	148
Hacker	4.58	0.04	505
Total	4.49***	0.04	653

Remark: *** Difference significant on 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

Table 5: Output

	Output per year		
	Mean	**Std.error**	**Number**
Number of patches per year	28.46	4.55	1187
Number of classes/modules/files per year	73.46	8.68	1193

Source: Benno Luthiger, FASD Study, University of Zurich

Table 6: Productivity

	Productivity		
	Mean	**Std.error**	**Number**
Productivity concerning patches	2.55	0.35	1168
Productivity concerning classes/modules/files	7.88	0.99	1172

Source: Benno Luthiger, FASD Study, University of Zurich

ture effort stands for the forthcoming type of engagement. Correspondingly, with output and productivity we find out aspects of engagement lying in the past. The starting point for the calculation of output and productivity are the questions about the number of patches (question 38) and classes/modules/files (question 39) produced so far. This part of the FASD questionnaire was influenced by the Linux study by Hertel et al. (2003), where the respondents have been asked about the number of patches (among other things). However, the question arises whether this measure (number of patches or number of classes/modules/files) is appropriate in the context of the FASD study too. We will discuss this issue below.

To calculate the output, we divided the values of the questions *38* and *39* by the number of years the person is active in the open source area (Table 5). To get the productivity, we set this calculated output in relation with the time spent per week for open source (question *40*) (see Table 6).

3.2 Flow of open source developers

We operationalized the dependent variable "fun" in our model using the flow concept developed by Csikszentmihalyi. By means of a factor analysis, we tried to identify the decisive factors which underlie the 28 questions of the FASD questionnaire concerning the flow experience. Both KMO- and Bartlett's test proved that the data is suitable for factor analysis. After some investigations, we decided to work with only one factor *flow/fun*. This single factor accounts for 26.8% of the original variance. The reliability analysis using Cronbach's α to asses the quality of the scale resulting from the factor analysis shows a good value for the extracted factor (0.86).

Table 7 shows the extracted factor's communalities and factor loadings. Factor loadings vary between 0.73 and -0.02. Not surprisingly, the item (*27: I completely concentrate on my programming work.*) showing the highest factor loading has the highest communality too. The items with the least factor loadings have no communalities at all.

The reduction of the 28 variables concerning the experience of flow to one basic factors sets up the basis for further evaluations.

To what extent does the joy of programming correlate with the commitment for open source? The more fun the developer has while programming and the more he is wrapped up in the activity, the more time he spends on open source projects. Table 8 clearly shows that the experience of flow has its strongest effect on the time spent on open source during spare time whereas the correlation with the time spent for open source at the work place is smaller and not significant.

The effect of the experience of flow is even stronger on the readiness for future activities for open source (see Table 9). The more fun a programmer has in his activity, the greater is his readiness for future commitment in open source projects.

Is there a connection between flow experience and the developer's output and productivity? We expect that flow affects these measures positively: The more fun a programmer has, the more frequently he works for open source and the more productive he is. However and in contrast to the study by Hertel et al. (2003, see p. 27), in which the number of patches correlated positive and with significant values with the motivational factors asked for in their study, we only get insignificant results. It is questionable whether the measure we used for the programmers' performance are useful for a study with a broad focus like the FASD study. Hertel et al. concentrated on the developers of the Linux kernel project. In this community, the interpretation of the terms (e.g. "patch", "class" etc.) might be more consistent then in the sample we investigated leading to the positive effect their study proved.

3.3 Regression analysis

Having prepared the data as described above, we're ready to carry out the regression analysis. First, we have to guarantee that the data fulfills the prerequisites needed for an OLS (*ordinary least square*) regression. The collinearity statistics showed that the data exhibited no problems with multicollinearity. On the other hand, the analysis of the standardized residuals proved the existence of considerable heteroscedastic error terms. However, this problem can be tackled using heteroscedasticity-consistent standard error estimators for the OLS regression coefficients (see Hayes and Cai (2004)). Thus, the data gathered meets the requirements for an OLS regression analysis.

We first calculated the regression analysis with the readiness for future effort as dependent variable. As independent variables we filled in the results of the factor analysis and the developer's spare time both in quadratic form. Table 10 shows the results of the first regression analysis. The factor "flow/fun" contributes

Table 7: Open Source Questionnaire: Factor loading of fun factor

	Flow/fun	**Communality**	**Loading**
1	I lose my sense of time.	0.16	0.40
2	I cannot say how long I've been with programming.	0.15	0.38
3	I am in a state of flow when I'm working.	0.25	0.50
4	I forget all my worries when I'm working.	0.29	0.54
5	It's easy for me to concentrate.	0.41	0.64
6	I'm all wrapped up in the action.	0.46	0.68
7	I am absolutely focused on what I'm programming.	0.48	0.69
8	The requirements of my work are clear to me.	0.33	0.57
9	I hardly think of the past or the future.	0.14	0.37
10	I know exactly what is required of me.	0.27	0.51
11	There are many things I would prefer doing. (-)	0.02	0.16
12	I feel that I can cope well with the demands of the situation.	0.28	0.53
13	My work is solely motivated by the fact that it will pay for me. (-)	0.00	-0.02
14	I always know exactly what I have to do.	0.30	0.55
15	I'm very absent-minded. (-)	0.00	0.01
16	I don't have to muse over other things.	0.13	0.36
17	I know how to set about it.	0.27	0.52
18	I'm completely focused.	0.51	0.71
19	I feel able to handle the problem.	0.41	0.64
20	I am extremely concentrated.	0.51	0.72
21	I'm looking forward to my programming work.	0.37	0.61
22	I enjoy my work.	0.31	0.56
23	I feel the demands upon me are excessive. (-)	0.01	0.09
24	Things just seem to fall into place.	0.31	0.56
25	I forget everything around me.	0.35	0.59
26	I accomplish my work for its own sake.	0.12	0.34
27	I completely concentrate on my programming work.	0.54	0.73
28	I am easily distracted by other things. (-)	0.13	0.36

Source: Benno Luthiger, FASD Study, University of Zurich

Table 8: Correlation of time spent for open source with flow/fun

	Total of time spent	**Spare time**	**Working time**
Pearson Correlation	0.12***	0.16***	0.03

Remark: *** Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

Table 9: Correlation of readiness for future effort with flow/fun

	Readiness for future effort
Pearson Correlation	0.37***

Remark: *** Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

significantly only with the linear term. The opportunity costs of the time do not affect the readiness for future effort at all. The model quality amounts to 15%.

An even better result occurs when using a linear model with the original items instead of the calculated factor as independent variable (see Table 11). Seven of these items contribute with significant values whereby item 20 with negative sign.[5] If this model is controlled with the demographic variables, only the respondent's age contributes significantly, namely with negative sign. The older the developers are, the less their readiness for future efforts in an open source project. This seems quite reasonable. This model explains 27% of the variance of the dependent variable.

Table 12 shows the regression analysis with the (spare) time spent as dependent variable. In this regression, the factor "flow/fun" contributes significantly, however, the quadratic term disappears again, whereas the spare time appears with quadratic term. The model quality of this regression amounts to 32%. The comparatively high beta coefficient of the spare time indicates that time spent for open source is mainly limited by the hours of spare time available.

In Table 3 we identified a subsample of our sample as professional open source

[5] 2: "I cannot say how long I've been with programming.", 20: "I am extremely concentrated.", 21: "I'm looking forward to my programming work.", 22: "I enjoy my work.", 24: "Things just seem to fall into place.", 26: "I accomplish my work for its own sake.", 27: "I completely concentrate on my programming work."

Table 10: Readiness as function of fun 1

Dependent variable	Readiness for future effort
Independent variables	**Estimated coefficients**
Flow/fun	1.122***
	(0.380)
Constant	13.671
R^2	0.145
Number of cases	917

Remark: *** Significant on 1% level
Standardized beta coefficients in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

developers. Which importance does fun while programming have for developers that are paid for their work? Table 13 shows that the factor "flow/fun" contributes only with the linear term significantly to the regression. The model quality of this regression is 24% which is significantly higher then the model quality of the regression calculated with all open source developers (Table 10). The professionals' readiness for future effort seems to be highly determined by the joy of programming.

In this study, professionals have been designated as those open source programmers that work for open source in their working hours for at least 90% of the time. If these programmers work in their spare time for open source, this engagement is exclusively determined by the availability of spare time. The factor "flow/fun" doesn't contribute with significant amount to this model. What is astonishing with this regression is the fact that it explains 83% of the dependent variable's variance (see Table 14). In addition, a regression analysis has been calculated with the whole time the professionals spend on open source projects. However, this regression did not show any significant contributions. Thus, it appears that the professionals' time spent for open source is determined by other factors then the joy of programming.

For the sake of completeness, the regression analysis has been calculated with output and production as dependent variables. However, for both of these regressions, the model quality was negligible (below 1%).

3.4 Conclusions

The results obtained in this section allow for the following interpretation:

1. Fun matters: With a model using this motivational factor we are able to explain between 27% and 32% of the engagement of an open source developer.

Table 11: Readiness as function of fun 2

Dependent variable	Readiness for future effort
Independent variables	**Estimated coefficients**
21: I'm looking forward to my programming work.	0.906*** (0.284)
22: I enjoy my work.	0.442*** (0.113)
26: I accomplish my work for its own sake.	0.261*** (0.112)
20: I am extremely concentrated.	-0.280*** (-0.097)
24: Things just seem to fall into place.	0.252*** (0.094)
27: I completely concentrate on my programming work.	0.267*** (0.090)
2: I cannot say how long I've been with programming.	0.146** (0.072)
Age	-0.060*** (-0.170)
Constant	5.954
R^2	0.270
Number of cases	1088

Remark: *** Significant on 1% level
** Significant on 5% level
Standardized beta coefficients in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

Table 12: Time spent as function of fun and time

Dependent variable	Time spent in spare time
Independent variables	**Estimated coefficients**
Flow/fun	1.210***
	(0.144)
Spare time	6.127***
	(0.781)
Spare time^2	-1.468***
	(-0.403)
Constant	8.738
R^2	0.323
Number of cases	899

Remark: *** Significant on 1% level
Standardized beta coefficients in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

Table 13: Professionals' readiness as function of fun

Dependent variable	Readiness for future effort
Independent variables	**Estimated coefficients**
Flow/fun	1.383***
	(0.493)
Constant	12.956
R^2	0.243
Number of cases	109

Remark: *** Significant on 1% level
Standardized beta coefficients in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

Table 14: Professionals' spare time spent as function of fun

Dependent variable	Time spent in spare time
Independent variables	**Estimated coefficients**
Spare time	1.173***
	(0.745)
Spare time^2	0.168***
	(0.224)
Constant	1.344
R^2	0.830
Number of cases	130

Remark: *** Significant on 1% level
Standardized beta coefficients in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

2. The availability of spare time does not matter if we ask the open source developers for their readiness for future efforts. In contrast, the availability of spare time is of great importance if we look at the amount of time actually spent for open source. This understandable result indicates the validity of the data gathered in the FASD study.

3. The joy of programming does not wear off: each additional unit of fun is transferred linearly into additional commitment. This statement can be deduced from the fact that the flow factors contribute only with linear terms significantly to the regression.

These results show that it is possible to quantify the importance of the motivational factor "fun" with the method used. Both the developers' engagement for open source (as readiness for further work or as amount of time spent) as well as the fun they enjoy can be determined. Using this data, then, we can look at the variation in the developers' engagement and calculate the portion caused by the variation of the developers' joy of programming. Thus, this method provides an explanation of a considerable amount of the open source developers' engagement.

But the outcome proves that "fun" does not account for all of the open source programmers' engagement. Our result conforms to the assumption that a multitude of motivations coexist to justify an individual developer's engagement for open source (see Franck and Jungwirth (2003) or Lakhani and Wolf (2003) for example). It would be interesting to have quantified other motivational factors as reputation or altruism with similar methods.

Our data shows further that the open source phenomenon is mainly a free time occurrence. Of the 12.6 hours a software developer works on average per week, 7.3 hours (58%) account for the developer's spare time. However, a considerable amount of 5.2 hours (42%) is done during working time. Furthermore, we have to take into consideration that our data is biased and the professional open source developers are underrepresented in our sample. We therefore assume that the paid development of open source software has catched up with the free time open source development.

Does this mean that fun becomes less important the more important paid development becomes? Our data shows clearly that the engagement of paid software developers is guided by other factors then fun. This can be deduced from the fact that the amount of time professionals (in our sample of open source developers) spend for developing open source software is completely independent of their joy of programming. However, this conclusion only holds if we look at the engagement in the form of hours worked for open source. If we look at their willingness to engage for open source, "fun" becomes even more significant. In the case of open source developers in general, the fun motive accounts for 15% of the variance in the developers' readiness (see Table 10) whereas in the case of open source professionals, this number rises to 24% (Table 13).

It can be assumed that enjoying the work increases the motivation and, thus, fun has a positive effect on the developer's productivity. If this assumption is true, the importance of fun does not diminish, on the contrary; and based on the conclusion drawn in the previous paragraph, fun should even become more relevant in the commercial software area.

In our study, one object of measurement was the developers' productivity. However, the data gathered for productivity shows little correlation with the joy of programming: Less then 1% of the developers' productivity can be explained by the developers' experience of flow. It was concluded that the way the developers' productivity was operationalized is not well suited to the area of this study. We will discuss the problem to measure and operationalize the productivity of software developers further in the next section.

4 Fun in the commercial software development area

In the previous section, the importance of fun for open source developers was examined. It was shown that the joy of programming plays an important role even for those developers that are paid for their open source engagement. In this section, the importance of fun in the commercial software development area will be investigated.

The questionnaire for the commercial software developers corresponded to the open source questionnaire concerning the flow part, but differed concerning the questions about the open source activities. Instead, the commercial developers were asked about their relation to their employer (e.g. how proud they are of their employer etc.) and their project situation (e.g. the frequency of deadlines, how strongly the feel the formal authority of the project manager etc.).

This questionnaire was filled by 114 software developers working in six Swiss software companies. This response rate is too less to make the sample representative. Therefore, the results presented in this section have to be taken with caution.

4.1 Flow of commercial developers

The preliminary analysis of the questionnaire data filled by commercial software developers showed that some items must be excluded in order to obtain acceptable values for the factor analysis. The anti-image correlation matrix showed poor values for the items 13, 15, 16 and 26.[6] After having excluded these values, the KMO value rose to a satisfactory level of 0.847. Again, only one underlying factor was sought: *flow/fun*. This single factor accounts for 35.2% of the original variance. The reliability analysis using Cronbach's α to assess the quality of the scale resulting from the factor analysis shows a good value for the extracted factor (0.91).

Table 15 shows the extracted factor's communalities and factor loadings. Factor loadings vary between 0.77 and 0.41. Thus, the data from the commercial questionnaire has significantly less variance in the factor loadings. This is consistent with the fact that the calculated factor *flow/fun* from the commercial questionnaire explains significantly more of the items' variance (35.2%) then the extracted factor *flow/fun* in the open source questionnaire (26.8%).

How does the joy of programming correlate with other factors that determine the work conditions of a commercial software developer? Table 16 shows the correlations of the flow components with the index "Engagement at work place". This index consists of the questions *29* to *31*.[7] Not surprisingly, this engagement correlates highly with the fun the developers have at their workplace.

Table 17 shows the correlations of the flow factor with the index "Relation to the employer" built with the questions *32* to *35*.[8] Again, this correlation is highly significant (although less then the previous). The better the relation to the employer, the more he enjoys his work. Or put the other way round, the more fun the software developer has while programming, the better his image of the employer.

A rather surprising impression results from the correlations of the flow factors with the questions concerning the project situation (questions *38* to *42*, see Table 18). The flow factor correlates significantly *positive* with the pressure coming from deadlines. This allows the conclusion that software developers enjoy the more fun and dive better into their activities the more the feel deadlines. This contradicts the

[6]13: "My work is solely motivated by the fact that it will pay for me.", 15: "I'm very absent-minded.", 16: "I don't have to muse over other things.", 26: "I accomplish my work for its own sake."

[7]29: "At the beginning of the week, I look forward to the work ahead.", 30: "I don't mind working overtime for my job.", 31: "I plan my future career in the IT area.".

[8]32: "I'm proud to work for the *firm*.", 33: "I can develop personally and professionally working for the *firm*.", 34: "There is a pleasant atmosphere at the *firm*, and the *firm* pursues the right objectives.", 35: "At the *firm*, the role model and the processes are implemented professionally and with success."

Table 15: Commercial Questionnaire: Factor loading of fun factor

Flow/fun		**Communality**	**Loading**
1	I lose my sense of time.	0.29	0.54
2	I cannot say how long I've been with programming.	0.21	0.46
3	I am in a state of flow when I'm working.	0.39	0.63
4	I forget all my worries when I'm working.	0.27	0.52
5	It's easy for me to concentrate.	0.42	0.65
6	I'm all wrapped up in the action.	0.52	0.72
7	I am absolutely focused on what I'm programming.	0.54	0.74
8	The requirements of my work are clear to me.	0.34	0.58
9	I hardly think of the past or the future.	0.17	0.41
10	I know exactly what is required of me.	0.16	0.41
11	There are many things I would prefer doing. (-)	0.28	0.53
12	I feel that I can cope well with the demands of the situation.	0.28	0.53
14	I always know exactly what I have to do.	0.21	0.45
17	I know how to set about it.	0.39	0.63
18	I'm completely focused.	0.59	0.77
19	I feel able to handle the problem.	0.28	0.53
20	I am extremely concentrated.	0.44	0.67
21	I'm looking forward to my programming work.	0.28	0.53
22	I enjoy my work.	0.31	0.55
23	I feel the demands upon me are excessive. (-)	0.20	0.44
24	Things just seem to fall into place.	0.54	0.73
25	I forget everything around me.	0.55	0.74
27	I completely concentrate on my programming work.	0.56	0.75
28	I am easily distracted by other things. (-)	0.24	0.49

Source: Benno Luthiger, FASD Study, University of Zurich

Table 16: Flow and engagement at work place

	Correlation coefficient	Significance level
Engagement at work place		
Flow/fun	0.402***	0.000

Remark: *** Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

Table 17: Flow and relation to the employer

	Correlation coefficient	Significance level
Relation to the employer		
Flow/fun	0.244**	0.015

Remark: ** Significant at 5% level

Source: Benno Luthiger, FASD Study, University of Zurich

Table 18: Flow and project situation

	Correlation coefficient	Significance level
38: How often do you feel deadlines?		
Flow/fun	0.271^{***}	0.006
39: How often are the projects supported by a clear project vision?		
Flow/fun	0.345^{***}	0.000

Remark: *** Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

findings made by Amabile, DeJong, and Lepper (1976), which discovered in their empirical study that deadlines affect adversely the intrinsic motivation of the persons concerned. It seems that the pressure imposed by deadlines does not impair the developer's fun in the context of a software project. On the contrary, deadlines seem to be perceived as challenge leading to higher concentration, such that the programmer buckles down to his work and, therefore, experiencing flow and having fun at work.

However, the significant correlation between the feeling of a project vision and the flow factor meets our expectations. It makes sense that an understandable project vision has a direct impact to the clearness of the task. By means of a project vision, it is possible to clarify uncertainties in the daily routine, which spurs the project work and helps the developers to concentrate.

4.2 Fun and productivity

In the open source survey we tried to assess the developers' productivity by counting the number of patches or modules developed per time unit. In the survey for the commercial developers, the number of checkins in the last days was requested. However, in both cases, the numbers gathered from the sources showed no significant correlations with the fun the developers experienced. There are two possibilities to interpret this outcome. Either we can conclude that in fact there exists no such relation between fun and developer productivity or we can infer that such a relation exists, although the productivity measures used in this study may not be appropriate for this situation.

If the first interpretation were true, this would mean that a developer would have fun while programming without being successful in solving problems as well as the reverse case, where a developer is very productive without enjoying the work. This does not seem stringent to me. A person experiences flow if his capabilities match the challenges posed by the task to fulfill. If the person is not able to accomplish the task, if she fails, she will be frustrated by the work, far from enjoying it. In the other case where the person easily achieves the work, the person will not be able to uphold the concentration needed to be productive if she does not enjoy the work but gets bored with it. Our interpretation, therefore, for the fact that we have not been able to show any correlations between fun and productivity is that we used a poor measure to assess the developer's productivity.

If the term "productivity" is considered to be a capacity to solve problems, a reasonable interpretation in the area of software development, it becomes understandable why our measures were poorly suitable for this aim. To fix a bug in a program it possibly needs only one small change on a program line in the code. However, this might be a bug that is very subtle and hard to find. New features to implement may need changes in very different areas in the project or can be implemented in a concise peace of code depending on the technology used. Inexperienced developers possibly need a number of checkins until the module runs error free whereas expert programmers realize new application functionality faultless at the first push. Depending on the circumstances and the problem to solve, high numbers reflect high productivity or exactly the opposite.

Having shown that the measures used are not adequate, the question arises how a good measure could look like. Based on our experience with the FASD study, it looks obvious to understand productivity as an interaction between professional skills, the tools used and the complexity of the task to accomplish. With constant skills and tools used, a person is more productive if the number of tasks achieved per time unit is of higher complexity then if their simple to complete. With given skills and fixed difficulty level, a person's productivity will improve if the person can use efficient tools compared to a situation where she has to use out-of-date means. With given means and fixed task complexity, a person's productivity will increase with growing capabilities.

According to Asendorpf (1996, p. 137) capabilities are "character attributes that allow achievements."[9] These capabilities consist of social and genetic components but they can also be acquired by education and experience, i.e. human capital investments. By the tools used in the case of software development we mean the combination of hardware infrastructure (performance of computer processors, transferrate of data communication etc.) and the programs used for the development process (e.g. code editors of integrated development environments, version and configuration management etc.). Therefore, to predict a software developer's productivity one method would be to review this person's experience and education as well as the tools the developer has at hand. The task to achieve and its complexity are exogenously given.

Now, what connection exists between productivity in this interpretation and fun? From our point of view, productivity is related to fun via the factor "motivation". However, motivation does not exist in our list of factors constituting

[9]"Persönlichkeitsmerkmale, die Leistungen ermöglichen."

a person's productivity. This makes sense: In our opinion, motivation results of a match between the person's capability and the challenge and complexity of the task to achieve.[10] A person with insufficient capabilities is overcharged and is not able to deliver the requested performance. An undercharged person on the other hand is bored and for this reason not able to completely exhaust her achievement potential. Thus, the element "motivation" relates the factor "capability" with the aspect "task complexity". For that a person can be productive, she not only has to have sufficient tools and capabilities available, furthermore the person's capabilities have to match the task complexity.

As shown by the research of the flow phenomenon, flow experience on its part implies a correspondence between abilities and challenges (see Figure 1). But if both motivation as well as flow experience indicate an optimal match of capabilities and challenges and if this correspondence determine a person's productivity (presupposing the availability of suitable tools and given the person's abilities), this implies nothing else than the fun enjoyed while working is a proxy for the person's productivity. Thus, both the original question of how the joy of work affects the productivity and the derived question of measuring a person's productivity become futile. Fun as such can be used as the sought-after measure for productivity.

If this deduction is true that fun is a measure for a software developer's productivity, the findings that a satisfied and confident programmer experiences more fun gets an additional importance: an employer's investments in the programmers' satisfaction and pride should pay off as increased developer productivity. Thus, the following recommendations for software companies can be made:

1. *Foster the developers pride in the firm*: Assist the programmer in his or her professional training, pay attention that the internal processes are professionally and efficiently implemented and take care of a pleasant atmosphere at the work place.

2. *Provide project visions*: Show the programmers which problems can be solved by the new software and what increase in value (for the firm, for the society) is created hereby.

3. *And above all*: Gain a clear impression of your programmers' capabilities and potential and try to offer the developers an appropriate challenge with their work.

5 Comparison between open source developers and software developers in the commercial realm

After having analyzed both data sets separately, a comparison will be provided in this section. The objective of this analysis is the verification of the hypothesis that

[10]There might be cultural and health reasons too that affect a person's motivation.

Table 19: Comparison of age

Comparison of age		
	Mean	Number
open source programmers	28.72	1274
commercial software developers	33.65	113
Total	29.12	1387

Source: Benno Luthiger, FASD Study, University of Zurich

open source developers enjoy more fun while programming as commercial software developers. In addition, provided that this verification succeeds, an attempt will be made to identify the elements of the open source model which are responsible for this difference.

First, the possibility of a systematic bias must be discussed. Although both data sets are responses of questionnaires that are very similar, partially even identical, chances are that the results are biased systematically. Subsequently, if the existence of significant differences can be proven between the two samples, these differences are not caused by a characteristic diversity of the samples, but result of the systematic biases provoked by the study design.

Indeed, looking at the two samples and how they have been gathered, there are some differences that give reasons for systematic biases. The open source survey took place in spring 2004 whereas the commercial survey was carried out six months later. The open source questionnaire was addressed at a world wide audience and was answered primarily in English whereas the respondents of the commercial survey were German speaking software developers in Switzerland. The programmers in the open source sample were truly self selected whereas the commercial developers were prompted by their employers to participate.

Luckily we have the possibility to bypass these differences by comparing not only the two samples, but the professionals and the open source hackers within the open source sample. If this late comparison gives the same result as the first, we have good reasons to disregard the impact of systematic biases.

5.1 Comparison of age and gender

Open source programmers are 28.7 years old on the average, developers in commercial software firms are about 5 years older (33.7 years, see Table 19). This difference is highly significant ($\alpha < 1\%$).

The share of female programmers in the open source sample amounts to 2%. In the commercial software area, however, the share of female developers is six times higher and amounts to 12% (see cross table 20). Again, this difference is highly significant.

Table 20: Comparison of gender distribution

Gender	Open Source	commer-cial	Total
male	1270 (1259) 98.1%	99 (110) 87.6%	1369 97.3%
female	24 (35) 1.9%	14 (3) 12.4%	38 2.7%
Total	1294	113	1407

Remark: Expected values in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

5.2 Comparison of project situation

Beside of the flow experience we have asked questions concerning the project situation in both surveys thus making it possible to compare the answers. The comparison shows clearly the differences between the two software development models. However, Figure 3 shows that as soon as money comes into play the differences get less accentuated.

As expected, deadlines play virtual no role in open source projects whereas in commercial software projects, they are very noticeable. The difference diminishes somehow when we compare hackers with open source professionals but is still highly significant. As expected, open source projects are backed more often by clear project visions, in comparison to commercial software projects. Again, payed open source programmers perceive less project visions than their volunteering counterparts. The comparisons concerning the desired amount of project visions and the professional competence of the project leader are pronounced (see Table 21).

5.3 Comparison of fun

To compare the flow experience of the two samples, the flow factor resulting from the factor analysis is reconsidered, by analyzing the data carried out for the two samples. Additionally we can calculate the mean of all 28 flow items and compare that value. Figure 4 makes apparent that open source developers show a higher value for both the flow factor and the average value. Table 22 proves that this difference is highly significant indeed ($\alpha < 1\%$).

To control for systematic biases, the flow experience of professional open source developers with the fun open source hackers enjoy must be compared. This

Table 21: Project situation: Comparisons

T-Test for equality of the means		
Project situation	Open source vers. Commercial	Hacker vers. Professional
deadlines	-2.203***	-0.779***
	(-21.311)	(-5.573)
project visions: actual	0.826***	0.354***
	(7.010)	(2.780)
project visions: desired	0.211*	-0.060
	(1.935)	(-0.570)
professional competence	-0.038	-0.213*
	(-0.357)	(-1.898)

Remark: *** Significant at 1% level
* Significant at 10% level
T values in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

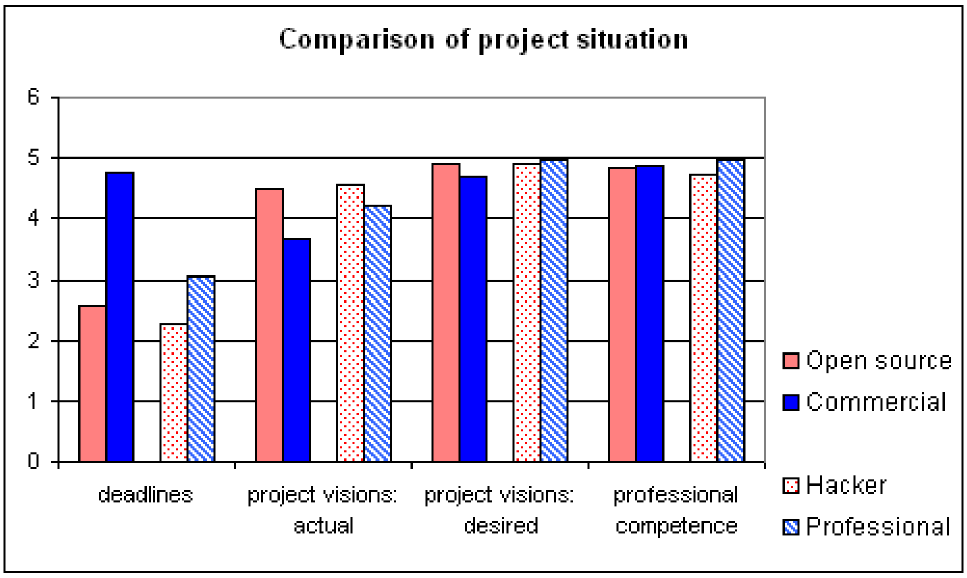

Figure 3: Comparison of project situation

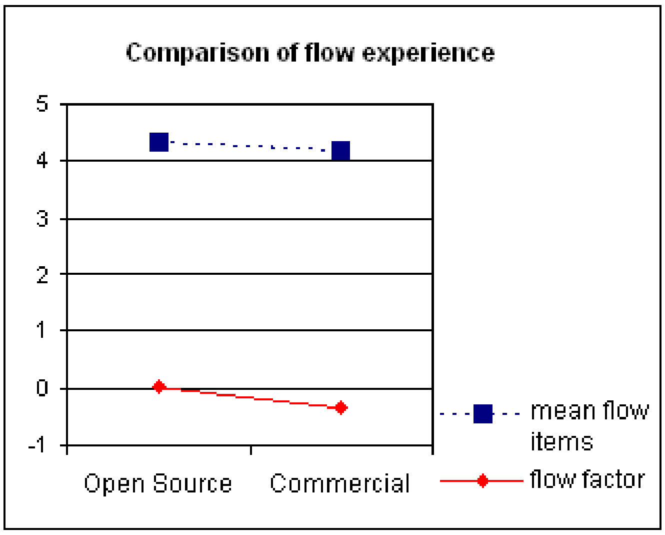

Figure 4: Comparison of flow experience of software developers

Table 22: Flow experience of open source and commercial developers: comparison

T-Test for equality of the means			
Flow factor	**Difference**	**Significance**	**T value**
Flow/fun	0.383***	0.001	3.475

Remark: *** Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

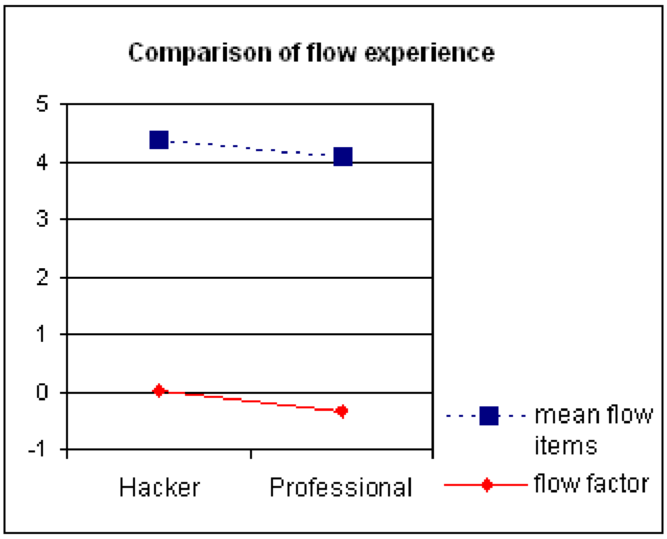

Figure 5: Comparison of the flow experience of open source developers

Table 23: Flow experience of hackers and professionals: comparison

T-Test for equality of the means			
Factor	**Difference**	**Significance**	**T value**
Flow/fun	0.360[***]	0.001	3.283

Remark: [***] Significant at 1% level

Source: Benno Luthiger, FASD Study, University of Zurich

comparison corroborates the findings of the first comparison. Again, the open source hackers show a higher value for the flow/fun experience compared with the programmers paid for their work (see Figure 5, Table 23).

Obviously, the context where the software development is done has an impact on the joy of work that can't be denied. Therefore, it can be concluded, the motivational factor "fun" can explain why software developers engage for open source projects for free: They do this in their spare time because it makes fun, i.e. it is much more fun to program for free then to develop software under commercial conditions.

Table 24: Characteristic differences of software development models

Feature	**open source**	**commercial**
project vision (+)	+	?
formal authority (-)	-	+
monetary incentives (?)	-	+
deadlines (-)	-	+
optimal challenge (+)	+	?

Source: Benno Luthiger, FASD Study, University of Zurich

5.4 What makes the difference

The results in the previous subsections confirm our hypothesis: The same activity, i.e. software development, generates less fun when it is practiced under commercial conditions. Thus, the question arises, which condition causes this difference. Which feature of the open source development model makes programming in this context more fun? And is this feature constitutive for the open source development model or is it possible to apply it to the commercial software development model too?

In Table 24, some characteristic differences of the two software development models have been identified. A plus sign in parentheses marks that the attribute is expected to positively affect the joy of programming, and a plus sign in the columns indicates that the attribute exists for the respective development model.

Open source projects[11] are usually driven by a clear *project vision*. The project leader knows the reason why he has founded the project and chosen the appropriate open source license. He has a clear concept of the project goals. In addition, he has good reasons to communicate the project goals to other persons, e.g. the actual and potential developers in the project. A convincing project vision is very well suited to convince potential contributors to engage as well as to coordinate the contributions of the different participating programmers and to align them to the shared goal. Project visions can play a role in commercial software projects too. However, in a software company, it is, first of all, the formal authority of a superior that ultimately decides which activities a programmer will work on and how this work is evaluated. Thus, the project vision doesn't have the same coordinating function as in open source projects.

The hierarchical structure of commercial software companies makes another difference. Hierarchies establish *formal authorities*. The owner of formal authority can demand a certain type of behavior from employees. This is not possible for project leaders of open source projects. The latter have to persuade contributors by

[11]The following descriptions apply to open source projects that result from a voluntary cooperation. They are less true for open source projects with developers that are paid for their project work.

their professional competence in an objective dispute if they are not satisfied with the contributions of participating developers.

Besides hierarchies and formal authorities, commercial software companies can provide *monetary incentives* that distinguish them from open source projects — software developers are paid for their work. This possibility is not available for open source projects based on voluntary cooperation.

Usually commercial software projects are part of a commercial production and exploitation process: the software has to be delivered at a certain time with specific functionality. Therefore, *deadlines* are an inevitable part of commercial production processes. In contrast, each participant in open source projects can easily evade any deadlines. As in other aspects of voluntary cooperation, deadlines come into effect only by voluntary agreement.

A last distinctive feature of the two development model is related to *challenges* to software developers. In an open source project a developer can control challenges by self–selecting involvement in specific projects. Thus, an optimal challenge can be derived. In commercial projects on the other hand project engagement is determined by the needs of the firm, not the individual. A given project, therefore, has only limited possibilities to take into consideration a developer's potential — his abilities, experience and expertise — as well as his interests in professional and personal evolution.

How can we verify that the described distinctive features really cause the difference concerning the experience of fun? If a feature has an effect on the flow experience, this should be indicated by appropriate correlations. In the case of deadlines for example we expect that the developer experience less flow the more he suffers from deadlines. The same holds for formal authority: the more such authority is sensible, the less fun the programmer should have. Vice versa we expect a positive effect in the cases of project visions and optimal challenge: the more sensible the project vision and the better the match between abilities and challenges, the more joy of programming we predict. Concerning the effect of monetary incentives we neither can predict the sign of the correlation nor have we the data to compute any correlations. The other predictions we are able to test using the answers of the questions concerning the developers' feelings about the open source project (questions *32* - *35* in the open source survey) and the employees' relation to the company (questions *38* - *42* in the commercial survey). The feature "optimal challenge" was calculated from the commercial developer's answers to the items *33* and *35*.[12] The optimal match of abilities and challenges is considered as given, in open source projects, at least for developers that aren't paid for their open source engagement; hence we have no data to calculate such a correlation for open source developers. Table 25 shows the significant correlations.

Interestingly, the feature "deadlines" is significantly correlated with fun. Against our expectations, however, the sign of this correlation is positive for both the open

[12] *33*: "I can develop personally and professionally working for the firm", *35*: "At the firm, the role model and the processes are implemented professionally and with success".

Table 25: Correlations of distinctive features with flow/fun

	Sample		
Feature	open source	commercial	all
Frequency of deadlines	0.103^{***} (930)	0.256^{**} (88)	0.057^{*} (1018)
Frequency of a clear project vision	0.339^{***} (929)	0.358^{***} (86)	0.354^{***} (1015)
Importance of the project leader's professional competence	0.169^{***} (927)	-0.034 (88)	0.152^{***} (1015)
Formal authority of the project manager	-	0.115 (101)	-
Optimal challenge	-	0.270^{***} (102)	-

Remark: *** Significant at 1% level
** Significant at 5% level
* Significant at 10% level
Number of values in parenthesis.

Source: Benno Luthiger, FASD Study, University of Zurich

source as well as the commercial sample, meaning that the software developers that feel more deadlines at the same time enjoy more fun while programming. This contradicts the findings made by DeMarco and Lister (1987) and Amabile et al. (1976) which observed a negative impact of deadlines on the subject's intrinsic motivation. Whether the project manager's formal authority in a commercial software project is strongly sensible or not does not affect the developer's experience of fun while programming. More in line with our expectations are the other results. A comprehensible project vision correlates strongly significant with fun as does the optimal challenge the developer has while programming. The project leader's professional competence has a positive impact on the developer's joy of programming in the open source context; however, this is not the case in a commercial software project.

Therefore, we can conclude from this comparison: Software development is significantly more fun when it takes place in the context of an open source project than when it is carried out under commercial conditions. However, the reasons for this difference are not deadlines, but differences based on the project's vision and the challenge of the work at hand. Because there is a correlation between a high frequency of deadlines and fun, the sheer existence of deadlines in commercial programming makes this sort of work more fun than less deadline–dependent open source development. Therefore, difference in the experience of flow between open source and commercial development hinges largely on project visions and optimal challenges. As shown in Table 21 (and Figure 3), open source projects have definite project visions and present unique challenges to programmers; hence open source is positively correlated with fun.

6 Conclusions

The analysis in the preceding sections shows that open source is mainly a part–time job. An open source developer participates on average for 12.6 hours per week in open source projects; 58% of this activity occurs in his spare time. From the 1330 software developers that participated in the survey, two-thirds work fully or partially for free for open source projects. In all probability, however, these numbers underestimate the share of paid contributions to open source. The open source platforms investigated in this research are very well suited to support open source projects started and driven by hobbyists. Professional open source projects on the contrary are able to operate their own project infrastructure and, therefore, are most likely underrepresented in the study sample. Based on this consideration, we may infer that paid contributions to open source have perhaps caught up with voluntary efforts.

In the previous section we were able to show that fun plays an important role to set up the open source developer's engagement. With the joy of programming, 27% of the open source developer's readiness for future effort can be explained. If the time the developer spends for his open source activities are accounted for, taking into consideration the available spare time he has, the joy of programming accounts for 33% of the developer's engagement. It is notable that fun matters for paid open source developers and commercial programmers as well to motivate their engagement. Thus, this study proves that it is possible to quantify the importance of a specific kind of motivational tool that leads a software developer to engage in open source activities. It would be interesting if other motivational tools behind open source could be quantified in an analogous manner.

The comparison between the data gathered from the open source survey and the commercial survey shows that open source developers experience more fun while programming then programmers working under commercial conditions. The same result is arrived at, if the answers of open source developers given by those working in their spare time (*hackers*) with those given by open source developers that are paid for their contributions. We conclude that this difference is not caused by a systematic bias in the study design but a consequence of the production conditions that govern the work of software developers.

With the results of this study, therefore, we are able to adjust the image of open source as product of unselfish and possibly somewhat weird computer hackers. Software developers that want to consume a *homo ludens payoff* act absolutely rational if they look for software projects and project situations that promise a lot of fun. Our data proves that developing for open source projects offers more and better possibilities to experience fun. Therefore, open source software can be understood, at least partially, as a by-product of an activity that makes fun and a development model that supports the need for fun in an optimal way.

The following finding offers interesting perspectives: neither deadlines nor the existence of formal authorities account for the difference concerning the fun experience between the open source and the commercial software development model. If the joy of programming is tied to project visions and optimal challenge, both factors that can be influenced be employers, wouldn't it be possible to make software development under commercial conditions as fun as it is in open source projects? To answer this question, we have to discuss the elements characterizing the open source development model: how important are they for the experience of fun and how they could realized under commercial conditions? In a short summarization, the characterizing elements of the open source development model are that this software is produced by a open community of software developers with low entry barrier, where the producers are at the same time the users of the software and where the programmer's status in the community is achieved not by formal authority but through meritocracy. Two additional important characteristics are that open source software is released early and often and that the programmers engage on a voluntary basis.

The openness of a developer community is hardly to achieve in commercial software projects. Under commercial conditions, the programmers are paid for their work and, thus, have some kind of contractual relationship to the employer. Therefore, the pool where the project can look for programmers is not potentially unlimited but restricted to the boundary of the company. However, the fact that the group of programmers involved in a software project is open or closed most probably has no impact of the fun the developers have while programming.

The characteristic that the developers of open source projects are most times actively using the application they develop ("eat their own dogfood") is true for commercial projects probably only by way of exception. This fact has implications for the project vision and, as our study shows, subsequently on the joy of work. For a "prosumer" the project vision results immediately from his requirements. In cases where the user is not identical with the producer, there is increased need for an explicit and understandable project vision.

The absence of a formal hierarchy that characterizes open source projects is not viable for commercial projects too. Beyond a doubt, the negative effects of such hierarchies can be mitigated, for example if the company has shallow hierarchies and where these hierarchies are created through transparent processes that are driven by merit economical mechanisms. Moreover, as our study shows (see Table 25), the existence of formals authorities seems to have no effect on the employees' ability to experience fun.

Open source projects distinguish themselves by high release frequencies. For a software developer, it is satisfactory to see his contributions quickly integrated into the productive application. On the other hand it is frustrating to find the contribution dumped by the project leader or overworked because of insufficient quality.

However, this is a matter of the software project's quality management and independent of whether the project is open source or commercial.

The last feature characterizing open source projects, the voluntary engagement of software developers, is not applicable for commercial projects where the programmers are paid for their work. However, voluntariness is the decisive element for finding the optimal challenge for a programmer given his abilities. If the engagement is voluntary, it's the programmer who decides which project he will engage. A crucial factor for his engagement is the challenge he will find in the project. He does not need complete information neither about the project nor his capabilities. The voluntariness of engagement allows him to select by trial and error the project that fits him most. This is possible because there's a kind of market place: Open source projects actively seek for the engagement of developers who are willing to contribute and these are able to pick a project and to leave it again at will.

This kind of voluntariness is not viable for commercial projects working with paid programmers (see Barnett, 2004, p. 8f). However, would it not be possible that a software company establishes some kind of internal market place where various projects in the company woo for the company's software developers? In such a company, the programmers are obliged to engage in projects, but have the choice to select the one that is most suitable. As a matter of course this would work only for companies with suitable size because only such companies have the variety of software projects needed to establish such a market place. A commercial software company offering such a market place would implement as much as possible from the open source development model and, therefore, would offer their software developers the best possibilities to experience fun while programming.

Again, this analysis boils down to "project vision" and "optimal challenge" as vital factors for the joy of software development. It is comprehensible that the project situation of a commercial project offers fewer possibilities to software developers to contribute their potential in an optimal way to the project. It is also understandable if the responsible persons in commercial software firms do without project visions that can be understood by the programmers. However, this is not imperative. It is at most expensive to formulate a project vision and to find an optimal challenge for the software developers participating in the project, but it is not impossible. For this reason, one can conclude from our analysis, it is not impossible either that a software project in a commercial context can be as much fun as a project occurring under open source conditions.

That said, a subsequent, but different question arises: would it pay off for an employer to offer a workplace situation that makes programming fun? In other words: do programmers that enjoy their work contribute as much to the company's success that it's worth the effort?

Based on the knowledge we gathered with this study we would answer in the affirmative. We come to this conclusion based on two considerations. In general it is true that employees that like their work contribute positively to the working climate, that they are more motivated at their work and that the show less absence from work. Specifically, as working with fun indicates that the developer's abilities match the task challenges in an optimal way, a programmer having fun is at the same time productive. They get the best of their abilities.

Because fun is a pervasive feature of software development, fun can be leveraged even for software development under commercial conditions. With the study of the open source development model we can learn, besides many other things, how fun is experienced while programming, which factor's are responsible that programmers have fun and which of the software development model's elements constitute the optimal conditions to experience fun while programming.

References

Amabile, T., DeJong, W., Lepper, M., 1976. Effects of externally imposed deadlines on subsequent intrinsic motivation. Journal of Personality and Social Psychology (34), 92–98.

Asendorpf, J. B., 1996. Psychologie der Persönlichkeit: Grundlagen. Springer-Verlag, Berlin.

Barnett, L., 2004. Applying open source processes in corporate development organizations. `<http://vasoftware.com/sourceforge/request_info-dl.php?paper=9>` (Zugriff 7.10.2004) .

Chen, H., 2002. Exploring web users' on-line positive affects .

Csikszentmihalyi, M., 1975. Beyond Boredom and Anxiety. Jossey-Bass, San Francisco.

DeMarco, T., Lister, T., 1987. Peopleware: Productive Projects and Teams. Dorset House Publishing, New York.

Franck, E., Jungwirth, C., 2003. Reconciling rent-seekers and donators. Journal of Management and Governance 7, 401–421.

Hars, A., Ou, S., 2001. Working for free? - motivations of participating in open source projects. 34th Annual Hawaii International Conference on System Sciences 7, 1–7.

Hayes, A. F., Cai, L., 2004. Using heteroscedasticity-consistent standard error estimators in ols regression with applications to moderated multiple regression. `<http://www.jcomm.ohio-state.edu/ahayes/hcse.pdf>` (Zugriff 20.1.2005) .

Hertel, G., Niedner, S., Herrmann, S., 2003. Motivation of software developers in open source projects. Research Policy 32 (7), 1159–1177.

Lakhani, K. R., Wolf, R. G., 2003. Why hackers do what they do: Understanding motivation effort in free/open source software projects. `<http://opensource.mit.edu/papers/lakhaniwolf.pdf>` (Zugriff 6.10.2003) .

Lerner, J., Tirole, J., 2002. Some simple economics of open source. Journal of Industrial Economics 52 (6), 197–234.

Luthiger, B., 2005. Fun and software development. In: Scotto, M., Succi, G. (Eds.), Proceedings of the 1st International Conference on Open Source Systems. ECIG, Genova.

Luthiger, B., 2006. Spass und software-entwicklung: Zur motivation von open-source-programmierern. <http://www.dissertationen.unizh.ch/2006/luthigerstoll/diss.pdf> (Zugriff 13.7.2006) .

Montgomery, H., Sharafi, P., Hedman, L. R., 2004. Engaging in activities involving information technology: Dimensions, modes, and flow. Human Factors 46 (2), 334–348.

Novak, T. P., Hoffman, D. L., 1997. Measuring the flow experience among web users. <http://www2000.ogsm.vanderbilt.edu/> (Zugriff 10.10.2002) .

Novak, T. P., Hoffman, D. L., Yung, Y.-F., 1998. Measuring the flow construc in online environments: A structural modeling approach. <http://www2000.ogsm.vanderbilt.edu/> (Zugriff 10.10.2002) .

Osterloh, M., Rota, S., Kuster, B., 2002. Open source software production: Climbing on the shoulders of giants. <http://www.iou.unizh.ch/orga/downloads/publikationen/osterlohrotakuster.pdf> (Zugriff 10.4.2006) .

Remy, K., 2002. Entwicklung eines Fragebogens zum Flow-Erleben. Fakultät für Psychologie und Sportwissenschaft, Bielefeld: Diplomarbeit.

Rheinberg, F., Vollmeyer, R., Engeser, S., 2002. Die erfassung des flow-erlebens. In: Stiensmeier-Pelster, J., Rheinberg, F. (Eds.), Diagnostik von Motivation und Selbstkonzept. Hogrefe, Göttingen.