2013 17th European Conference on Software Maintenance and Reengineering

Analyzing the Eclipse API Usage: Putting the Developer in the Loop

John Businge, Alexander Serebrenik, Mark van den Brand
Eindhoven University of Technology
Eindhoven, The Netherlands
{*j.businge,a.serebrenik,m.g.j.v.d.brand*}*@tue.nl*

***Abstract*—Eclipse guidelines distinguish between two types of interfaces provided to third-party developers, i.e., APIs and non-APIs. APIs are stable and supported, while non-APIs are unstable, unsupported and discouraged as they are subject to arbitrary change or removal without notice. In our previous work, we found that despite the discouragement of Eclipse, the use of non-APIs in Eclipse third-party plug-ins (ETPs) is not uncommon. Furthermore, we found that these non-APIs are the main cause of ETP incompatibilities in forthcoming releases of the Eclipse.**

In the current work we conducted a survey aiming at understanding why do the ETP developers use non-APIs. We have observed that developers with a level of education of up to master degree have a tendency not to read product manuals/guidelines. Furthermore, while for less experienced developers instability of the non-APIs overshadows their benefits, more experienced developers prefer to enjoy the benefits of non-APIs despite the instability problem. Finally, we have observed that there are no significant differences between Open Source and commercial Eclipse products in terms of awareness of Eclipse guidelines and interfaces, Eclipse product size and updating of Eclipse product in the new SDK releases.

***Keywords*-Eclipse; APIs; non-APIs; Survey; Developers**

I. Introduction

Software engineering researchers constantly carry out studies with the aim of obtain "convincing evidence" that may improve software projects. Oram and Wilson [1] state that despite the decades of software engineering research, so far they has seen relatively few examples of convincing evidence that have actually led to changes in how people run software projects. Oram and Wilson conjecture that the possible reason for relatively few examples of convincing evidence could be a result of a context problem: researchers have been generating evidence about different topics than the ones the developers care about. To understand the actual developers' needs Oram and Wilson [1] recommend the researchers to include the developer in the loop.

Typical examples of excluding a developer from the loop are in our previous studies of Eclipse interfaces [2], [3]. Eclipse distinguishes between two types of interfaces: API that are stable, supported and non-APIs that are unstable, unsupported and discouraged as they are subject to arbitrary removal without notice. In [2], we discovered that 44% of Eclipse third-party plug-ins (ETPs) on SourceForge use non-APIs. We also discovered that the ETPs that depend on at least one non-API are larger than those that depend only on APIs [2]. Furthermore, we also discovered that the Eclipse non-APIs cause ETPs to fail when ported to new releases of Eclipse SDK [3]. However, we could not answer why the ETPs that depend on at least one non-API are larger than those that depend only on APIs and why the developers use problematic non-APIs. Furthermore, in our related studies [2], [3], [4], [5], as a threat to external validity, we were not certain whether our results could be generalized as our investigation was based on only open-source software products.

Therefore, we decided to include the developer in the loop by carrying out a survey on Eclipse interface usage by Eclipse product developers so as to complete the picture.

The paper is organized as follows: In Section II we discuss the study definitions, design, and planning. Results are presented and discussed in Section III. Section IV discusses the threats of the study. Section V discusses related work and Section VI concludes the paper.

II. Study Definition, Design, and Planning

The survey was conducted during the summer of 2012. The main aim of the survey was to achieve an accurate picture of the state-of-the-practice of Eclipse interface usage by Eclipse product developers. The goals of the survey are twofold. First, we aim at identifying whether application developers are aware of the differences between APIs and non-APIs and related guidelines. Second, we aim at understanding the differences in characteristics of software applications that use APIs and non-APIs.

A. Research questions

Given the above goals, the survey answers three main research questions below. First, the research questions are based on the following categories encapsulating the different factors that may affect the development and maintenance of Eclipse products (the factors are outlined in Section II-B): 1) Eclipse interfaces and guidelines, 2) Eclipse product size, and 3) updating Eclipse product.

- RQ1 What is the relationship between factors in the identified categories and how can we explain the observed relationship?
- RQ2 What are the differences in characteristics between Eclipse products that use and do not use non-APIs in terms of the factors in the identified categories?

1534-5351/13 $26.00 © 2013 IEEE
DOI 10.1109/CSMR.2013.14

RQ3 What are the differences in characteristics between commercial and open-source Eclipse products in terms of the factors in the identified categories?

B. Survey terms and entities of interest

In this section we define the entities of interest that form the basis for design of the survey questionnaire. First, we define the terms used in the survey:

- *Eclipse SDK*: This is an extensible platform that provides a core of services for controlling a set of tools working together to support programming tasks. The SDK provides generic interfaces that can be used to build specialized applications.
- *Eclipse non-APIs*: In addition to Java access levels, Eclipse has another level called internal implementations. According to Eclipse naming convention [6], the non-APIs are artifacts found in a package with the segment `internal` in a fully qualified package name. These artifacts may include public Java classes, interfaces, public or protected methods, or fields in such a class or interface.
- *Eclipse APIs*: As opposed to non-APIs, APIs are found in packages that do not contain the segment `internal`.
- *Eclipse provisional API guidelines*: This is a document provided by Eclipse describing the different interfaces it provides and the evolution of these interfaces from initial embryonic forms to real battle-hardened APIs with guarantees of long term support. The document defines public interfaces that can be used, i.e., APIs and those that should not be used or used at a risk (non-APIs). The document also sets out rules for committers on how to indicate APIs that are still under development and subject to change. The guidelines are also useful for API clients who want to know about the state of a given Eclipse interface they are using [7].
- *Eclipse product*: This is a software system that reuses the functionality provided by Eclipse SDK. The Eclipse product is sometimes referred to as Eclipse solution and sometimes Eclipse third-party plug-in.

We identified five groups to help us investigate the previously formulated the research questions:

- *Education and experience*: The aim of this category of questions was to establish the experience and education levels of the respondents. In particular, we were interested in collecting information on years of education and years of experience as a software developer, Java developer and Eclipse product developer.
- *General Eclipse product development*: The aim of this category of questions was to identify the characteristics of the Eclipse product developers and the human aspects influencing maintenance of the Eclipse product. In particular, we were interested in collecting data about reasons why developers develop their Eclipse products, development team size, weekly hours dedicated for maintenance and development, importance of updating, number of versions (NOV) released, and number of files (NOF) of the latest version of product.
- *Eclipse interfaces and their guidelines*: The aim of this category of questions was to discover if developers are aware of the existence of the Eclipse provisional guidelines and accessibility levels of the interfaces that Eclipse provides. In particular, we wanted to collect data about developers awareness of the "Eclipse Provisional API Guidelines" and if they are aware of the two types of interfaces, APIs and non-APIs.
- *Use of non-APIs*: The aim of the questions in this category was to identify why developers use or do not use non-APIs, avoid using non-APIs and if they deliberately or not avoid using non-APIs.
- *Testing of Eclipse product*: The aim of this category of questions was to identify if developers test their versions with new Eclipse SDK releases/milestones/betas.

C. Questionnaire design

The questionnaire contains both open-ended and multiple-choice questions. The questions in the survey are organised according to the identified categories in Section II-B. The survey was designed using `limesurvey`,[1] an open-source web-based survey application. Tables I and VIII present the multiple-choice and open-ended questions, respectively.

D. Survey execution

In order to increase acceptability of the survey by respondents, as well as avoid ambiguous and biased questions in the survey, the survey was reviewed internally and externally. The internal review was done by ten colleagues at the university. For the external review, about three months before the survey was deployed, we invited Eclipse product developers to review the survey. Three developers accepted our invitation and gave us remarks, which we incorporated in the survey.

The survey was announced on Eclipse Twitter account and news page by Wayne Beaton, the director of open-source projects, Eclipse foundation. The data was collected between 15 June and 31 July 2012.

III. Results

In this section, we will first present the information about the non-respondents, then present the descriptive statistics and finally discuss the results of the analysis corresponding to the research questions. We shared the survey data of our study for replication or further analyses [8].

[1] http://www.limesurvey.org

Var #	Variable	Scale	Description
Education and Experience of Respondents			
i	educa	1–6	How many years of education after secondary school graduation do you have?
ii	expsd	1–6	How many years of experience as a software developer do you have?
iii	expjd	1–6	How many years of experience as a Java developer do you have?
iv	exped	1–6	How many years of experience as an Eclipse product/solution developer do you have?
General Eclipse Product Development			
v	whydev	N/A	Why did you develop your Eclipse product?
vi	licnc	N/A	What is the licence type of your Eclipse product?
vii	store	N/A	If option in (vi) was open-source. Where did you publish the source code of your Eclipse product?
viii	tsize	1–5	What is the average size of your Eclipse product development team?
ix	upimp	1–3	How important is it to update your Eclipse product in relation to new versions of Eclipse SDK APIs?
x	NOV	1–7	How many new versions of Eclipse product have you released since the initial release?
xi	NOF	1–4	What is the size (estimate) of the latest version of your Eclipse product in terms of number of files?
xii	whours	1–5	How many hours/week do you dedicate updating your Eclipse product in relation to Eclipse SDK API usage?
Awareness of Eclipse Interfaces and their Guidelines			
xiii	awAPg	1–2	Are you aware of the "Eclipse Provisional API Guidelines"?
xiv	flAPg	1–4	Do you follow "Eclipse Provisional API Guidelines" when developing your Eclipse product(s)?
xv	awAnNA	1–2	Are you aware that Eclipse provides two types of interfaces (APIs and non-APIs)?
Use of non-APIs			
xvi	usenNA	N/A	Do you use non-APIs?
xvii	avusNA	N/A	If answer was "NO" in (xvi). Do you deliberately avoid using Eclipse non-APIs?
xviii	deusNP	N/A	If answer was "YES" in (xvi). Do you deliberately use the non-APIs?
Testing of Eclipse Product			
xix	1stSDK	N/A	On which Eclipse SDK release did you develop the first version of your Eclipse product?
xx	testP	N/A	Have you tested your Eclipse product on newer eclipse SDK releases, betas or milestones?
xxi	freTst	N/A	If answer was "YES, the on next release and/or on a release after the next or later. Are you frequently testing you Eclipse product on new SDK releases?

Table I: Variable description and corresponding questions. Scale—Ordinal scale

A. Non-respondent analysis

First, we conducted a data reduction step where incomplete and invalid responses were excluded from the analysis. The survey attracted 114 respondents who started to fill in the questionnaire. All the 114 respondents filled the first section—*Education and experience*, of the survey. Only 40 of the 114 continued and filled the second section—*General Eclipse product development*. 36 of the 40 who filled the second section continued to fill the third section–*Eclipse interfaces and guidelines*. 31 of the 36 who filled the third section continued to fill the fourth and fifth sections—*Use of non-APIs* and *Testing of the Eclipse product*, respectively. Only one respondent of the 31 filled in meaningless answers.

B. Descriptive Statistics

In this section we present the descriptive statistics for the variables corresponding to multiple choice questions. Table I presents the questions and the respective variable names. Due to question branching, some questions were not filled by all the 30 respondents. In the descriptive statistics, we shall explicitly state the number of respondents that answered the question only when the number is less than 30.

1) Education and Experience of Respondents: Table II shows the frequency and percentage distributions of the four variables that belong to this category, i.e., *educa*, *expsd*, *expjd*, and *exped*.

2) General Eclipse Product Development: The following list presents the frequency and percentage distributions of the cases of the variables in this category. The item number corresponds to the variable number in Table I.

Years	**educa** (i)		**expsd** (ii)		**expjd** (iii)		**exped** (iv)	
	fre	%	fre	%	fre	%	fre	%
<2	1	3.3	0	0.0	0	0.0	1	3.3
≥ 2 & <4	0	0.0	2	6.7	3	10.0	7	23.3
≥ 4 & <6	16	53.3	1	3.3	2	6.7	8	26.7
≥ 6 & <8	6	20.0	2	6.7	7	23.3	7	23.3
≥ 7 & <10	2	6.7	5	16.7	4	13.3	6	20.0
≥ 10	5	16.7	20	66.7	14	46.7	1	3.3
Total	30	100	30	100	30	100	30	100

Table II: Years of education and experience of the respondents. fre–Frequency and %–Percentage. The number in brackets after every variable name corresponds to the variable number in Table I.

v) *Why did you develop your Eclipse product?*: 13 (43.3%) of the 30 were employed on *full-time job*, 4 (13.3%) were employed as a *part-time job*, 6 (20.0%) developed plug-ins as a *hobby*, 2 (6.7%) developed the plug-in as part of an *educational use*, and 5 (16.7%) were in the *other* category.

vi) *What is the licence type of your Eclipse product?*: 16 (53.3%) of the 30 selected *open-source* and 14 (46.9%) selected *commercial*.

vii) *If option in (vi) was open-source. Where did you publish the source code of your Eclipse product?*: 16 respondents answered this question, where 6 (37.5%) of the 16 had SourceForge as their repository, 2 (12.5%) had GoogleCode, 2 (12.5%) had GitHub, 4 (25.0%) had eclipse.org and 2 (12.5%) were in the *other* category.

viii) *What is the average size of your Eclipse product development team?*: 11 (30.8%) of the 30 developed the Eclipse product *solely*, 11 (36.7%) had a team of *2*

to 4 persons, 2 (6.7%) had a team of *5 to 7 persons*, 3 (10.0%) had a team of *8 to 10 persons* and 3 (10.0%) had a team of *more than 10 persons*.

ix) *How important is it to update your Eclipse product in relation to new versions of Eclipse SDK APIs?*: 6 (20.0%) of the 30 updating their Eclipse product in relation to Eclipse SDK APIs was *not important*, 12 (40.0%) updating was *moderately important* and 12 (41.0%) updating was *very important*.

x) *How many new versions of Eclipse product have you released since the initial release?*: 1 (3.3%) of the 30 had not released a version of the Eclipse product since the initial release, 11 (36.7%) had released *2 to 5* versions, 7 (23.3%) had released *6 to 10* versions, 5 (16.7%) had released *11 to 15* versions, 1 (3.3%) had released *16 to 20* versions, 0 (0.0%) had released *21 to 25* versions and 5 (16.7%) had released *more than 25* versions.

xi) *What is the size (estimate) of the latest version of your Eclipse product in terms of number of files (NOF)?*: 7 (23.3%) of the 30 had *less 100 files* for the latest version of the Eclipse product, 10 (33.3%) had *101–500* files, 2 (6.7%) had *501–1000* files and 11 (36.7%) had *more than 1000 files*.

xii) *How many hours/week do you dedicate updating your Eclipse product in relation to Eclipse SDK API usage?*: all the thirty respondents chose the option of *less than 10 hours*.

3) Awareness of Eclipse Interfaces and their Guidelines: The following list presents the frequency and percentage distribution of the cases of the variables that belong this category.

xiii) *Are you aware of the "Eclipse Provisional API Guidelines"?*: 15 of the respondents chose the option they are *this is the first time I have heard of them* of the *Eclipse provisional API guidelines* and the remaining 15 the option *I am already aware*.

xiv) *Do you follow "Eclipse Provisional API Guidelines" when developing your Eclipse product(s)?*: 14 (46.7%) of the 30 responded that *they do not know, because they do not know the guidelines*, 3 (10.0%) responded that they *never* follow the guidelines, 12 (40.0%) responded that they *sometimes* follow the guidelines and 1 (3.3%) responded that they *always* follow the guidelines.

xv) *Are you aware that Eclipse provides two types of interfaces (APIs and non-APIs)?*: 3 (10.0%) of the 30 responded that the survey *just informed* them that Eclipse provides two types of interfaces, APIs and non-APIs while 27 (90%) responded that they were *already aware* of the Eclipse APIs and the non-APIs.

4) Use of non-APIs: The following list presents the frequency and percentage distributions of the cases in the variables that belong to this category

	Frequency (Percentage)				
Resp	nxtRL	rlANA	nwMV	nwBV	NO
Selected	20(66.7%)	8(26.7%)	12(40%)	8 (26.7%)	5(16.7%)
Not selected	10(33.3%)	22(73.3%)	18(60%)	22(73.3%)	25(83.3%)
Total	30	30	30	30	30

Table III: Distribution and percentage of the responses on variable *testP*.

xvi) *Do you use non-APIs?*: 21 (70%) of the 30 responded that they use non-APIs while the remaining 9 (30%) responded that they do not use non-APIs.

xvii) *If answer was "NO" in (xvi). Do you deliberately avoid using Eclipse non-APIs?*: 9 respondents answered this question, where 8 (88.9%) of the 9 responded that they *deliberately avoid using* non-APIs and 1 (11.1%) responded that he/she does *not deliberately avoid using* non-APIs.

xviii) *If answer was "YES" in (xvi). Do you deliberately use the non-APIs?*: 21 respondents answered this question, where 18 (85.7%) of the 21 responded that they *deliberately use* non-APIs and the remaining 3 (14.3%) do not deliberately use non-APIs.

5) Testing of Eclipse Product: The following list presents the frequency and percentage distributions of the cases in the variables that belong to this category.

xix) *On which Eclipse SDK release did you develop the first version of your Eclipse product?*: 1 (3.3%) of the 30 developed the first Eclipse product on SDK 1.0, 4 (13.3%) on SDK 2.0, 1 (6.5%) on SDK 2.1, 7 (23.3%) on SDK 3.0, 2 (6.7%) on SDK 3.1, 4 (13.3%) on SDK 3.2, 2 (6.7%) on SDK 3.3, 3 (10%) on SDK 3.4, 3 (10%) on SDK 3.5, and 1 (3.3%), each, on SDKs 3.6, 3.7, and 4.0m4.

xx) *Have you tested your Eclipse product on newer Eclipse SDK releases, betas or milestones?*: had multiple choice options where one was allowed to select all that applied. We have subdivided the variable into five sub-variables according to the choices. The sub-variables are *nxrRL* for option *Yes, on the next new release*, *rlANA* for option *Yes, on a release after the next or later*, *nwMV* for the option *Yes, on a new milestone release*, *nwBV* for the option *Yes, on a new beta version*, and *NO* for the option *NO*. Table III show the distribution and percentage of the sub-variables.

xxi) *If answer was "YES, the on next release and/or on a release after the next or later. Are you frequently testing you Eclipse product on new SDK releases?*: 23 respondents answered this question, where 12 (52.2%) of the 23 responded that *YES* they frequently test their Eclipse product on new SDK releases and 11 (47.8%) responded *NO*.

Var	Correlation (Significance)									
	educa (i)	expsd (ii)	expjd (iii)	exped (iv)	tsize (viii)	upimp (ix)	NOV (x)	NOF (xi)	awAPg (xiii)	flAPg (xiv)
educa	–	.23(.11)	.22(.13)	**.32(.04)***	-.04(.42)	.02(.46)	.07(.35)	.04(.41)	**.39(.02)***	.28(.07)
expsd	.23(.11)	–	**.67(.00)****	**.72(.00)*****	.20(.14)	.25(.09)	.03(.43)	.20(.15)	.17(.18)	.21(.14)
expjd	.22(.04)	**.67(.00)****	–	**.72(.00)*****	.00(.48)	.07(.36)	-.06(.37)	.22(.13)	.11(.28)	.15(.22)
exped	**.32(.04)***	**.72(.00)*****	**.72(.00)*****	–	.19(.16)	**.40(.02)****	.13(.24)	**.38(.02)***	**.37(.02)***	**.32(.04)***
tsize	-.04(.42)	.20(.14)	.00(.49)	.19(.16)	–	.04(.41)	.05(.40)	**.62(.00)****	.09(.32)	.19(.15)
upimp	.02(.46)	.25(.09)	.07(.36)	**.40(.02)****	.04(.41)	–	**.37(.02)***	.20(.15)	.15(.22)	.18(.17)
NOV	.07(.42)	.03(.43)	-.06(.37)	.13(.24)	.54(.40)	**.37(.02)***	–	**.46(.01)****	.13(.24)	.14(.23)
NOF	.04(.42)	.20(.15)	.22(.13)	**.38(.02)***	**.62(.00)****	.20(.15)	**.46(.01)****	–	.15(.21)	.21(.13)
awAPg	**.39(.02)***	.17(.18)	.11(.28)	**.37(.02)***	.08(.33)	.15(.22)	.13(.24)	.15(.21)	–	**.88(.00)*****
flAPg	.29(.07)	.21(.14)	.15(.22)	**.32(.04)***	.18(.17)	.19(.15)	.18(.17)	.14(.23)	**.88(.00)*****	–
mean	3.910	5.290	4.690	3.430	2.200	2.260	3.400	2.630	1.510	2.060
std	1.292	1.250	1.409	1.290	1.232	0.741	1.818	1.239	0.507	0.998

Table IV: Matrix of observed Spearman's correlation coefficients and their significance. The values in bold indicate correlations that are significant at 95% confidence interval. By the rule-of-thumb, correlations of .70 and higher are considered very strong (***), .40 to .69—strong (**), .30 to .39—moderate (*), .20 to.29—weak, and .01 to .19—No or negligible relationship.

C. Quantitative Analysis of the Variables

In this section we present the quantitative analysis of the survey. First, we present the observed correlation between the answers of the pairs of questions whose answers can be expresses on the linear scale. From Table I, only 12 of the 21 questions have answers that can be expressed on a linear scale. Variable names in Table I will represent the questions.

Table IV presents a matrix of observed correlation coefficients and their significance. Variables *whours* and and *awAnNA* are not included in Table IV since they do not correlate with any of the variables. Correlations that are significant at 95% confidence interval are bold-faced.

1) RQ1—Relationship between identified factors: : For this research question, we discuss only significant correlations between pairs of variables related to use-of-Eclipse-interfaces. From Table IV, we observe the significant correlations between the following pairs of variables:

- *educa vs awAPg (.39*)*: We observe a moderate positive relationship between education and awareness of the existence of the Eclipse provisional API guidelines. This implies that the more educated a developer is, the more likely he/she will try to find guidelines/manuals. However, when we look at the results of years of education in Table II, we observe 29 of the 30 have 4 and more years after secondary school. If we relate the years to level of education, then, $\geq$4 & $<$6, is equivalent to masters degree. The results of years of education therefore imply that 96.7% of the respondents have a high level of education. The results of the relationship are surprising as they imply that developers with higher level of eduction (beyond masters degree) are the ones who take time to find and read the manuals/guidelines.
- *educa vs exped (.32*)*: We observe a moderate positive relationship between education and experience as an Eclipse product developer. This implies that developers with many years of experience are also more educated. However, we were not able to find a plausible explanation of the observation.
- *exped vs upimp (.40**)*:We observe a strong positive relationship between experience as an Eclipse product developer and importance of updating the Eclipse products with new releases of the Eclipse SDK APIs. This implies that experienced Eclipse product developers value updating their products with new SDKs.
- *exped vs NOF (.38*)*: We observe a moderate positive relationship between experienced product developers and NOF. This implies that more the experience of the developers, the bigger their Eclipse products will be.
- *exped vs awAPg (.37*)*: we observe a moderate positive relationship between experience of Eclipse product developers with awareness of the "Eclipse provisional API guidelines". This implies that less experience product developers are not aware of the guidelines.
- *tsize vs NOF (.62**)*: We observe a strong positive relationship between size of the development team and NOF. This implies that the more the number people involved in developing the Eclipse product, the bigger their products.
- *upimp vs NOV (.37*)*: We observe a moderate positive relationship between importance of updating and NOV. This implies that developers who value the importance of updating their Eclipse products with new SDK interfaces, have released many versions of their products.
- *NOV vs NOF (.46**)*: we observe a strong positive relationship between NOV and NOF. This implies that a product with many versions have a lot of functionality in the later release of the product.
- *awAPg vs flAPg (.88***)*: We observe a very strong positive relationship between the awareness of the existence of the Eclipse provisional guidelines and following the guidelines. This implies that developers who are aware of the guidelines do follow them. This finding is concurrent with our earlier observation that most students adhere to explicit guidelines [9].

From the discussion above in the relationship between the pairs of variables, we make two main observations: First, we

observe that experience of an Eclipse product developer correlates with 7 out of 10 variables as can be seen in Table IV. This implies that the experience of the Eclipse developer plays an important role in development/maintenance of the Eclipse products. One very important pair of correlation is *exped vs awAPg (.37*)*. Second, in Section III-B we reported that 50% of the respondents were not aware of the Eclipse API provisional guidelines. We now observe that level of education correlates strongly with awareness of the guidelines. Companies/organizations involved in developing Eclipse product and in general software development, should encourage developers to read product manuals/guidelines.

2) Exploratory Factor Analysis: For research questions RQ2 and RQ3, we employ exploratory factor analysis [10] to help us analyze the differences between the two groups of respondents in each of the research questions, respectively. Each of the research questions can be analyzed on the 10 variables whose answers are represented on a linear scale. Exploratory factor analysis helps in grouping the large number of variables into a small number of independent factors, i.e., groups of variables that are correlated with each other, on which each of the two identified group for research questions RQ2 and RQ3, respectively, can be compared. Exploratory factor analysis is a statistical method for investigating common but unobserved sources of influence (factors) in a collection of variables [11].

To perform factor analysis, we used software IBM SPSS Statistics 20. Principal component analysis (PCA) was used to extract the factors from the correlation matrix [10]. Equation 1 represents the relationship between the extracted factors and a given variable, where, *var* is a variable, β_1,...,β_N are the coefficients, also called *factor loadings*, for *factor1*,...,*factorN*, and U_{var} is the error. In addition, we used *Varimax* rotation method to provide us with a *simple structure* of the data. In a simple structure, each factor has large loadings in absolute value for only one of the variables.

After extracting the factors, to determine the number of factors to retain, we used the eigenvalue-greater-than-one criterion [10]. Eigenvalues explain the variance of the factors. We retained three factors that had eigenvalues >1. The final percentage of variance explained by the 3 factors retained is 71% of the total variance.

$$var = \beta_1(factor1) + ... + \beta_N(factorN) + U_{var} \quad (1)$$

Table V presents the communalities of nine of the variables (reasons for excluding variable *upimp* (ix) will be explained later). *Communality* of a variable is the proportion of variance explained by the common factors [10]. Communality ranges from 0 to 1, with 0 indicating that the common factors do not explain any of the variance, and 1 indicating that all the variance is explained by the common factors. Table VI presents the rotated matrix between the factors and the variables after the factors have been sorted by the absolute values of loading.

Variable	Initial	Extraction
educa (i)	1.000	.426
expsd (ii)	1.000	.746
expjd (iii)	1.000	**.825**
exped (iv)	1.000	.775
tsize (viii)	1.000	.563
NOV (x)	1.000	.374
NOF (xi)	1.000	**.874**
awAPg (xiii)	1.000	**.919**
flAPg (xiv)	1.000	**.836**

Table V: Communalities using the principle component analysis extraction method

	Factor		
Variable	1	2	3
expjd (iii)	**.904**	.063	.054
expsd (ii)	**.855**	.118	-.028
exped (iv)	**.807**	.265	.233
awAPg (xiii)	.070	**.950**	.110
flAPg (xiv)	.103	**.889**	.188
educa (i)	.328	.552	-.128
NOF (xi)	.170	-.033	**.919**
tsize (viii)	.072	-.044	.746
NOV (x)	-.049	.158	.589

Table VI: Rotated factor matrix showing factor loadings on the variables. Extraction method–PCA, Rotation method–Varimax with Kaiser normalization.

As opposed to our small sample size of 30 subjects, the rule-of-thumb requires at least 100 subjects to perform factor analysis. Costello and Osborne [12] report that a large percentage of researchers report factor analysis using relatively small samples and that strict rules of sample size of exploratory factor analysis have disappeared. "The stronger the data, the smaller the sample can be for an accurate analysis. *Stronger data* in factor analysis means uniformly high communalities without *cross loading* (variables having high loadings in more than one factor), plus several variables loading strongly on each factor" [12]. Communalities of .80 or higher are considered high [13] but uncommon with real data. More common magnitudes are low to moderate communalities of .40 to .70 [12]. Tabachnick and Fidell [14] report that .32 as a good rule-of-thumb for minimum loading of a variable. The authors further report that, researchers need to decide whether to drop the variable with low loading. In our analysis, the communality of the variable *upimp* (ix) was .269, the reason we decided to exclude it. Tabachnick and Fidell [14] further report that a cross loading variable is a variable that loads at .32 or higher on two or more factors. The researcher need to decide whether a cross loading variable should be dropped from the analysis.

From Table V, we can observe that the variable communalities are all above .32 and four of the communalities are high (ones in bold). Furthermore, from Table VI we can observe that only *eduac* (i) variable has cross loading on both *Factor 1* and *Factor 2*. The rest of the variables have uniformly high factor loadings on one factor and very low loadings on the remaining two factors. We decided not to drop the cross loaded variable *eduac* (i) in our analysis since

we have few number of variables. From the observations, we can say that our data is relatively strong and therefore our small sample size of 30 subjects does not affect the accuracy of the results.

Looking at the factor loadings of the variables in Table VI, we can observe that factor 1 has high loadings (shaded in gray) on variables *expjd* (iii), *expsd* (ii), *exped* (iv) and *educa* (i), factor 2 has high loadings on variables *awAPg* (xiii), *flAPg* (xiv), and *educa* (i), and factor 3 has high loadings on *latvn* (xi), *tsize* (viii), and *verep* (x). Interpreting the factors based on the natural selection of variable loadings we could say that the conceptual meaning of factor 1 is related to *education and experience*, factor 2 is related the *education and API guideline awareness*, and factor 3 is related to *size of the plug-in*.

With the small number of factors that explain the collection of variables, we can now compare the two groups of respondents that use or do not use non-APIs. Recall in Section III-B we stated that 9 of the 30 respondents do not use non-APIs and the remaining 21 use non-APIs.

During factor extraction, we saved factors scores for each of the subjects using the regression method [10]. We will use the factor scores in analysing research questions RQ2 and RQ3. To test for normality, we used *one-sample Kolmogorov-Smirnov* test at 95% confidence interval. We grouped the factor scores independently according the groups identified, i.e. those that do not use non-APIs and those that use non-APIs. Since the p values associated with the test are all above 0.32, we could not reject the normality hypothesis.

3) RQ2—Differences between Eclipse products that use and do not use non-APIs: : We will start by stating the hypotheses for this research question:

- H_0: *Developers who use non-APIs and those who do not use non-APIs have the same average values on each of the identified factors.*
- H_a: *There is a difference on the average values of the identified factors between the developers that use and do not use non-APIs.*

Since we could not reject the normality hypothesis, we use the traditional t-test and the traditional 95% confidence level to investigate H_0. The test reveals significant difference between developers who use and do not use non-APIs on factors *Education and experience* and *Size of the plug-in* (p-values equal 0.003 and 0.026, respectively). However, developers who use and do not use non-APIs have no significant difference when it comes to *education and API guideline awareness* factor (p-value 0.702). Therefore, we reject the H_0 for *Education and experience* and *Size of the plug-in* and accept H_a, however, we do not have enough evidence to accept H_a for *education and API guideline awareness*.

Furthermore, to identify how different the two groups are, we used the *gamma measure of association* [15] to quantify the relationship between the variables in the significant factors and the two groups. Since the gamma measure of association can only be applied to test the relationship between two variables on an ordinal scale, we impose an ordinal scale, as suggested in [15], on the two of respondents, i.e., those who do not use non-APIs (1) and those who use non-APIs (2).

Correlation (Significance)						
educa	expsd	expjd	exped	NOF	NOV	tsize
-.0(.78)	**.71(01)**	**.66(.02)**	**.77(00)**	**.68(00)**	.19(.48)	**.47(.08)**

Table VII: gamma correlation coefficients between the variables and the two groups of respondents that use and do not use non-APIs.

The results in Table VII reveal that compared to developers who do not use non-APIs, the developers who use non-APIs have more years of experience as software developers, Java developers and Eclipse developers, the latest versions of their Eclipse products have more NOF and their Eclipse development team size is higher.

In our previous study [2], [3], we found that Eclipse products that depend on at least one non-API are larger in terms of NOF and also have more NOV compared to Eclipse products that depend on only API. The current study confirm the previous findings on NOF but we do not have sufficient evidence for the differences on NOV for the two groups of Eclipse products. The possible reason for the findings on NOV is that, NOV is dependant on the Eclipse SDK on which the initial version of the product was developed. Recall that in Section III-B we reported that the first versions of the 30 Eclipse products were developed on 12 Eclipse SDK releases, where SDK 3.0 had the highest number of 7 products. The distribution of the 30 Eclipse products on the 12 SDK reduces the number of data points to statistically test NOV for the two groups of software systems.

The findings of research question RQ2 now reveals an answer for previously unanswered question in [2], of "why ETPs that depend on at least one non-API are larger than those that depend on only APIs?". The reasons for the difference in size between the two groups of ETPs are related to two main human factors: 1) developers of the ETPs that use at least one non-API are more experienced as software/Java/Eclipse products developers, and 2) the Eclipse development team sizes are larger.

4) RQ3—Differences between open-source and commercial Eclipse products: : We will also first state hypotheses for this research question:

- H_0: *Developers of open-source and commercial Eclipse products have the same average values on each of the identified factors.*
- H_a: *There is a difference on the average values of the identified factors between the developers of open-source and commercial eclipse products.*

#	Question
ix(a)	Please give reasons for your choice of importance of updating your Eclipse product
xiv(a)	If answer in (xiv) was "Sometimes" or "Never" follow guidelines. Why don't you always follow the guidelines?
xvii(a)	If answer was "NO" in (xvi) and answer was "YES" in (xvii). Please give an explanation of why you deliberately avoid the use of Eclipse non-APIs.
xviii(a)	If answer was "YES" in (xvi) and answer was "YES" in (xviii). Please give an explanation of why you deliberately use Eclipse non-APIs.
xxi(a)	If answer was not "NO" in (xx). Please name some of the challenges you have faced in testing your product with newer Eclipse SDK releases.

Table VIII: Open-ended questions. Follow-up questions in Table I for the corresponding question numbers without the lower-case letters in brackets.

By performing three t-tests we have observed that at 95% confidence interval, there is no significant difference between developers of open-source and commercial Eclipse products (p-values always exceeded 0.1). We therefore accept the H_0 that developers of open-source and commercial Eclipse products have the same mean values on each of the identified factors.

The findings of research question RQ3 clear our doubts on the generalizability of the results of our previous studies in [2], [3], [4], [5], that there is no significant difference open-source and commercial Eclipse products in terms of the factors we have considered.

D. Qualitative Analysis

To complement the quantitative analysis presented so far, we sought the opinions of the developers by asking the open-ended questions. Table VIII contains the questions that we asked in the survey. We analyze the answers questions qualitatively. Due to space limitations, we only discuss the common responses. Interested readers can refer to the survey for the rest of the answers [8].

1) Reasons for the developers choice of importance (question ix(a) in Table VIII): The answers were given according to the respondents' chosen option on how they value the importance of updating their Eclipse products, i.e., *not important*, *moderately important* or *very important*. Recall that in research question RQ1 (Section III-C), we discussed the relationship between experience of Eclipse developers and importance of updating, i.e., *expev vs upimp (0.4**)*. This means that the respondents whose answers correspond to *very important*, are more likely to be experienced Eclipse product developers.

After examining the common answers grouped according to the respondents value importance of updating their product with new versions of Eclipse SDK, we observed that the reasons are based on the following:

- For those who chose the option *not important* and *moderately important*, the reasons are related to: 1) *use of basic APIs* that do not change very often, 2) *product finished* and no longer maintained, 3) taking care of *product functionality first* before they can update with Eclipse, and 4) new versions of the product have to *keep working with older versions* as well.
- For those who chose the option *very important*, the reasons are related to: 1) users of the product require compatibility with the latest release of the SDK, 2) getting new features, 3) benefiting from simplified, improved functionality and performance enhancement of the evolved interfaces, 4) keeping up with the Eclipse release train to survive in time.

2) Reasons why developer do not always follow the guidelines (question xiv(a) in Table VIII): The answers were given according to the respondents chosen option on whether they follow Eclipse API provisional guidelines, i.e., *never follow* or *sometimes follow*.

After examining the common answers grouped according to whether respondents never or sometimes follow the guidelines, we observed that the reasons were based on the following:

- For those who chose the option *never follow*, the reason is that they know that they are using unstable interfaces, but they are willing to update the code in new releases of the SDK.
- For those who chose the option *sometimes follow*, the reasons are related to, 1) sometimes they require functionality from non-APIs when they cannot find the functionality in the APIs, and 2) they use some useful non-APIs and try to get Eclipse org make them public.

3) Reasons why developers deliberately avoid the use of non-APIs (question xvii(a) in Table VIII): The respondents that answered the question the gave reasons for deliberately avoiding the non-APIs that were related to, knowing/assuming that the non-APIs are unstable.

Recall that in during the discussion of research question RQ2 we found that the respondents of this question are less experienced Eclipse product developers and their products are small in terms of NOF. It is possible, for these developers, that the benefits of using non-APIs are overshadowed by the instability of the non-APIs or the the developers do not require extensive functionality yet since their software products are still relatively small.

4) Reasons why developers deliberately use non-APIs (question xviii(a) in Table VIII): The common answers to the question are as follows: 1) There is no API with necessary functionality whereas, re-implementation (copy & paste) of the functionality provided by the non-API would be more difficult to maintain. Re-implemented code misses out on improved quality of the non-API in the new releases of Eclipse. 2) Avoiding reinventing the wheel and avoiding possible pollution of my code (copying in Eclipse public

licence (EPL) code, vs linking, can be very problematic). Old non-APIs are unlikely to disappear. 3) We use the non-APIs so as to expose them.

Recall that during the discussion of research question RQ2 we found that the respondents of this question are more experienced Eclipse product developers and their products are larger in terms NOF. This could possibly mean that with experience one gets to uncover the benefits of using the non-APIs that can be more important than their instability.

5) Challenges faced by developers in testing their products with newer Eclipse SDK releases (question xxi(a) in Table VIII): The challenges that developers face in testing their products in new SDK releases are related to: 1) drastic changes in Eclipse leading to the need to learn a large set of APIs to make the product function properly again (especially e4 APIs are a prime example), 2) a full regression testing required, and 3) manual testing of the product.

IV. Threats

In this section, we discuss validity and reliability threats that may have affected our study our study.

A. Threats to Validity

As any other survey investigation, our findings may subject to validity threats. We categorize the possible threats into construct, internal, and external validity.

Construct validity threats in our study mainly concern how the measurements were performed in the study. In particular, how questions were asked, the way they could be answered (e.g. open vs. closed questions), the scales used to codify the answers. As we discussed in Section II, as well as our experience in our previous empirical research about Eclipse API usage, we took time to design the questionnaire. Recall that in Section II, the questionnaire was reviewed internally and externally to avoid ambiguous and biased questions. The way the survey was executed poses a threat of respondent selection bias. Since survey was deployed on the Eclipse twitter account, the survey could have missed out Eclipse product developers that do not follow the Eclipse twitter account and those not registered on twitter. Construct validity of our findings can also be threatened if a respondent filled in the questionnaire more than once. We tried to minimize this threat during the design of the questionnaire by asking a compulsory question of the name the Eclipse product the respondents are involved in. All respondents developed different plug-ins. To protect anonymity of the respondents, the plug-in names and contact addresses in the survey data we shared [8] were excluded.

Internal validity threats of our study concern confounding factors that may affect the outcomes of the results and difficult to control. Confounding factors like human memory, knowledge, motivation, personality and the fact that people want to be seen in a good light are possible threats to internal validity with questionnaires [16], [17]. It is impossible to know whether the respondents answer truthfully, or whether other effects like pressure of the projects they are working on affect their response. To limit these threats, we tried to motivate the respondents by explaining the importance of the survey where we also promised to reveal the results of the survey including the data we collected. Furthermore, to minimize this threat, we employed both quantitative and qualitative methods and in some cases we employed both methods at the same time.

External validity threats concerns the extent to which our findings can be generalized. Our findings on Eclipse API usage may be threatened by external validity since they are based on only 30 respondents. However, we tried to minimize this threat by the providing a comprehensive questionnaire comprising five sections of different questions. Furthermore, the threat is minimized by the different groups of respondents, for example open-source and commercial developers, developers who use and do not use non-APIs.

B. Threats to Reliability

This section discusses the most relevant threats to reliability using the internal consistency reliability [18]. Internal consistency reliability applies not to one item but a group of items that are thought to measure different aspects of the same concept. Litwin [18] interprets a correlation coefficient $\geq$0.7 to be strong between pairs of the groups of items. In our scenario, internal consistency reliability can be observed between variables that measure experience of the developer, i.e., *expsd*, *expjd*, and *exped*, where we have correlation of around 0.7 between pairs of the variables. Another pair of variables measuring the different aspects of the same concepts are *awAPg* and *flAPg* which measure awareness and following of the Eclipse guideline. Between the pair we have a correlation of 0.88. The high values or correlation reveal how the reliability threat is minimized.

V. Related Work

In the previous sections, we implicitly discussed how the current work relates to our previous work [2], [3], [4], [5]. In general, our previous work is based on empirical analysis of the co-evolution of the Eclipse SDK framework and its third-party plug-ins (ETPs). During the evolution of the framework, we studied how the changes in the Eclipse interfaces used by the ETPs, affect compatibility of the ETPs in forthcoming framework releases. We only used open-source ETPs in the study and the analysis was based on the source code. The current study, based on analysis of the survey, complements our previous studies by including commercial ETPs and taking into account human aspects.

Besides our own work, the current study is related to the study of Oram and Wilson [1], where the authors analysed the difference between open-source and commercial products. No significant across-the-board code quality differences were found between the four systems studied. Similarly

to this work, we also did not find significant differences between open-source and commercial ETPs.

VI. Conclusions

This paper reports on a survey, where 30 Eclipse product developers took part. The aim of the survey was to achieve a clear picture on the state-of-the-practice of the Eclipse interface usage by Eclipse product developers. The lessons learned from the findings of the survey can be used to improve the development and maintenance of Eclipse products.

The results from the analysis reveal a number of findings: First, we observed that the Eclipse product developers' experience plays a very important role in the development and maintenance of the Eclipse products. Experienced Eclipse product developers are more aware of the benefits of updating their Eclipse products with new releases of Eclipse, for example they are aware that evolved Eclipse interfaces are simplified, they have improved functionality and have performance enhancement. Second, we observed that developers with a level of education of up to master degree have a tendency of not reading product manuals/guidelines. Third, we observed that compared to Eclipse products that do not use non-APIs, Eclipse products that use non-APIs are bigger and have larger Eclipse development teams. Furthermore, the developers are more experienced as software developers, as Java developers and as Eclipse product developers. The reasons for using the non-APIs are that while there could be a possibility that instability of the non-APIs overshadows the benefits that the non-APIs offers to less experienced developers, the experienced developers have uncovered benefits of using non-APIs that are more important than the of instability of the non-APIs. For example, when there is no API with the functionality they require, one alternative is to write their own API from scratch. The other simplest alternative is to re-implement (copy & paste) the code of the non-API. The experienced Eclipse product developers state that re-implemented code is difficult to maintain and also misses out on the improvements of the non-APIs in new versions of Eclipse. Moreover, they are aware that if the non-API has been there for years, it is unlikely to disappear. The developers observation of the stability of the old non-APIs coincides with our finding in [5] that old non-APIs relatively more stable than newly introduced non-APIs. Finally, we have observed that there are no significant differences between open-source and commercial Eclipse products in terms of awareness of Eclipse guidelines and interfaces, Eclipse product size and updating of Eclipse product in the new SDK releases.

Acknowledgment

We are very grateful to Koo Rijpkema for guiding us on the statistical analysis, Martijn Klabbers for guiding us in the survey design, and Wayne Beaton who helped us to deploy the survey.

References

[1] A. Oram, G. Wilson, Making software: What Really Works, and Why We Believe it, OReilly Media, Inc., 2010.

[2] J. Businge, A. Serebrenik, M. G. J. van den Brand, Eclipse API usage: the good and the bad, in: SQM, 2012, pp. 54–62.

[3] J. Businge, A. Serebrenik, M. G. J. van den Brand, Survival of Eclipse third-party plug-ins, in: ICSM, 2012, pp. 368–377.

[4] J. Businge, A. Serebrenik, M. G. J. van den Brand, An empirical study of the evolution of Eclipse third-party plug-ins, in: IWPSE-EVOL'10, 2010, pp. 63–72.

[5] J. Businge, A. Serebrenik, M. G. J. van den Brand, Compatibility prediction of Eclipse third-party plug-ins in new Eclipse releases, in: SCAM, 2012, pp. 164–173.

[6] J. des Rivières, How to use the Eclipse API, http://www.eclipse.org/articles/article.php?file=Article-API-Use/index.html, consulted on January 01, 2011 (2001).

[7] Provisional API guidelines (Consulted on January 20, 2011). URL http://wiki.eclipse.org/Provisional_API_Guidelines

[8] http://www.win.tue.nl/~jbusinge/CSMR13.

[9] W. Poncin, A. Serebrenik, M. G. J. van den Brand, Mining student capstone projects with FRASR and ProM, in: OOPSLA Companion, ACM, 2011, pp. 87–96.

[10] M. J. Norušis, SPSS 16.0 Guide to Data Analysis, Prentice Hall Inc., Upper Saddle River, NJ, 2008.

[11] R. Cudeck, Exploratory factor analysis, in: H. E. Tinsley, S. D. Brown (Eds.), Handbook of Applied Multivariate Statistics and Mathematical Modeling, 2000, pp. 265–296.

[12] A. B. Costello, J. W. Osborne, Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis, Practical Assessment, Research and Evaluation 10 (2005) 173–178.

[13] W. F. Velicer, J. L. Fava, Effects of variable and subject sampling on factor pattern recovery, Psychological Methods 3 (2) (1998) 231–251.

[14] B. G. Tabachnick, L. S. Fidell, Using Multivariate Statistics (6th Edition), Prentice Hall, 2012.

[15] A. Agresti, Categorical Data Analysis, J.W.& Sons,Inc., 2002.

[16] C. Robson, Real World Research, Prentice Hall, 2011.

[17] M. Torchiano, M. Di Penta, F. Ricca, A. De Lucia, F. Lanubile, Migration of information systems in the Italian industry: A state of the practice survey, Information and Software Technology 53 (1) (2011) 71–86.

[18] M. S. Litwin, How to Measure Survey Reliability and Validity, SAGE Publications Inc., 1995.