Comparing Web Development Platforms Through the Eyes of Professional Developers

Will Hardy
http://willhardy.com.au/

Technical Report B-07-14
Institut für Informatik
Freie Universität Berlin
http://www.inf.fu-berlin.de/

November 2007

Abstract

Objective comparisons of web development platforms are much needed by development teams. The Plat__Forms experiment sought to provide these objective comparisons, and this survey was designed to provide the subjective context for the experiment's results.

The survey, run in June 2007 asked 268 web developers to directly compare their two best known platforms for 12 aspects of software quality. The results of this survey ideally show where the objective results of the Plat__Forms research agree with the current opinions of web developers, and where they disagree. In addition to the 3 platforms measured in the Plat__Forms research (Java, Perl, PHP), Ruby and Python developers were also involved.

Many of the survey results agreed with the objective research, for example PHP was less dependent on tools and frameworks. Some results differed from the Plat__Forms research, for example, PHP was rated poorly for security and robustness.

Contents

List of Tables

List of Figures

1 Introduction

1.1 The Plat_Forms competition

In February 2007 the first Plat__Forms web development contest [1] took place in Nuremberg, Germany. The contest also served as a controlled experiment, where nine professional web development teams realised a web application in 30 hours, in a controlled environment and using the same specification. Three platforms (Java, PHP and Perl) were represented this time around, each by three separate teams.

The goal of the experiment was "to produce solid empirical information that allows for a largely unbiased discussion of the relative pros and cons of different platform technologies with respect to realizing modest interactive web-based applications."[1] This is a particularly ambitious goal, and certainly a worthwhile cause, given the general lack of objective analysis on web development platforms and the importance of web development today.

To complement the *objective* evidence collected during the Plat__Forms competition, we are interested in learning about the personal experiences of experienced web developers. Thus, a survey was conceived to collect *subjective* evidence from the broader developer community.

In addition to our involvement in the Plat__Forms contest, this survey forms part of our assessment for the subject *Empirische Bewertung in der Informatik* (Empirical Analysis in Computer Science) at the Free University of Berlin.

1.2 Survey overview and goal

In line with the goals of the Plat__Forms contest, the survey aims to measure the underlying pros and cons of a given platform, this time, through the experiences of a large number of web development professionals. The survey also aims to identify any underlying prejudices in the web developer community, that may exaggerate or invent differences between the platforms.

As was the case with the contest, the survey approaches platform comparison through several established characteristics of software including robustness, readability, maintainability, usability and resource requirements.

1.3 Sociological research

Although the question of prejudices in the web developer community is particularly interesting, we are not in a position to perform sociological research. Given the time and resource constraints imposed on us by the subject assessment, we could only choose an online self-selection survey. By doing this, we cannot achieve a typical cross-section sample

of the general population, and any underlying prejudices we see may simply be a phenomenon, unique and local to the group of developers who were exposed to the survey. For example, Perl, PHP and Python developers do not make up the general developer population, and they may have different prejudices as opposed to .NET or Java developers.

However, any prejudices that underlie this group of developers are still of interest to us, especially if they differ markedly from the observed results of the Plat_Forms contest.

1.4 Collecting experiences

The central input of this survey is the collective experiences of a large number of web developers. This means, that each participant will need to have recallable experience and any comparisons we make need to be closely related to these experiences. Given this constraint, our approach was to have the participant draw comparisons from their own experience, between two platforms for a range of relevant software attributes. We are then in the position to summarise these comparisons.

2 Method

2.1 Survey formulation

Each participant was asked to pick two well known platforms, and make a series of comparisons, each focusing on a given software characteristic. Each comparison would ideally be made as a result of the participant's particular experiences. Twelve software characteristics were chosen to be compared, balancing depth with survey duration. Depth would provide us with more insight, and a short survey would typically provide us with more participants. It was estimated that the twelve comparisons would take around 10 to 15 minutes, which should be short enough to attract at least 100 responses.

Within the survey was also a series of demographic questions, which aimed to establish the quality of the comparisons being made. We used amount experience, programming capability and nature of experience to assess the quality of the comparisons, of course this is not perfect: Refer to section 3.1 on page 11 for a more detailed discussion on this.

The complete questionnaire can be found in Appendix C on page 75.

2.2 Target participants

Our target group was web development professionals, people who have experience working on serious projects. As our survey asks for concrete comparisons between platforms, we needed to ensure that all participants had adequate experience in at least two platforms. If this were not the case, we could not be sure which comparisons were based on actual experience, and which ones were 'educated guesses'. This most certainly reduces the number of potential participants, but the extra participants would not have been likely to be able to reliably compare two different platforms.

Four demographic questions[1] were created to help validate whether a given participant belonged to our target group. Participants were asked to detail their level of developer experience, assess their expertise and provide other information, such as number of programming languages used and tried, which can be used as a rough indicator of programming ability. In addition to that, the nature of their development experience (whether it is professional or hobby based) and nature of their recent involvement (amount of technical software development activities in the last year) was also probed. Participants were also asked to detail their experience with each platform, and their comparative knowledge of their selected platforms.

Isolating the comparisons means that it is unimportant how many representatives we have from each platform. If one platform is underrepresented, it can be safely ignored, and its absence does not affect the other comparisons made. With this in mind, we can safely look for as many participants as possible, without the concern of over-representation of one platform.

[1] These were questions 2,3,18 and 19.

2.3 Survey tools

None of the freely available off-the-shelf survey tools were suitable for our needs (we needed variable questions), so a custom application was written in-house using the *Django* web framework.[2] This framework was chosen because we were already familiar with it (no training costs) and it was more or less suitable for the task. The server was hosted at the university under the `plat-forms.org` domain, to help maintain a professional image and make use of the Plat__Forms brand.

2.4 Publicity

For contacting potential participants we primarily used development mailing lists, which offer access to a large number of professional developers, particularly within the open source communities. We also made use of Forums and public bookmarking sites (such as Reddit, Slashdot and Digg), which was necessary to reach the Java and .NET communities. Some personal contacts in the Perl and PHP community helped out with publicity, and we also provided pre-packaged HTML and a button for bloggers to link to our survey. For details of all known publicity see Appendix E on page 85.

The survey was open for 12 days, from Tuesday, 26 June 2007 until Saturday, 7 July 2007.

[2] A modern Python-based web framework, see `http://djangoproject.com/`.

3 Analysis

3.1 Threats to validity

3.1.1 General

Immature "In-House" software

We have written a significant amount of "In-House" software to run both the survey and analysis. Given that there are always bugs present in any non-trivial software implementation, it is highly likely that there are flaws in our code. Any flaws in the survey tool are unlikely to change the results, as this was built on top of an established framework and tested before the survey was run. Two minor implementation flaws were uncovered,[3] however, they did not result in loss of data or misreported answers.

A serious problem would exist if there were flaws in the analysis software (written in R), as this may not be obvious and has a direct influence on the results of our survey. We see this risk as being entirely possible, given that we are only human and that we have little experience developing with R.

3.1.2 Invalid comparisons

Non-professionals

Our goal relies on surveying professional web developers. Being an online survey, we cannot control who participates, and we have little additional information about each participant, other than their responses. There is a very high possibility that non professional developers take part.

We decided to allow anyone to take the survey, and to attempt to identify which participants are professionals. We are confident that the demographic data is sufficient for recognising professionals, and that at most only a small number of participants would have submitted false data. A detailed evaluation of our validation data can be found in section 4 on page 16

[3] For 27 participants, IP addresses were mangled by the Proxy server at the university. Our survey tool however, noted that something was not right, and wrote the correct IP address to a 'notes' field, which was created to recognise errors. This can be manually corrected for the 27 participants it affected. A patch was made, however the production server was not updated, as we had made a decision not to change the survey in any way after it was made public.

A second flaw produced a Server Error (500), when the user typed in an URL by hand, with an invalid page number. The system handled this gracefully, with a polite message, a link back to the home page and sent us an email to let us know. According to the server logs, only one user did this, and appeared to be playing with the survey software, after having completed the survey.

Subjectivity

Given this is a survey, there is of course the problem of subjectivity in the responses of the participants. This is particularly stark, when opinions are entered instead of experience and can be a significant problem to the assertion that our data are based on experience. Looking back at our questions, we can see that they might encourage a participant to ignore experience and voice their opinion,[4] and so this should be kept in mind when reading the results. One would not expect the difference between experience and opinion to be large, however it is nevertheless an issue in the background of our results.

Question misunderstanding

A number of participants voiced concern over some of our questions, in particular, that we ask the participant to compare the usability of software developed with the given platforms. Ideally, the participant would use their experience and rate the usability of software they have developed using each of the platforms. But it is clear this did not occur for some questions. We have tried to identify where this has occurred, and we mention this in our interpretation of our results. More discussion on this issue can be found in section 6 on page 56.

Lack of expertise for a given characteristic

Although most professionals are qualified to make a range of valid comparisons between two platforms, there are no doubt many areas in which they might not have expertise. For example, a programmer that is primarily interested in usability, may not be able to tell which platform is faster, or which platform uses more memory. It was certainly possible to skip questions in the implementation, but few participants did this.

We cannot be sure how much this affects our results, but we will suggest where this might have affected a particular question during the discussion of the results in section 5 on page 28

For our secondary goal of discovering prejudices, this is actually helpful, but certainly not for our primary goal of understanding how the platforms differ.

Validity of original comparisons

Comparisons were originally made between two platforms, it may be artificial to expand this into a general overview. To put it another way, taking two separate comparisons $x < y$ and $y < z$ and making the assertion $x < y < z$, may not be valid if our original comparisons were made by two different people in two different contexts. The original comparisons were certainly valid within the scope of the two platforms involved, but it is possible that expanding those comparisons may stretch the limits of what was intended by the participants who originally made the comparison. Given that we are aggregating many comparisons, any such limited context would need to be universal, but this can be the case when, for example, comparing the IDE tools of .NET and Java on one hand, and the database administration and migration tools of .NET and Ruby: all under the banner of tools. It would

[4] See discussion in 6 on page 56.

not make sense to expand these two comparisons to compare the "tools" of Java and Ruby.

We avoid making such comparisons in this report, but it is important to keep this in mind when interpreting our results.

3.1.3 Bias

Self selection bias

All participants chose to take 15 minutes of their time complete the survey, and so this may represent people who are more opinionated (i.e. have something to say) than those who chose not to participate. If a number of experienced developers believe there is little difference between their platforms, but did not choose to participate, then our results may be exaggerated.

Keeping this in mind, we are more interested in the direction of the comparisons and the relative difference, rather than any absolute scale. If the results are more exaggerated, they should be overall more exaggerated, and relative differences between the platforms' comparisons will still be easy to identify.

Own platform Bias

It is possible that there exists a bias towards the first choice platform over the second choice platform. This would have no effect if the distribution of first to second platforms is even. If it is not, then it may advantage or disadvantage a particular platform.

This is evident in our results for some of the platforms, namely Ruby, Python and PHP. Whether or not such differences stem from bias is impossible to determine. More detail on this can be found in section 4.2.9 on page 25.

Uneven expertise

Some platforms may have more expertise than others. Languages such as Java EE and .NET are typically used in the corporate world, whereas languages such as PHP have a strong base of "amateur" developers. Such differences may result in an imbalance in expertise, which affects the validity of the comparisons.

This issue is discussed in section 4 on page 16 and from this analysis, no widespread differences are obvious.

Zealots and fanatics

Programming languages, given the amount of time invested in training, all have to a certain degree "fans", developers who would like to affirm their choice of development platform. This may hinder objective comparisons, and the results will be disproportionately skewed if this phenomenon is more prevalent in some languages than in others.

We did not have any questions in the survey that attempted to identify this, however we believe that if the amount of experience is even across the platforms, there should not be a substantial imbalance in 'platform-enthusiasm'. The level of experience appears to be even across the platforms (see again section 4 on page 16).

Question suitability

As we are not primarily .NET developers, we did not notice that some questions were not particularly suitable for .NET developers. For example, many questions asked about frameworks, but .NET is its own framework and Ruby only has one popular framework. This only affects 2 questions, and it is discussed during our interpretation of the results.

Secondly "classic ASP" and .NET were grouped together although they are completely different languages. This should be kept in mind when interpreting the results.

3.1.4 Bad submissions

Malicious participation

As with any unsupervised online questionnaire, there is the possibility of deliberate, malicious participation, with the intention of skewing the results for a variety of reasons. We made no efforts to prevent this during the running of the survey, instead, we took the approach of making it possible to recognise any such attempts. To do this, we recorded IP the date, time, IP address, session ID, browser user agent and any IP address changes that occurred during the session.

While it is possible to sidestep all of these measures, it would be time consuming and may lead to unexpected results which would in themselves signal such an event. We are confident that the likeliness that this happened is low.

3.2 Threats to relevance

Irrelevant characteristics

We restricted ourselves to only 12 software characteristics, and there is certainly room to argue that other characteristics may have been more relevant for some. Nevertheless, we are confident that we chose a representative set of criteria.

Selection bias

We chose the announcement channels to our best knowledge. As open source developers use public communication channels (mailing lists, forums, IRC) more often than closed source and commercial developers, we may have only attracted .NET and Java developers who work with open source, and not the typical commercial developer for the respective platforms.

Many industries are also under-represented by an open-source sample, and so this survey may have less relevance for these industries.

Missing platforms

Traditional ASP, Smalltalk, C, PL/SQL all have their advantages and disadvantages. We believe that the platforms we chose form the web development community at large and these other platforms only have negligible market shares in that area. So it may be these platforms provide some qualities not present in any of the platforms included in our survey but we do not believe that we would have found enough participants fluent in these platforms to have enough data to back up any claims.

Geographic concentration

As shown in table Appendix A on page 62, the survey was dominated by participants from Germany, the US and the UK. If the experiences in these countries are not typical for other countries, then some of these comparisons may no longer be relevant. For example, most web developers based in the US may be unaware of character encoding issues, which might typically effect some platforms more than others.

small number of .NET people

With the smaller sample of .NET developers, there is a reasonable chance that it is not representative of the .NET population at large..

monolingual developers

We only surveyed programmers with experience with at least two web development platforms. It is possible that people with experience with one platform have different views of its characteristics. Since our survey only looked at characteristics in comparison to another platform, they would not have added any additional insight. We believe that looking at one platform in isolation does not provide objective results.

4 Participants

This section looks to ascertain whether or not our target participants were found. As we were looking for professional web developers, we need to be able to identify any participants that were not professional web developers, and exclude them.

We also need to ensure that the expertise of the participants was even among the platforms, During the survey, we asked some biographical questions, such as age, experience, capability, percentage of time spent on technical tasks, languages used, languages tried and frameworks tried. All of these metrics are aimed at measuring the expertise of each of the participants, and allows us to observe whether or not this is even.

Throughout our analysis we will only be considering participants who submitted at least one comparison.

4.1 Participant overview

Altogether, 268 participants started the questionnaire. As expected, there was a strong representation from PHP and a somewhat weaker turnout from .NET and Java. Given some enthusiastic promotion from the Perl community, there was a slightly more than expected turnout of Perl developers.

The breakdown of participants who submitted at least one comparison is shown in figure 1.

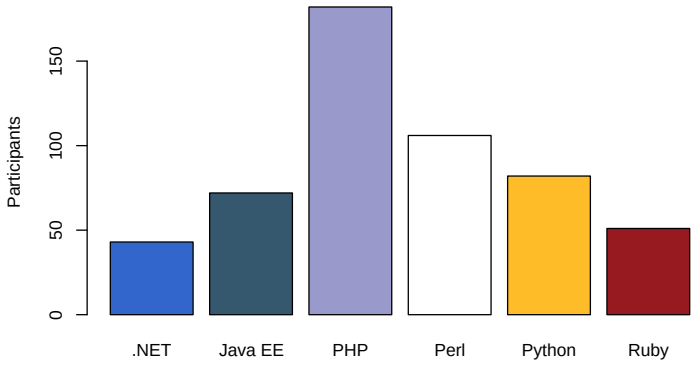

Figure 1: Participants, who submitted at least one comparison by platform

A more important figure is the total number of comparisons made. Each participant could

make up to 12 comparisons, if they completed the questionnaire to the end. Given that a higher percentage of Perl and PHP developers finished the survey, their contribution is exaggerated. The distribution is shown in figure 1 on the preceding page. Although the higher turnout of Perl and PHP developers will strengthen the results involving these platforms, it does not in any way introduce bias, as we will be considering each platform separately.

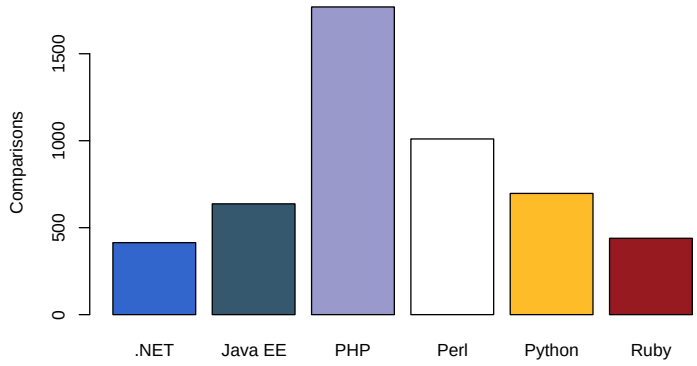

Figure 2: Comparisons by platform

4.2 Validation

4.2.1 Multiple participants

Table 2 on page 61 shows every duplicate participation by IP address. Only one IP address submitted comparisons across two sessions, and although this could indicate two different users sharing the IP address, the second session has nevertheless been removed from the results.

Another 8 participants changed their IP address after beginning the survey, this is however a common phenomenon for people who use portable devices such as laptops. None of these 8 changes occurred within a short time frame from one another and there is no reason to suspect any malice. This had no effect on the collecting of the results, as the IP address was used for bookkeeping purposes, only.

4.2.2 Reading boxplots

The following sections compare the distributions of certain attributes for each of the platforms. This will be done using boxplots, which provide a graphical representation of the

data for each platform. If you are unfamiliar with boxplots, the most important information shown is the arithmetic mean (indicated by "M"), median value (the large black dot) and some percentiles (10, 25, 75 and 90). The dotted line around the arithmetic mean represents one standard error on each side of the mean. The box itself, represents 50% of the participants for that platform, that is, the participants between the 25- and 75-percentile. The extending "whiskers" for each box show the area where 80% of the participants lie.

4.2.3 Age

Figure 3 shows that there are no significant differences in the age distribution of participants between the platforms.

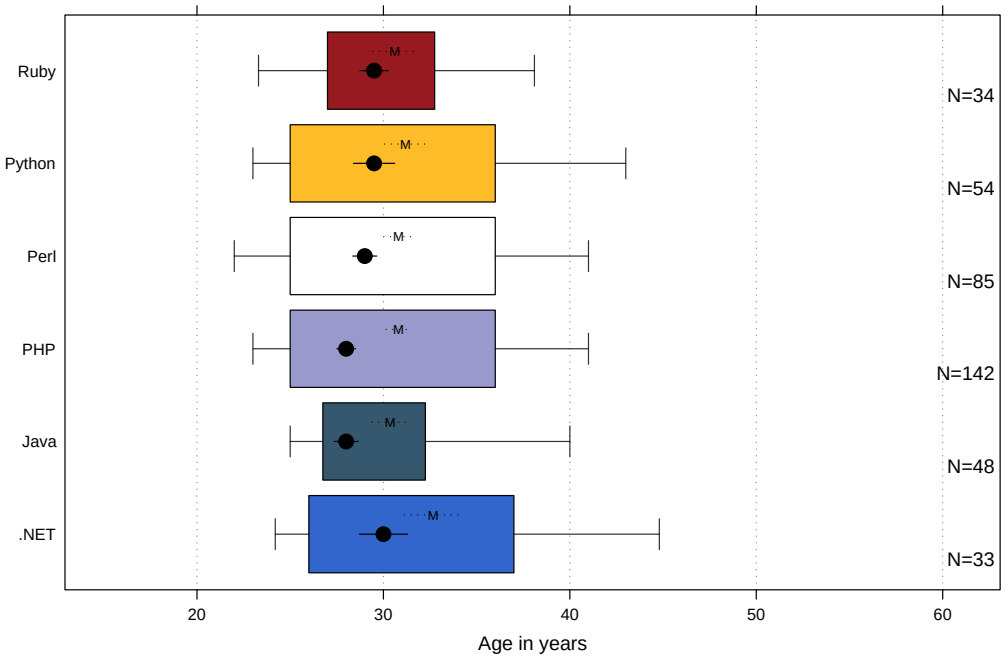

Figure 3: Age by platform

4.2.4 Experience

Figure 4 shows that the experience distribution between the platforms is generally even. Note that this does not imply developer experience on the given platform, as that would render some values impossible for Java. Rather, this is the number of years developer experience from the participant in any language.

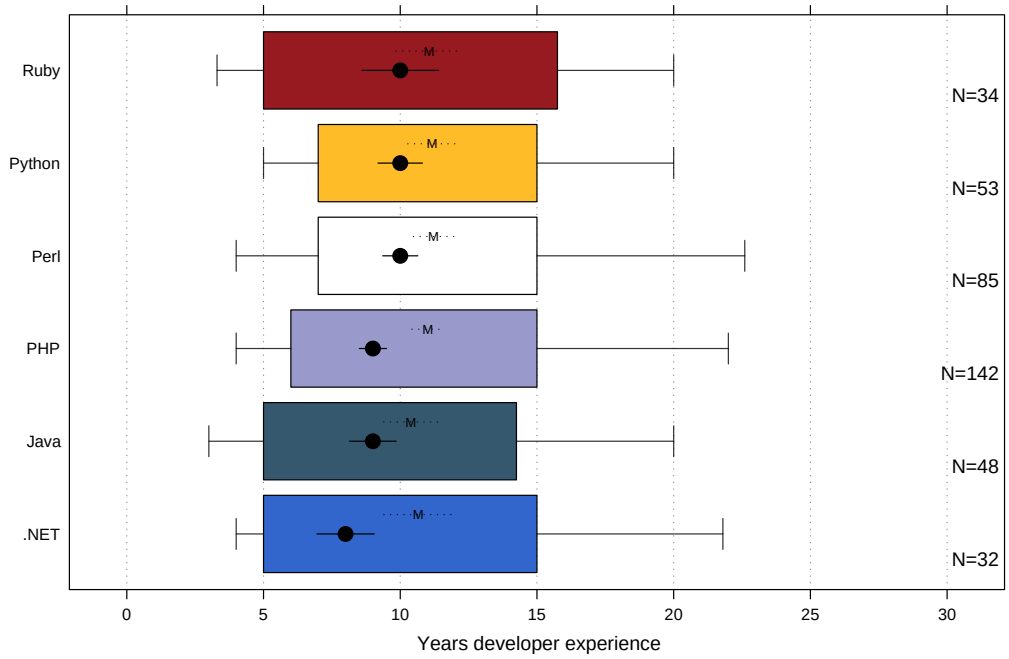

Figure 4: Experience by platform

4.2.5 Capabilities

Figure 5 shows that the capability distribution between the platforms appears to be even. The seemingly dramatic rise displayed for the median for Python and .NET is due to the closed question, which forced the participants to choose from set levels, and does not represent such a dramatic rise in the expected value for these platforms.

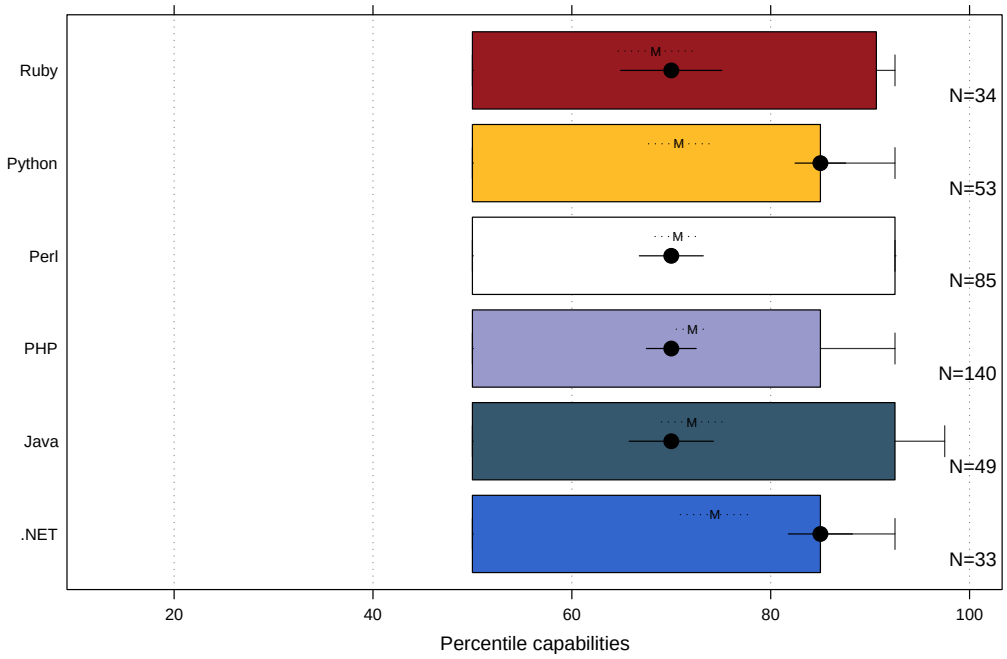

Figure 5: Capabilities by platform

4.2.6 Technical activities

Figure 6 shows the amount of time participants spent on technical activities in the last year. It indicates that the .NET developers have spent slightly more time on technical activities recently, but this difference is not too dramatic. It may be worth keeping this in mind when interpreting the results.

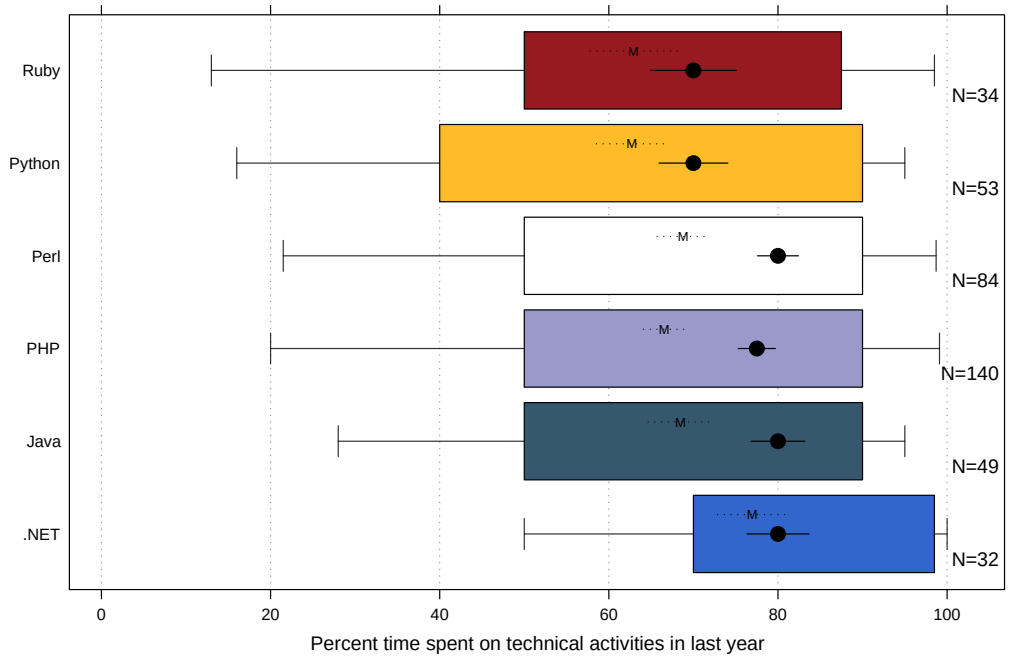

Figure 6: Percent time in last year spent on technical development

4.2.7 Programming languages

Page 19 of the questionnaire asked the participant to list the programming languages they have used and the languages they have tried out. The number of languages a developer has learnt can in some cases be used as a reasonable predictor of performance in software development. As is shown in figures 7 and 8, the participants are sufficiently similar across platforms. Note that this particular metric is roughly calculated, by counting the number of lines or comma separated values in the given free text field.

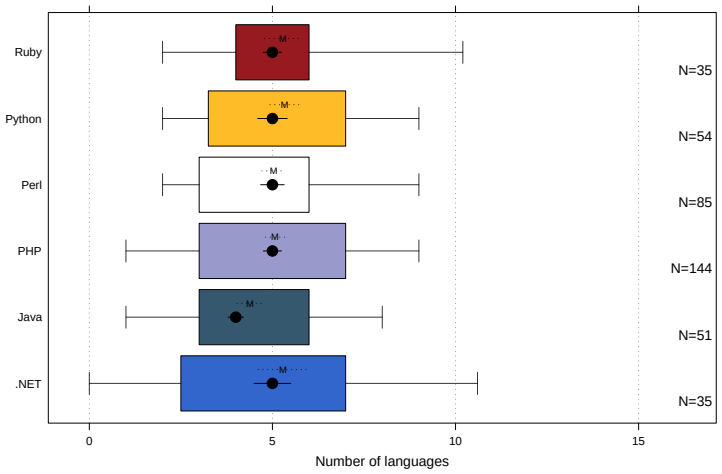

Figure 7: Number of languages used

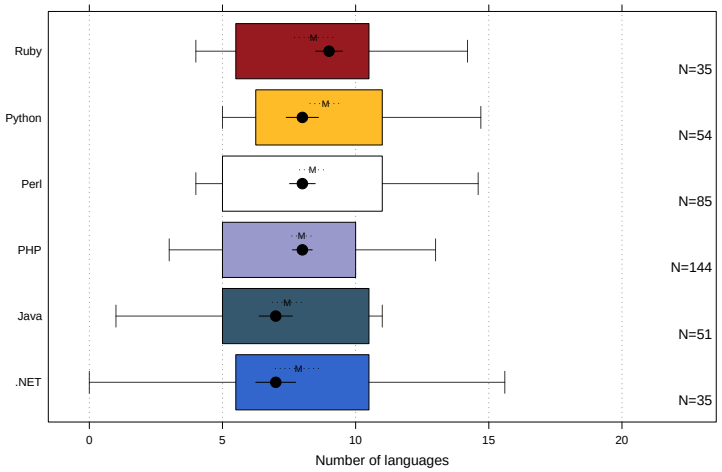

Figure 8: Number of languages

4.2.8 Frameworks

Figure 9 shows the total number of frameworks the participants have used. Once again, this is a relatively even distribution, and there does not appear to be any meaningful differences between the platforms. Note that this particular metric is roughly calculated, by counting the number of lines or comma separated values in the given free text field.

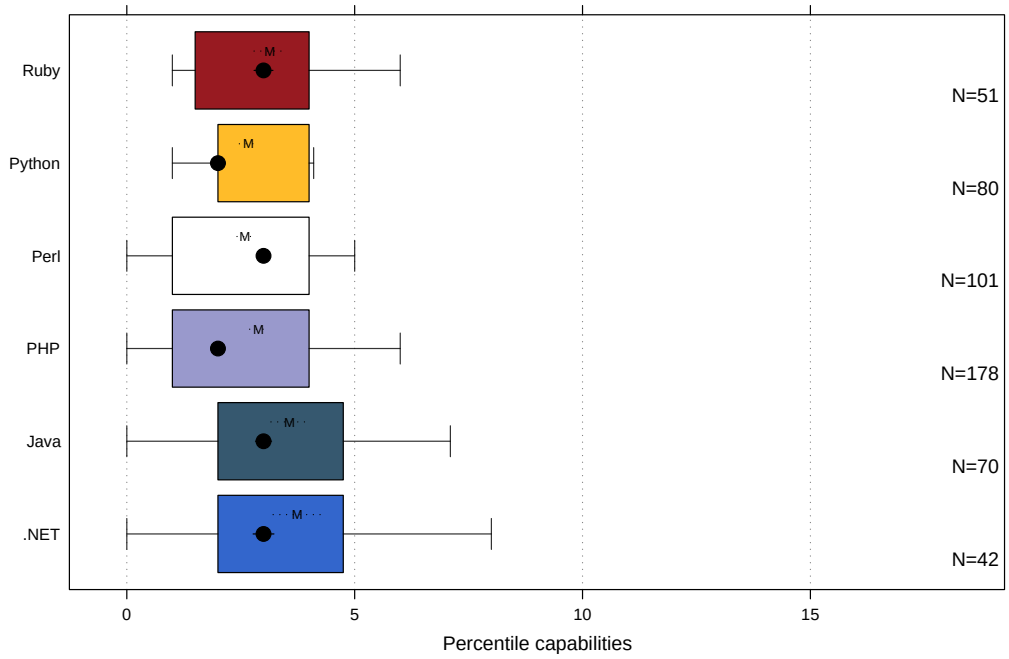

Figure 9: Number of frameworks

All our validation measurements indicate that we succeeded in reaching a balanced sample across platforms. All platforms seem to have a comparable developer structure so answers should be comparable throughout all platforms.

4.2.9 Own platform bias

Figures 10 and 11 provide an insight as to where some platforms exhibit an "own platform bias", that is, the participant rates a platform highly, because it is their most used platform. The charts show the distribution of comparisons made for each platform, using a *small-heavy* weighting system. This weighting system is explained in more detail in section 5 on page 28.

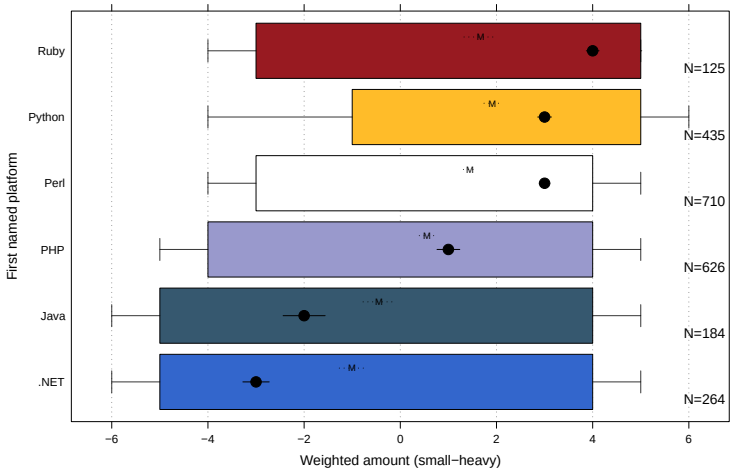

Figure 10: Small-heavy comparisons by first named platform

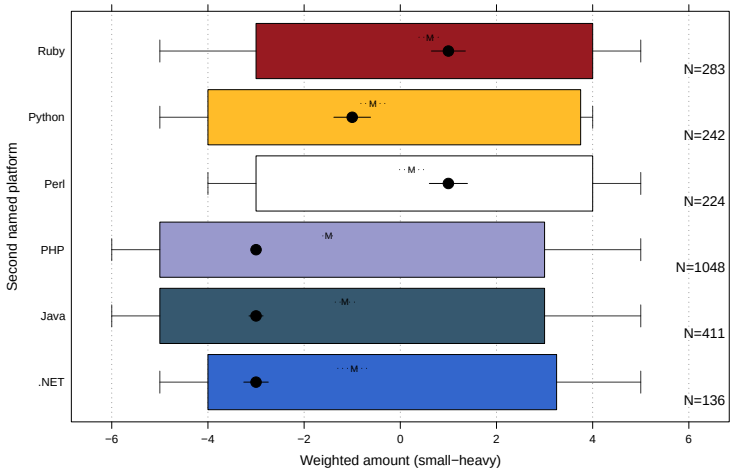

Figure 11: Small-heavy comparisons by second named platform

It appears that there may indeed be an "own platform bias" in our results. In these charts, we see that Ruby, Python and PHP are more highly rated when they are the participants'

primary platform, as opposed to when they were compared by other participants as a secondary platform.

This should be kept in mind when reading the *small-heavy* charts in the results section (section 5 on page 28). The effect of such a bias is unknown, as this bias would typically affect every one of the 12 compared aspects, but the results vary from aspect to aspect (again, see section 5 on page 28 for full results). In addition, the participants were often able to supply textual reasonings behind their choice. These reasonings helped provide a qualitative context for the charted results in section 5 on page 28.

This does not, however apply to the charts displaying the results according to the *large-heavy* weighting (figure 12 on the facing page and figure 13 on the next page). Using a *large-heavy* weighting (see section 5 on page 28 for more information), less bias is noticeable in all platforms except Ruby.

This possible "own platform bias" for Ruby should also be kept in mind, as mentioned earlier for the charts according to the *small-heavy* weighting.

4.3 Origin

Using the IP addresses we can lookup which country the participants completed the survey in. Although this is not 100% fail-safe, given the use of proxies, TOR[5] and international ISPs, it does give a broad sense for where he majority of users are coming from. It is clear from table Appendix A on page 62 that the survey is dominated by participants in Germany, the US and the UK. This may effect the relevance of this survey for developers in other countries, see the discussion on geographic concentration in section 3.2 on page 15.

[5] An anonymity network: `http://tor.eff.org`.

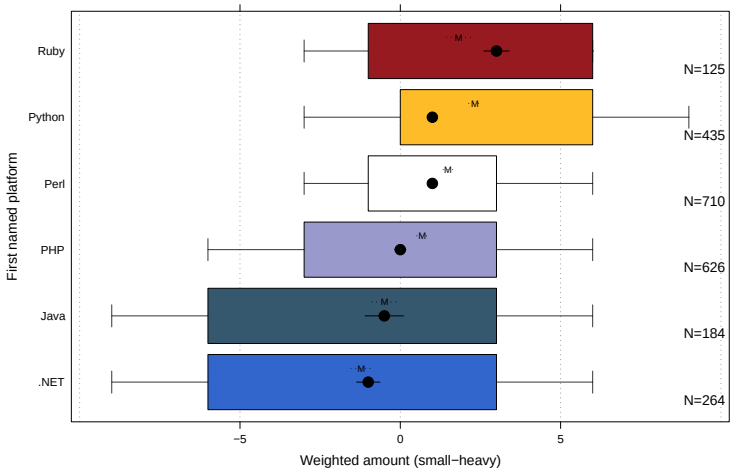

Figure 12: Large-heavy comparisons by first named platform

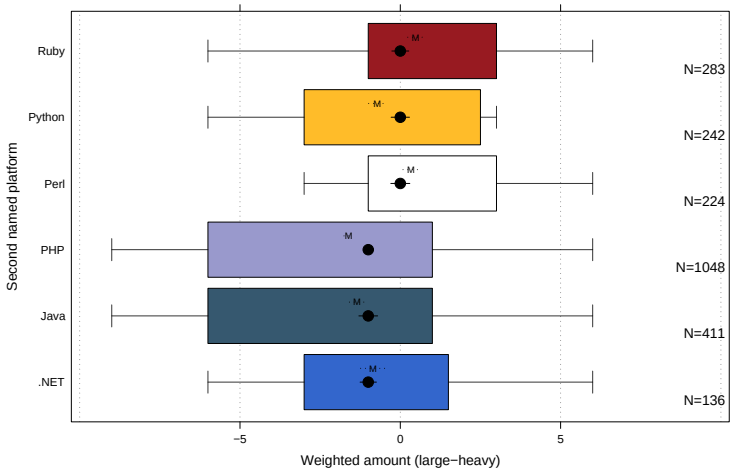

Figure 13: Large-heavy comparisons by second named platform

5 Results

We will now look at the results for each software attribute individually, observing how the participants compared the platforms. Here are, once again, our 12 attributes and where they will be discussed:

For each attribute above, we will highlight any interesting observations. An interesting observation would typically be one of:

- if the participants indicated a *clear difference* between the platforms
- if the participants indicated *very little difference* between the platforms
- if there was *considerable disagreement* amongst the participants

Each comparison submitted by a participant expressed the "difference" between the two platforms in an ordinal scale (see table 1). To help summarise these observations, the values were converted to numeric values, using two different systems. The first system (*small-heavy*) puts emphasis on each value, whereas the second system (*large-heavy*), puts emphasis on larger differences. Using these values, it is possible to calculate an *expected value* and observe the extent of agreement among the participants.

original	*small-heavy*	*large-heavy*
very large	6	9
large	5	6
modest	4	3
small	3	1
about zero	1	0

Table 1: Ordinal scale for differences and corresponding weightings

Using the numerical data we now have, we can work out a confidence interval for the true mean, in our case we have chosen an 80% confidence interval. This interval is calculated using the `t.test` function in R.

We can plot these confidence intervals graphically, in order to gain an overview of the complete set of comparisons for each given software attribute. An example is presented below to help understand how to read our plots.

5.1 Example

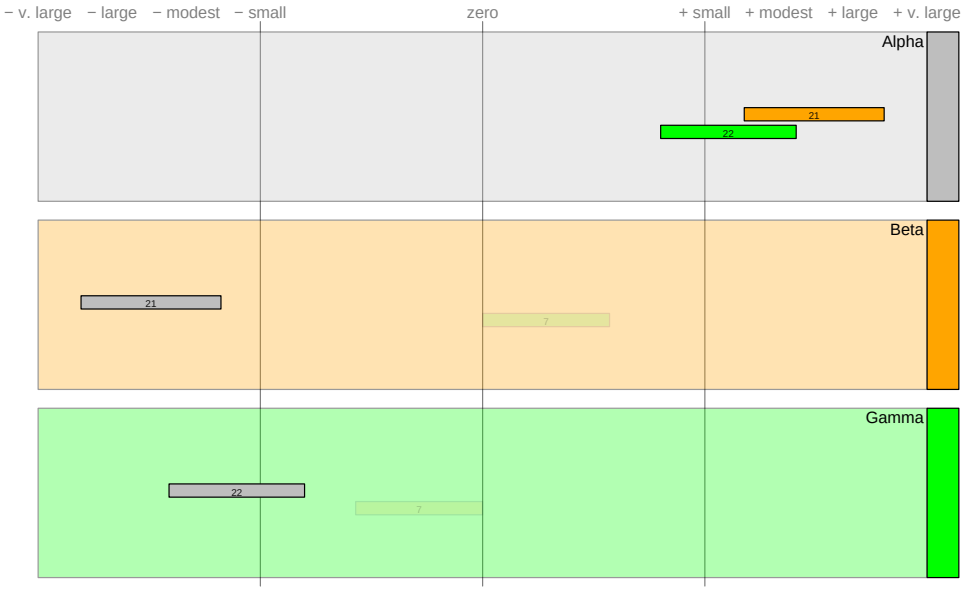

Figure 14: Demonstration (small-heavy weighting)

In the example provided in figure 14, three hypothetical platforms are compared (*Alpha*, *Beta* and *Gamma*). Each small dark rectangle represents the 80% confidence interval for a given comparison. The exact comparison it is referring to is indicated by the respective colours of the rectangle and the larger, fainter rectangle in the background.

There are four dark rectangles and two faint rectangles representing comparisons. The faint rectangles represent the comparison *Beta-Gamma*, which had too few participants to make a useful observation (less than seven). These faint rectangles are provided for completeness only, and are not of central concern.

In the top third, we see *Alpha*'s section, showing the two comparisons made involving the *Alpha* platform. Both of these comparisons are well to the right, indicating that *Alpha* was

rated strongly against both *Beta* and *Gamma*. Likewise, in the middle third (*Beta*'s section), we see the single comparison *Alpha-Beta*, which is positioned well to the left, showing that *Beta* was rated poorly when compared to *Alpha*.

As the darker comparisons are reasonably short, we know that the confidence interval is also small. The faint rectangles exhibit a much longer confidence interval, more or less because they represent so few participants.

5.1.1 How to read the example quickly

For a given platform (let's take *Beta* for example), find its main row, using the label on the right hand side. In our example, this is the middle row of the three main rows. The smaller boxes in this row represent the comparisons against *Alpha* and *Gamma*. As the comparison with *Gamma* did not have enough participants, it is only shown faintly.

The comparison against *Alpha* is shown deep on the left hand side, indicating that our platform *Beta* rated poorly when compared with *Alpha*.

The textual description of the results and the overview are also provided to help scan through the results quickly. The textual description highlights all comparisons that exhibit at least a small difference, and the overview highlights any noteworthy comparisons and unusual clusters from the charts.

5.1.2 Textual description

For increased accessibility, and as a general guide, a textual description of the results is also included. Of course, this is by no means as accurate and comprehensive as the graphical representations. Each line will make a statement about the outcome of a comparison, and provide the average estimated difference (*ave. est. diff.*). Where our various weighting systems have produced different average differences, both are shown. The number of participants (*N*) is also displayed for completeness. An example using the *effort* comparison is included below.

Asked to compare the effort required for each platform, participants indicated that:

Alpha *requires less effort than* Beta, *the ave. est. diff. was:* modest-large	*(N=14)*
Alpha *requires less effort than* Gamma, *the ave. est. diff. was:* large	*(N=58)*

5.1.3 Overview

Noteworthy comparisons and unusual clusters are mentioned in the overview. A noteworthy comparison is one where the arithmetic mean lies above a small difference, with 90% con-

fidence. This must be the case for both weighting systems (*small-heavy* and *large-heavy*) and the comparison must represent the opinions of a sufficient number of participants (more than 8). Helper lines are drawn on the charts to help see which comparisons classify.

5.2 Effort

Observations

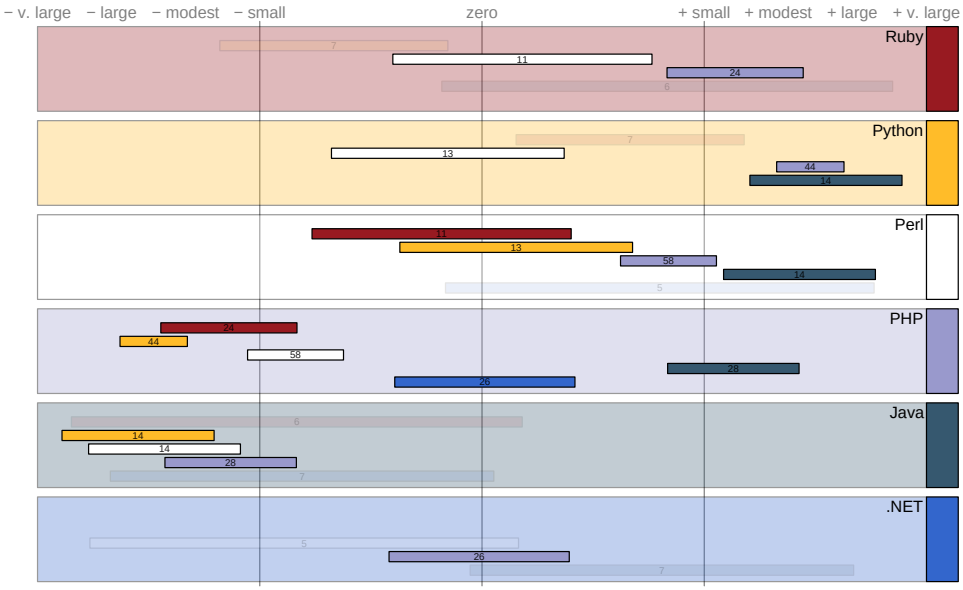

Figure 15: Effort comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 27 on page 63.

Textual description of observations

Asked to compare the effort required for each platform, participants indicated that:

Python *requires less effort than* Java, *the ave. est. diff. was:* large	*(N=14)*
Python *requires less effort than* PHP, *the ave. est. diff. was:* modest-large	*(N=44)*
Perl *requires less effort than* Java, *the ave. est. diff. was:* modest-large	*(N=14)*
Ruby *requires less effort than* PHP, *the ave. est. diff. was:* small-large	*(N=24)*
Perl *requires less effort than* PHP, *the ave. est. diff. was:* small-modest	*(N=58)*
PHP *requires less effort than* Java, *the ave. est. diff. was:* small-modest	*(N=28)*

Overview

There are four points in the charted results worthy of a mention. First are the clear differences indicating that Python requires less effort than PHP and Java, and that Perl requires less effort than Java. Also worth mentioning is the cluster of negative comparisons against Java.

Detailed interpretation

On the one hand this is quite unsurprising, as modern script languages are well-known and accepted for their power of expression and lack of formalities. On the other hand, the large differences in the effort required for Python and Ruby compared to PHP are interesting. This may be due to several factors. Firstly, there are a number of very popular, modern and agile frameworks freely available for Python and Ruby. Less superficial may be that PHP lacks some of the "modern" features of Python and Ruby that such as namespaces and advanced object support which enable the more powerful frameworks to be built. Thirdly, the fact that older versions of PHP lacked object support altogether, may have influenced some of the participants who were drawing on earlier experience.

Although Python was rated higher than Java and PHP more consistently than Ruby was, comparing Python directly against Ruby showed a strong consensus that there is little difference between the two languages, as far as effort was concerned.

Opinions were divided however, when comparing .NET with Java or PHP. Almost all individual responses professed a "modest", "large" or "very large" difference between the platforms in both directions, indicating strongly held opinions. One possible explanation for this is that the experience with Java or PHP can vary considerably depending on the development team and the framework used. Particularly good experiences appear to require less effort and particularly bad experiences, more. It appears that may apply to any or all of Java, .NET or PHP.

5.3 Readability

Observations

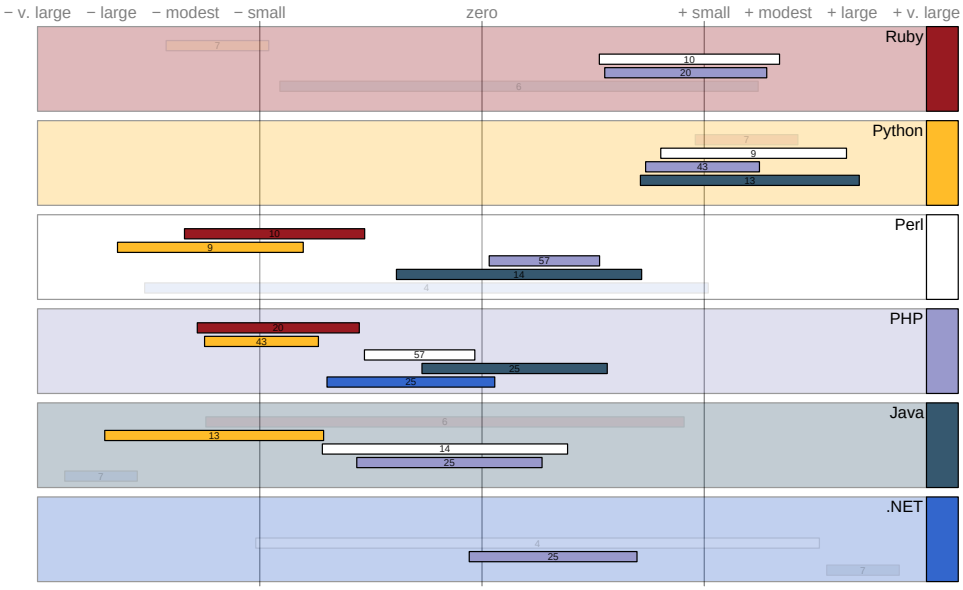

Figure 16: Readability comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 28 on page 64.

Textual description of observations

Asked to compare the readability of applications written in each platform, participants indicated that:

Python *applications are more readable than* Java, *the ave. est. diff. was:* modest-large	*(N=13)*
Python *applications are more readable than* Perl, *the ave. est. diff. was:* modest	*(N=9)*
Python *applications are more readable than* PHP, *the ave. est. diff. was:* small-modest	*(N=43)*
Ruby *applications are more readable than* PHP, *the ave. est. diff. was:* small-modest	*(N=20)*
Ruby *applications are more readable than* Perl, *the ave. est. diff. was:* small	*(N=10)*

Overview

No comparisons here exhibit a particularly clear difference. It is interesting however to note the cluster of positive comparisons for Python.

Interpretation

Readability relies heavily on the programming style adopted by the developers, however it is possible that some languages may be inherently more readable than others.

Python applications exhibit more readability over Perl, Java, Ruby and PHP. This is unsurprising: Python is often advertised as being readable[6] Readability is a central consideration in the design of the language, which uses less punctuation, properties, indexers and enforces consistent indentation[7].

One surprise is the large difference and the global agreement in the .NET and Java comparisons. Keeping in mind that these are the opinions of only seven participants, this is a particularly stark difference for which there are no obvious explanations.

All this is interesting because C#'s syntax resembles Java syntax and the differences in code style generated by the standard IDEs would not explain such a great divide. The best explanation would have to be an accumulation of many minor features (such as verbatim string literals[8], properties[9], indexers and an integrated framework[10]) and the memory of older versions of Java that lacked features such as `foreach`[11]. Once again, it is difficult to explain this difference, because of the vague definition of .NET used in the survey A final influence (which would help exaggerate the relatively minor differences) may be an underlying prejudice, fueled by advertising and rivalry between Java and .NET developers.

The large variation in the Python-Java comparisons is due to two participants, who reasoned that more people know Java and will be able to read and understand the code.

[6] See e.g. `http://www.python.org/about`.

[7] `http://www.python.org/doc/essays/blurb/`. See also decision logs from Python development: `http://www.python.org/dev/peps/pep-0308/` and `http://www.python.org/dev/peps/pep-0318/`.

[8] `http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4472509`

[9] This was mentioned by one participant in their reasoning.

[10] One participant lamented Java's reliance on external libraries.

[11] This was mentioned by one participant in their reasoning.

5.4 Modifiability

Observations

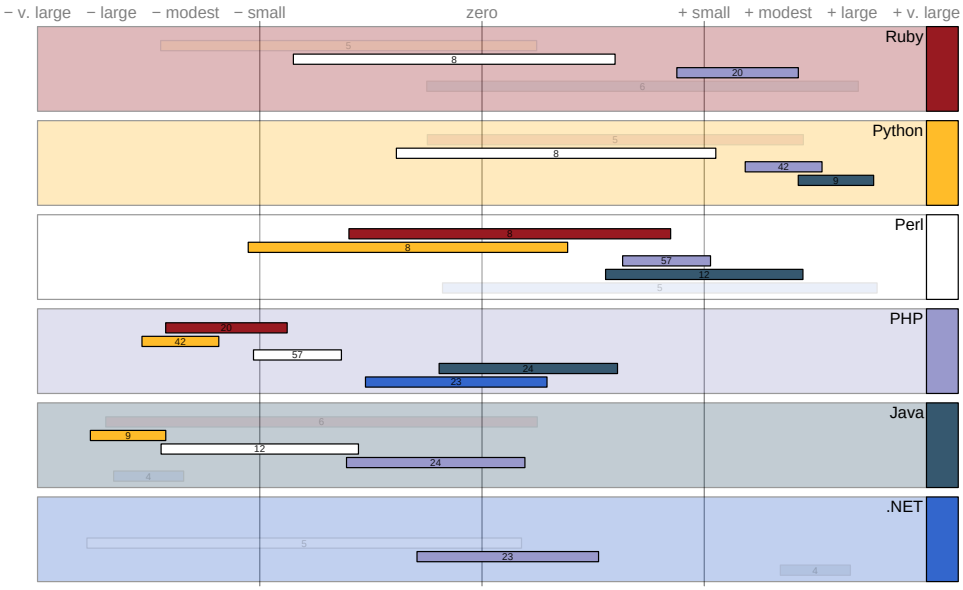

Figure 17: Modifiability comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 29 on page 65.

Textual description of observations

Asked to compare the modifiability of an application in each language, participants indicated that:

Python *applications are more modifiable than* Java, *the ave. est. diff. was:* large		*(N=9)*
Python *applications are more modifiable than* PHP, *the ave. est. diff. was:* modest-large		*(N=42)*
Perl *applications are more modifiable than* PHP, *the ave. est. diff. was:* small-modest		*(N=57)*
Ruby *applications are more modifiable than* PHP, *the ave. est. diff. was:* small-modest		*(N=20)*
Perl *applications are more modifiable than* Java, *the ave. est. diff. was:* small-modest		*(N=12)*

Overview

Two clear results can be seen here: Python applications are more modifiable than both Java and PHP applications.

Interpretation

One particularly interesting observation is the high agreement and clear difference between Python and Java. The reasons given by the participants do not appear to properly explain this difference. It may be that there is no single, identifiable reason, instead, there is a set of more trivial reasons that lead the participants to feel that Python applications are more modifiable. Two likely contributors are "bloat", that is Java typically has more lines of source code, and the need to compile to see the changes in Java.

In contrast, there were many reasons given for the difference between Python and PHP. The recurring reasons submitted were: better modularity[12], unit testing and better frameworks[13].

Also of interest is that the comparisons between .NET/PHP and Java/PHP showed great disagreement, with almost equal advocating on both sides. Reading through the comments gives the sense that both sides have strengths and weaknesses[14] but which ones become relevant seems highly dependent on the type of project.

[12] Including namespaces and object-oriented-programming support

[13] Especially those that practice DRY (Don't Repeat Yourself), which emphasises a single point of control.

[14] PHP: small changes made quickly (no compile necessary), simpler architecture; .NET: Visual Studio features, structure; Java: structure

5.5 Robustness

Observations

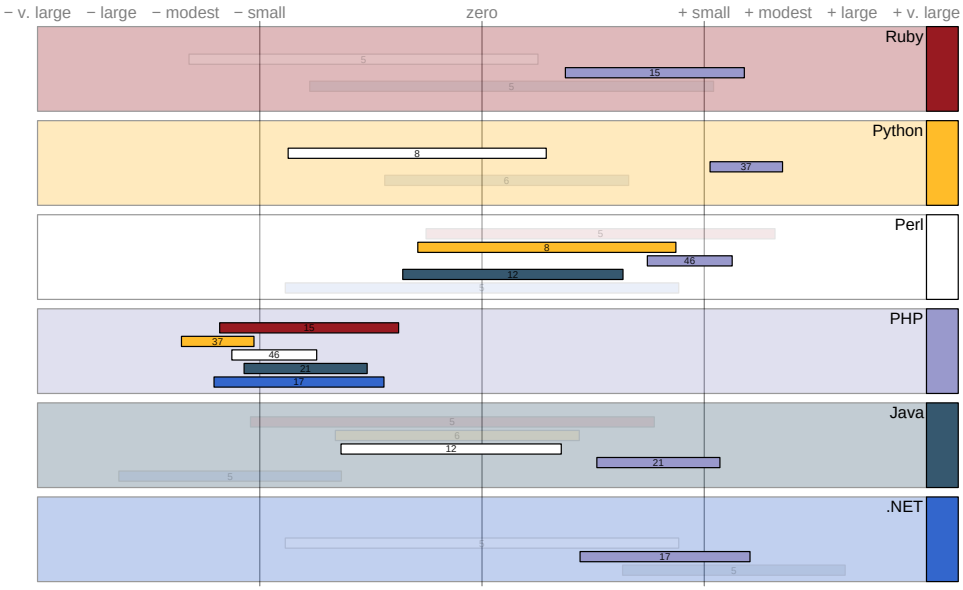

Figure 18: Robustness comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 30 on page 66.

Textual description of observations

Asked to compare the robustness of an application in each language, participants indicated that:

Python *applications are more robust than* PHP, *the ave. est. diff. was:* modest	*(N=37)*
Perl *applications are more robust than* PHP, *the ave. est. diff. was:* small-modest	*(N=46)*
Java *applications are more robust than* PHP, *the ave. est. diff. was:* small-modest	*(N=21)*
.NET *applications are more robust than* PHP, *the ave. est. diff. was:* small-modest	*(N=17)*
Ruby *applications are more robust than* PHP, *the ave. est. diff. was:* small-modest	*(N=15)*

Overview

Only one comparison showed a clear difference: Python being more robust than PHP. Also of note is the cluster of negative comparisons against PHP.

Interpretation

PHP does not draw any favourable comparisons as far as robustness is concerned, but it should be noted that the *average estimated differences* for each of these comparisons are all reasonably small. This is to be expected as far as popular belief is concerned, PHP is often characterised as having potential security problems. But such results contradict the objective results from the Plat_Forms experiment. Common explanations for these subjective results from the participants include:

- better frameworks for Python, Perl, .NET and Java
- poor exception handling, difficult to test
- have experienced bad PHP examples in the past
- `register_globals`
- PHP developers are inexperienced
- Robustness requires more work for PHP

With these comments in mind, it is easier to understand our results. Aside from the first two (perhaps three) valid reasons, the remaining reasons offered represent an underlying prejudice. This contention is supported by the fact that many more participants who listed PHP as their *second* language, rated it much more poorly than those who consider themselves primarily PHP programmers[15]. Combined with the fact that none of the differences were large, it seems that any robustness issues that PHP has can be avoided and do not present a serious

What is interesting is the uniformity displayed between Java and Perl and Perl and Python. Between Java and Perl, it seems both languages offer frameworks with sufficient features to allow an experienced developer to obtain sufficient robustness. While participants were able to point out the features, few responses were able to find flaws in either language. Between Perl and Python, there were also few language flaws discussed, although Perl's validation modules (Cited three times) and Python's exception handling (Cited once.) raised the disagreement level and pushed the mean slightly in Perl's favour.

[15] 9% positive comparison for PHP as first platform versus 47% positive comparison for PHP as second platform.

5.6 Scalability

Observations

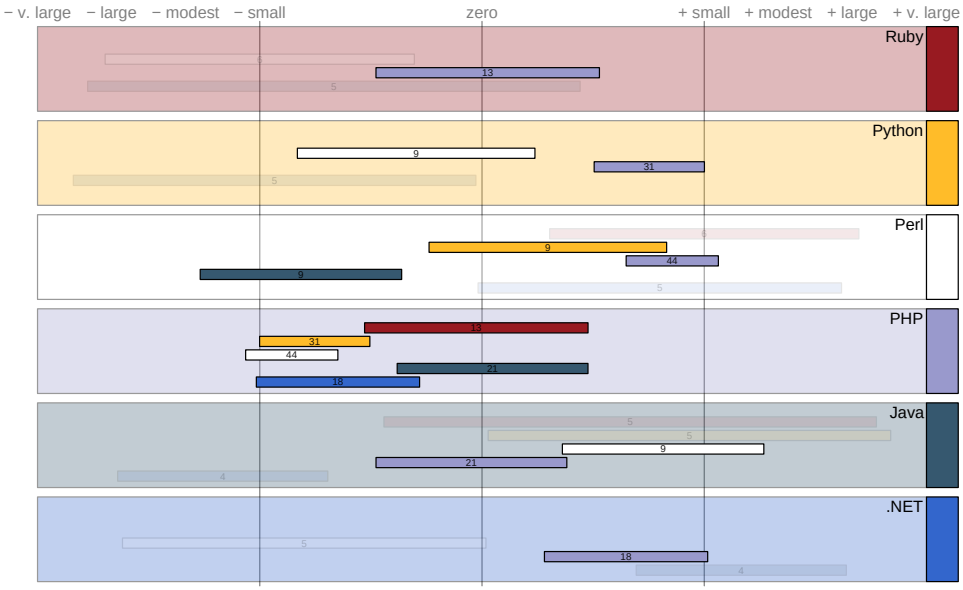

Figure 19: Scalability comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 31 on page 67.

Textual description of observations

Asked to compare the scalability of each platform, participants indicated that:

Perl *applications are more scalable than* PHP, *the ave. est. diff. was:* small-modest	*(N=44)*
Python *applications are more scalable than* PHP, *the ave. est. diff. was:* small-modest	*(N=31)*
Java *applications are more scalable than* Perl, *the ave. est. diff. was:* small-modest	*(N=9)*
.NET *applications are more scalable than* PHP, *the ave. est. diff. was:* small	*(N=18)*

Overview

There are no sufficiently clear differences in the results for scalability.

Interpretation

Scalability is an issue that many developers may not need to address. The lack of any clear differences may suggest a lack of experience in this area, on top of any level of similarity that may have been intended.

Nevertheless, Java was shown to be more scalable than Perl and Perl more than PHP. Interestingly, completing the chain by comparing Java to PHP showed no clear difference, rather, particularly high disagreement between the participants.

Both Java and PHP are often scaled successfully for very high traffic websites.

5.7 Usability

Observations

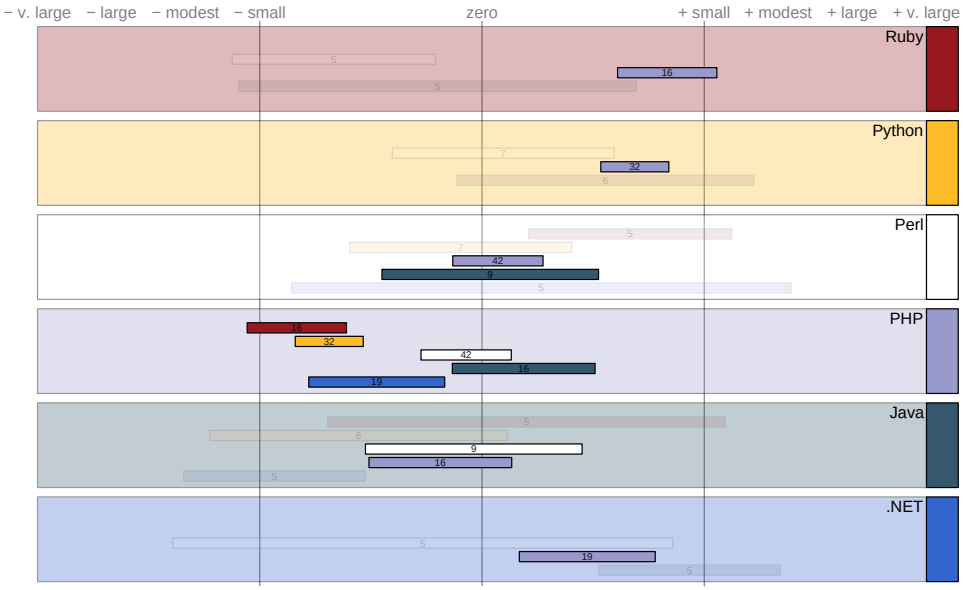

Figure 20: Usability comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 32 on page 68.

Textual description of observations

Asked to compare the usability of applications written in each language, participants indicated that:

Python *applications are more usable than* PHP, *the ave. est. diff. was:* small	*(N=32)*
Ruby *applications are more usable than* PHP, *the ave. est. diff. was:* small	*(N=16)*

Overview

There are no sufficiently clear differences in the results for usability.

Interpretation

Many people did not answer this question, and most of those that did, selected their best-known platform, and indicated that there is little difference involved.

Our general belief is that a large number of participants misunderstood our intentions with this question, The comments echo the notion that platforms have little to do with things such as usability. We did not wish to ask the participants directly, whether or not a platform influences usability. Rather, we asked if one platform tended to have more usable applications than the other. The key difference is that both languages may allow the same level of usability to be produced, however some platforms allow more time to be spent on such things as usability, or perhaps some platforms abstract the interface enough, so that non-technical usability and design experts can work effectively.

As the results are clouded by this common misunderstanding there is little that can be drawn from these results. Perhaps such preconceived notions are best discovered through objective observations, such as those in the Plat__Forms competition.

5.8 Security

Observations

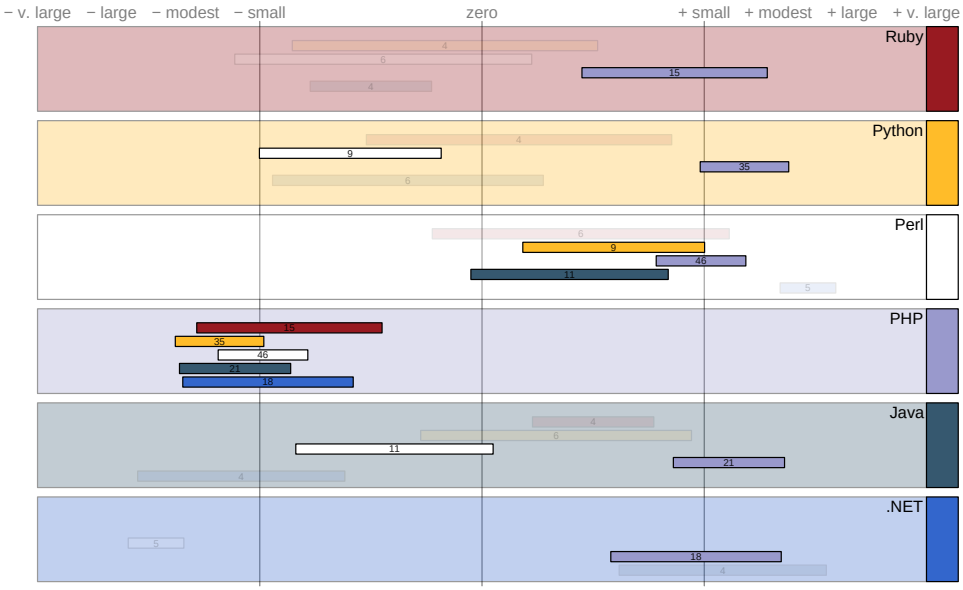

Figure 21: Security comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 33 on page 69.

Textual description of observations

Asked to compare the security of applications written in each language, participants indicated that:

Python	*applications are more secure than*	PHP,	*the ave. est. diff. was:* modest	*(N=35)*
Ruby	*applications are more secure than*	PHP,	*the ave. est. diff. was:* small-modest	*(N=15)*
Java	*applications are more secure than*	PHP,	*the ave. est. diff. was:* small-modest	*(N=21)*
.NET	*applications are more secure than*	PHP,	*the ave. est. diff. was:* small-modest	*(N=18)*
Perl	*applications are more secure than*	PHP,	*the ave. est. diff. was:* small-modest	*(N=46)*
Perl	*applications are more secure than*	Python,	*the ave. est. diff. was:* small	*(N=9)*

Overview

There are no sufficiently clear differences in the results for security, although Python being more secure than PHP came very close. Also of note is the cluster of negative responses with respect to PHP's security.

Interpretation

PHP applications were consistently seen as less secure than all 5 other platforms. PHP developers themselves were seen as the problem, as they are generally viewed to be less experienced. Other reasons were however also often cited, such as unstable libraries and poor design. Not all participants felt PHP was problematic; there is a reasonable amount of disagreement in the Perl, .NET and Ruby comparisons. It seems that PHP can avoid many of its security issues with a good framework.

Not much other information came out in this comparison. Platform similarity was seen in Java/Perl and Perl/Ruby with reasonably high agreement and there was strong disagreement over the respective security of .NET and PHP.

5.9 Speed

Observations

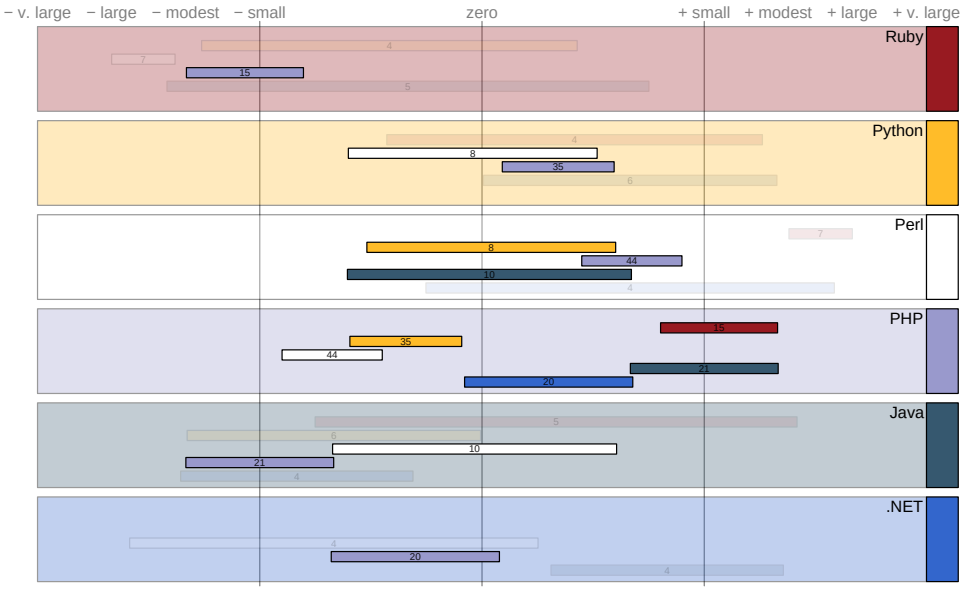

Figure 22: Speed comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 34 on page 70.

Textual description of observations

Asked to compare the speed of applications written in each language, participants indicated that:

PHP *applications are faster than* Java,	*the ave. est. diff. was:*	small-modest	*(N=21)*
PHP *applications are faster than* Ruby,	*the ave. est. diff. was:*	small-modest	*(N=15)*
Perl *applications are faster than* PHP,	*the ave. est. diff. was:*	small	*(N=44)*
Python *applications are faster than* PHP,	*the ave. est. diff. was:*	zero-small	*(N=35)*

Overview

There are no sufficiently clear differences in the results for speed. Perl being faster than Ruby is clear, however this result represents the collective opinion of only 7 participants.

Interpretation

Participants rated Ruby slower than Perl and PHP. Other Ruby comparisons lacked sufficient participants to record a mention.

The participants saw other platforms (.NET, PHP, Java, Python and Perl) as roughly equal, with Perl being rated slightly higher. A variety of reasons are given, some focusing of language speed[16], others look at platform features[17]. As with scalability, this inconclusiveness may be a result of little experience in testing applications for speed. On the other hand, if speed issues are expected for a given language, it may not be seen as a problem because sufficient hardware is organised beforehand.

[16] For example, compiled, semi-compiled and script languages.
[17] For example, Perl's caching features and `mod_perl`.

5.10 Memory

Observations

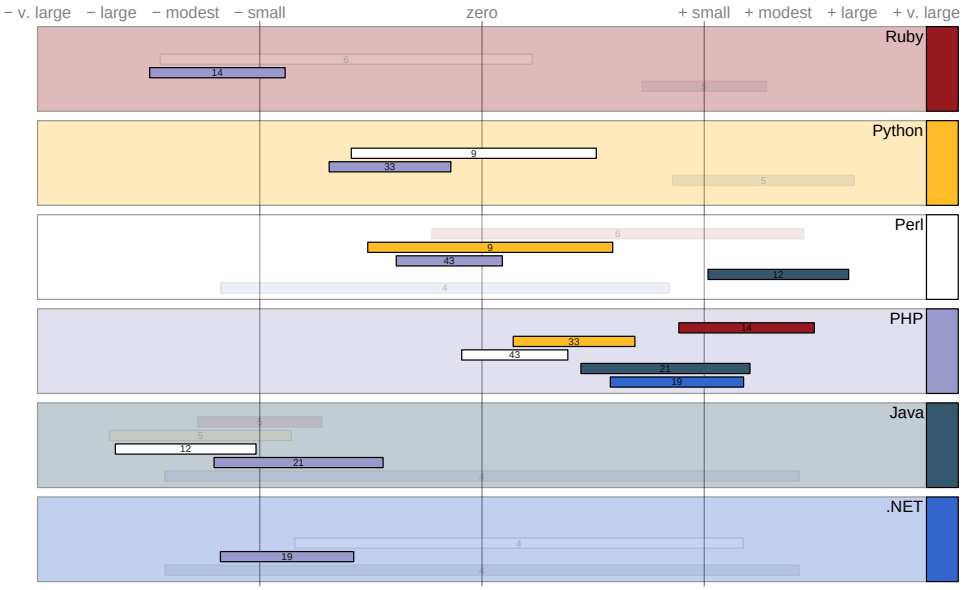

Figure 23: Memory comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 35 on page 71.

Textual description of observations

Asked to compare the memory requirements of applications written in each language, participants indicated that:

Perl *applications require more memory than* Java, *the ave. est. diff. was:* modest-large	*(N=12)*
PHP *applications require more memory than* Ruby, *the ave. est. diff. was:* modest	*(N=14)*
PHP *applications require more memory than* Java, *the ave. est. diff. was:* small-modest	*(N=21)*
PHP *applications require more memory than* .NET, *the ave. est. diff. was:* small	*(N=19)*

Overview

One result is of interest here: Perl applications use less memory than Java applications. Also of note is the possible cluster of positive responses for memory usage in PHP applications, and the possible cluster of negative responses for memory usage in Java applications.

Interpretation

Java was rated consistently poorly, which is no surprise (all but two Java comparisons lacked sufficient participants). Again, there was a series of comparisons showing little difference between the platforms. This may also be on account of a lack of knowledge on the subject from the participants, as memory is rarely an issue for many developers.

5.11 Tools

Observations

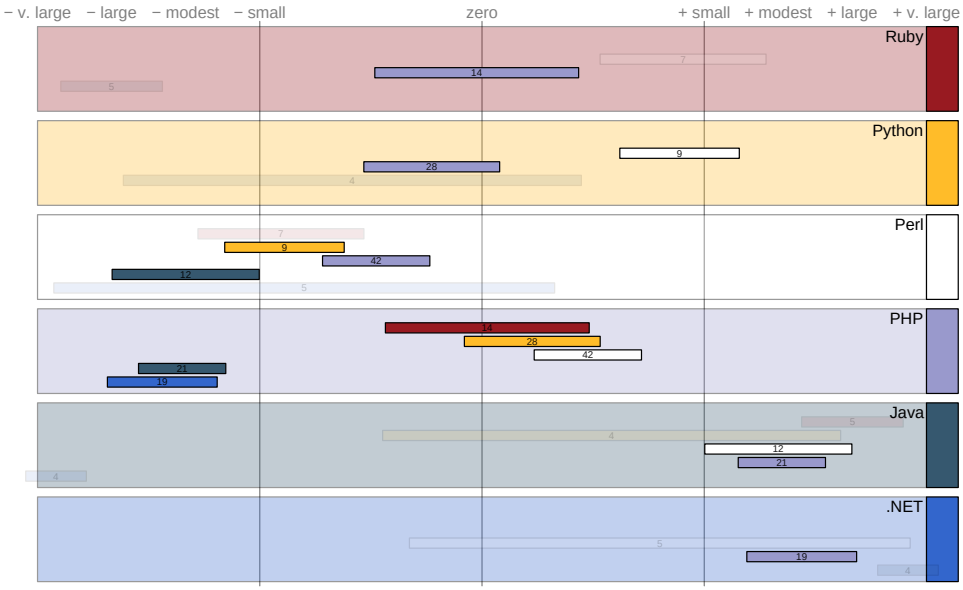

Figure 24: Tools comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 36 on page 72.

Textual description of observations

Asked to compare the tools available for each language, participants indicated that:

Java *has better tool support than* PHP, *the ave. est. diff. was:* modest-large	*(N=21)*
.NET *has better tool support than* PHP, *the ave. est. diff. was:* modest-large	*(N=19)*
Java *has better tool support than* Perl, *the ave. est. diff. was:* modest-large	*(N=12)*
Python *has better tool support than* Perl, *the ave. est. diff. was:* small	*(N=9)*

Overview

Three comparisons are worthy of a mention here: Java has better tool support than Perl and PHP, and .NET has better tool support than PHP. Also of note is the cluster of negative responses for Perl's tool support.

Interpretation

.NET was strongly favoured and the reason consistently given was, of course, Visual Studio. The strong support for Java in comparison with PHP and Perl may point to the difference between a heavily commercially supported languages (Java) and the open source script languages.

The comments lament Perl's poor IDE support, which comes through in the *small-heavy* weighting.

5.12 Tool dependence

Observations

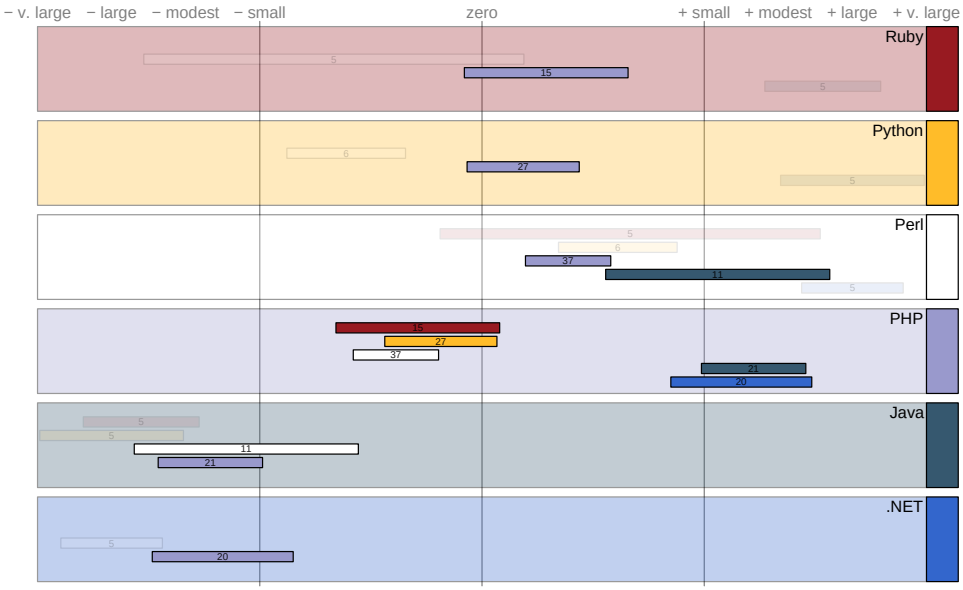

Figure 25: Tool dependence comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 37 on page 73.

Textual description of observations

Asked to compare the dependency on tools of each language, participants indicated that:

PHP *is less dependent on tools than* .NET, *the ave. est. diff. was:* modest-large	*(N=20)*
PHP *is less dependent on tools than* Java, *the ave. est. diff. was:* modest	*(N=21)*
Perl *is less dependent on tools than* Java, *the ave. est. diff. was:* small-modest	*(N=11)*
Perl *is less dependent on tools than* PHP, *the ave. est. diff. was:* zero-small	*(N=37)*

Overview

There are no sufficiently clear differences between the individual languages, although Java's dependency on tools in comparison with PHP almost qualified.

Interpretation

The only clear distinction that can be made here is between Java and .NET and the script languages. Both Java and .NET are complicated to write "by hand", that is, using a text editor. On the other hand, PHP and Perl are quite often written using a text based editor, such as `vim` or `emacs`. This is evidenced in the comments, almost all commentary involving Java or .NET could not imagine life without a comprehensive IDE.

Again, the purpose of this question was not made entirely clear. By 'dependent', we did not try to compare life with the current set of IDEs to life without. Rather, it was hypothesised that the success of a project for some languages, depends more on the strength of the tools (e.g. IDEs) than other languages. A future survey would do well to make this point clearer.

5.13 Framework dependence

Observations

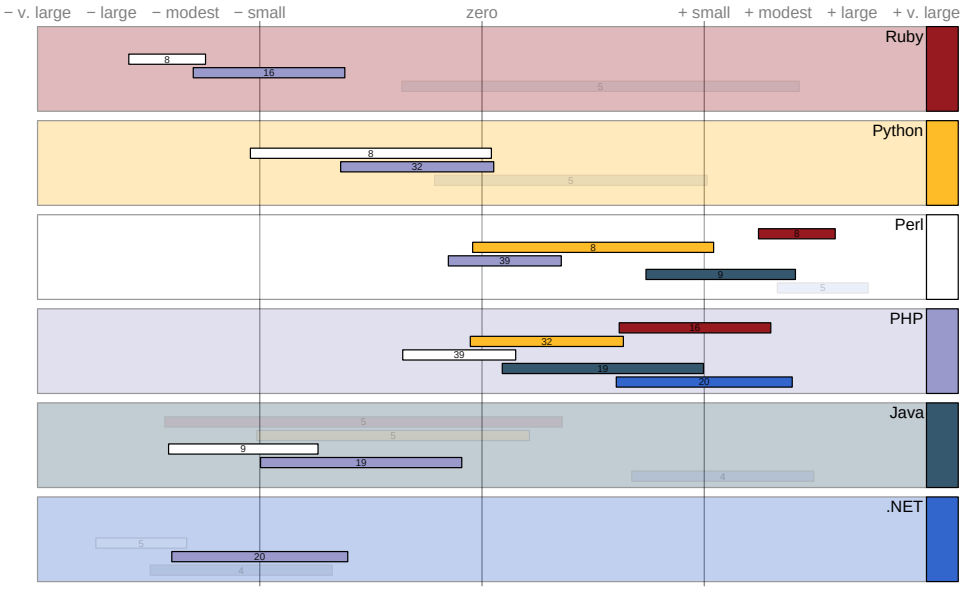

Figure 26: Framework dependence comparisons (small-heavy weighting)

The large-heavy weighting chart can be found in figure 38 on page 74.

Textual description of observations

Asked to compare the dependency on frameworks for each language, participants indicated that:

Perl *is less dependent on frameworks than* Ruby, *the ave. est. diff. was:* modest	*(N=8)*
PHP *is less dependent on frameworks than* .NET, *the ave. est. diff. was:* small-modest	*(N=20)*
PHP *is less dependent on frameworks than* Ruby, *the ave. est. diff. was:* small-modest	*(N=16)*
Perl *is less dependent on frameworks than* Java, *the ave. est. diff. was:* small-modest	*(N=9)*
PHP *is less dependent on frameworks than* Java, *the ave. est. diff. was:* small	*(N=19)*

Overview

One comparison is noteworthy here, that being Ruby's dependency on frameworks as opposed to Perl. Also of note may be the clusters of positive comparisons for Perl and PHP.

Interpretation

Again, it seems the participants did not understand the question as intended. A useful result would be the comparison of how the development experience depends on the quality of the framework. Although this question is useful for languages such as Java, PHP and Python, it is troublesome .NET and Ruby, where there exists only one dominant framework.

Ignoring .NET and Ruby, there is the finding that the Java experience is more dependent on the quality of the framework than Perl, which is in line with the observations from the Plat_Forms project. What was not evident in the Plat_Forms contest, was the high disagreement between Java and PHP over framework dependence. PHP proved itself to be a particularly consistent platform during the competition, but this is not clearly evident in the survey results.

6 Methodological lessons learnt

6.1 Major problems encountered

Our biggest issue was ensuring that the participants properly understood exactly what we wanted from them. Ideally, we wanted to collect comparisons on isolated topics, from participants who draw on their professional experience.

Quite often, comparisons were submitted along with comments along the lines of

> "The developers influence *Attribute X* much more than the language used"

This represents a major misunderstanding of the task. Given the frequency with with this sentiment was expressed, it is clear that we had not properly communicated our goals to our participants.

> "But when you choose a language, you're also choosing a community."
> — Paul Graham

To illustrate the misunderstanding, we can take a look at *usability* of a web application, which does not seem to have much to do with the language. This does not mean, however, that certain languages wont lead to more usable applications than others. A language is much more than its syntax and libraries; certain languages may have better frameworks, better abstraction, better developers, more design-oriented developers, or even simply allow more time and effort to be spent on usability engineering.

The individual participants need not make this aggregation themselves, we would have been happy if they simply thought about the applications they have created, and which ones were (in their opinion) more usable.

6.2 Minor problems encountered

To facilitate any future attempts at a similar survey, here is a list of minor problems we encountered.

- Almost no referrer information was received when using mailing lists, we could not be certain which channels were working (too) well.
- "Classic ASP" was inappropriately mixed with .NET.
- The questions were not completely suitable for .NET participants.
- The .NET and Java people are difficult to reach, there is no centralised community.

- There was no "I don't know" option on the comparisons, which would allow participants to 'opt-out' of questions where they feel they have little expertise.
- We underestimated the time needed to come up with a good analysis approach and to read and classify the large amount of comments.
- Limited budget, we did not really have enough resources to do a sophisticated analysis.

6.3 Improvements in future questionnaires

Here are some technical improvements that may be made for future questionnaires.

- Include a "Quit" button, leading to "why are you leaving?" page, with feedback opportunity. Although users can simply leave the page, by closing the window, many will click on the button simply because it is there. It is foreseeable that people might accidentally click on it: For this case, "undo" absolutely must be possible. Only about 4–5 answers are required to identify problems. Thus, this may be only be visible in the test phase.
- Put "effort" comparison question last (specialised questions should be before general questions)
- "I don't know" escape option to prevent or identify guesses in the comparisons.
- Get bloggers involved, to reach a more heterogeneous audience.
- Put the university logo on the front page, to appeal to authority.
- Implement a separate, write-protected data-browsing site to monitor/improve publicity efforts.
- Implement a separate site, with up-to-date statistics, that are relevant for future invitations. Number of visitors and comparisons made, referrer information and anything that is mentioned in the participants section of this report (section 4 on page 16). It should also be possible to update this page, without disturbing the production server.
- Use a test survey, aimed at identifying why and where people give up answering.
- Randomise the order of the comparison questions, so that all comparisons have an equal number of participants.

7 Conclusion

7.1 Summary of findings

Aside from the findings for each individual comparison between languages, it is interesting to look at some groups of languages, for example strongly typed or compiled languages compared with script languages or comparing modern platforms to traditional.

7.2 Comparing compiled languages with script languages

Given the under-representation of .NET, it is difficult to separate compiled-language traits from Java language traits when Java is the sole representative of compiled languages. Using a combination of common sense and qualitative feedback through the comments fields of the survey, it is possible to draw some careful conclusions.

In the "effort" comparison, Java was consistently seen as requiring more effort, although there are many frameworks and tools available to make development easier. The comments make it clear that the compilation step, and the required structure explain this difference. Script languages all facilitate small, iterative changes to be made, which can often mean less effort. It is important to note that some development approaches do not require small iterative changes (cleanroom for example), and that scripted languages may not significantly reduce the effort required in these cases.

Similar groupings were seen in the "memory" and "tool dependence" comparisons: The script languages require less memory (except Ruby) and are less tool dependent. This agrees with our expectations: compiled languages are targeted at larger applications, and so speed rather than memory is important. Tool independence achieves flexibility for the (open source) script languages. Scripting frameworks also lack "boilerplate code" of the more formal languages, which increases reliance on development tools, and the added power of expression reduces the benefit of code generation.

7.3 Comparing modern languages with traditional languages

Of course, many traditional web development languages were not included in this survey, but PHP and Perl, and classic .NET are clear members of this category. Languages such as Java, Ruby and Python are relatively new to the web development scene[18] and offer linguistic features and platforms that were never in the traditional languages' designs.

To this end, we notice some differences in the results. Quite often, either Python or Ruby or both managed to differentiate themselves from PHP. The both require less effort and are more modifiable and readable than PHP. Neither exhibited some of the security problems and robustness problems that participants identified in PHP.

[18] Python and Java have been in web deployments for some time, but have evolved rapidly in the last 5 years.

Interestingly, the modern languages were not able to consistently set themselves apart from the very traditional Perl. Readability and tool support were the only noted Perl shortcomings in comparison to the modern languages.

7.4 Secure languages

Participants made it particularly clear that they felt PHP applications were less secure and less robust. This is a direct contradiction of findings from the Plat_Forms experiment, which may suggest that this observation is a result of an underlying prejudice.

The explanations offered by the participants pointed to the large number of inexperienced PHP developers, and the trouble they can make with PHP. This might mean that PHP requires a certain level of expertise to avoid security issues.

Worthy of a mention in the security comparisons is Perl, which was consistently rated as being secure.

7.5 Efficient languages

Looking at speed and scalability, Perl and PHP were rated well for speed, Java and Perl were rated well for scalability. This confirms our expectations, Java is known to have an overhead, but also to scale particularly well. What is mildly surprising is that PHP did not rate particularly well in comparison with the other platforms, yet runs a number of extremely high traffic web sites.

7.6 Memory intensive languages

One clear result from the memory comparisons was PHP, whose applications use less memory than all other platforms. There was an impressive difference between PHP and Ruby, likewise Perl also clearly out-performs Java.

7.7 Further work

7.7.1 Further work for our data

We did not have the resources to categorise the comments from the participants, which would provide a useful overview of the opinions and results. Further work might include cross-comparisons, trying to identify patterns of responses from the participants.

7.7.2 Possible follow-up work

A repeat of this survey would be particularly valuable if it could address some of the issues mentioned earlier and reach a larger number of participants with a professional background. Also interesting would be results with a higher number of .NET and Java comparisons.

7.8 Acknowledgements

The preparation of this survey and document is the work of more than one person. In particular, the management and formulation of the survey was done by Lutz Prechelt, Florian Thiel, Ulrich Staerk and myself, with feedback from Stephan Salinger. Facilities for the running of the survey were provided by the Free University of Berlin. This document was written with great help from Lutz Prechelt, Florian Thiel and Ulrich Staerk.

Appendix A Validation data

Table 2: Duplicate participating IP addresses. Actual IP addresses are replaced by a pseudo-name, in line with the privacy policy in Appendix D on page 84. One participant was consequently removed, this is also indicated below.

ID	IP address	start hour	comparison	no. comparisons	completed
93	A	25	Perl-PHP	0	NO
385	A	48	Perl-PHP	12	YES
110	B	33	Perl-Java EE	12	YES
452	B	59	Perl-PHP	0	NO
~~471~~	~~B~~	~~61~~	~~Perl-Java EE~~	~~6~~	~~YES~~
475	B	61	Perl-Ruby	0	NO
496	B	64	Perl-PHP	0	NO
128	C	35	Python-Java EE	10	YES
208	C	38	PHP-Java EE	0	NO
249	D	40	Java EE-.NET	2	NO
677	D	166	Perl-PHP	0	NO
743	E	178	PHP-Ruby	12	YES
745	E	178	PHP-Ruby	0	NO
748	E	178	PHP-Java EE	0	NO
751	E	178	.NET-Java EE	0	NO
797	F	203	.NET-Python	0	NO
798	F	203	.NET-Java EE	0	NO

Table 3: Number of participants per country. This also includes visitors who did not submit a comparison.

Country	Participants
Germany	260
United States	128
United Kingdom	43
Canada	25
Austria	19
Australia	17
Switzerland	15
India	10
Italy	9
Romania	9
France	7
Netherlands	7
Russian Federation	7
Brazil	6
Israel	6
Argentina	5
Spain	5
Poland	4
Portugal	4
Belgium	3
Japan	3
Lithuania	3
Denmark	2
Finland	2
Greece	2
New Zealand	2
Norway	2
Slovenia	2
Sweden	2
22 other countries	1

Appendix B Additional results charts

The following charts are the same charts from the results section (section 5 on page 28), with large-heavy weighting, as opposed to small-heavy.

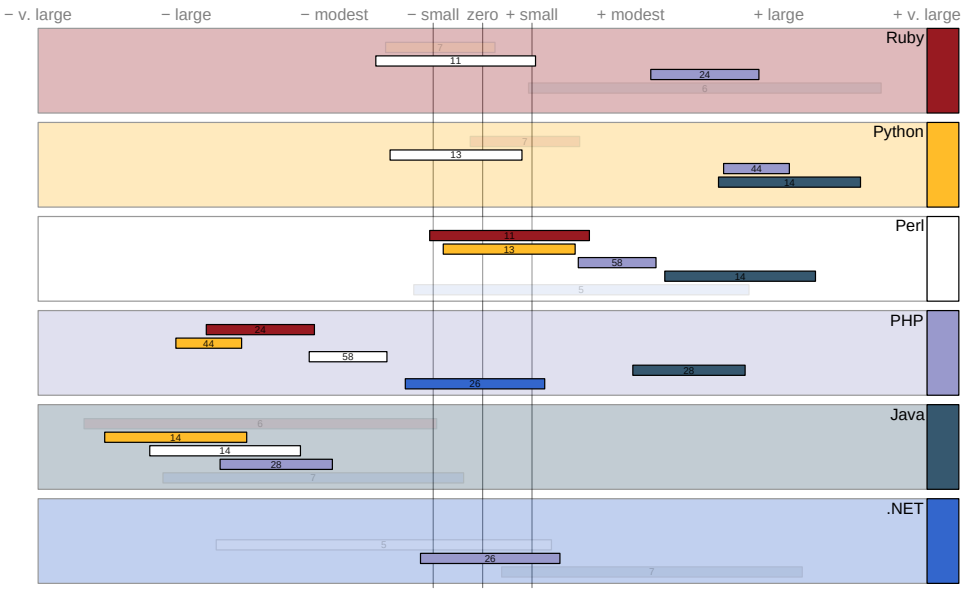

Figure 27: Effort comparisons (large-heavy weighting)

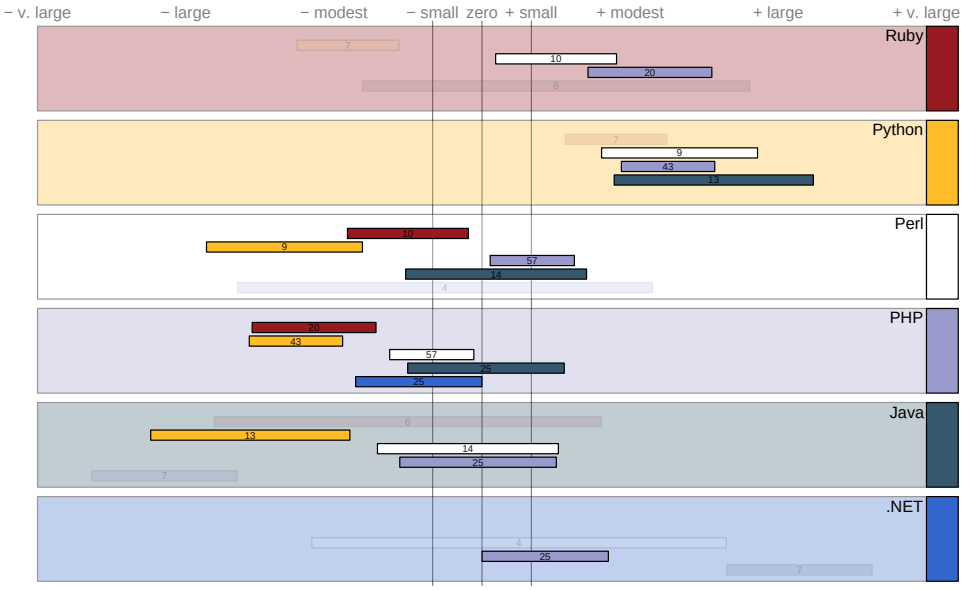

Figure 28: Readability comparisons (large-heavy weighting)

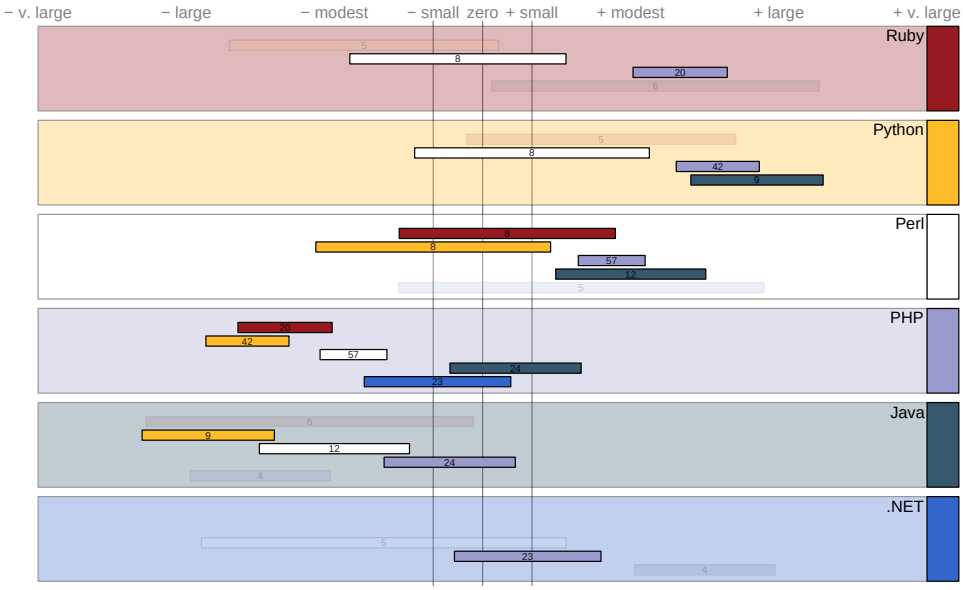

Figure 29: Modifiability comparisons (large-heavy weighting)

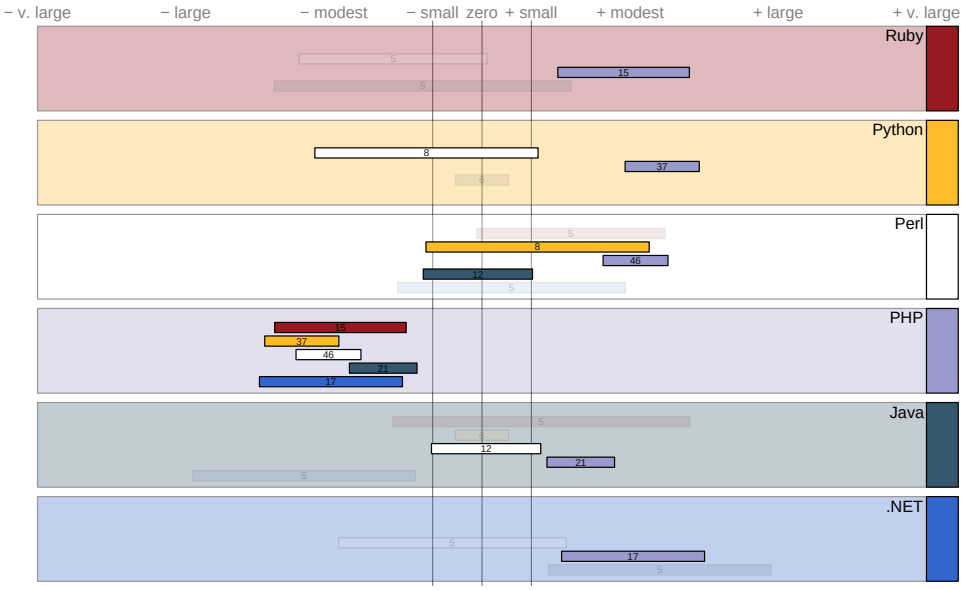

Figure 30: Robustness comparisons (large-heavy weighting)

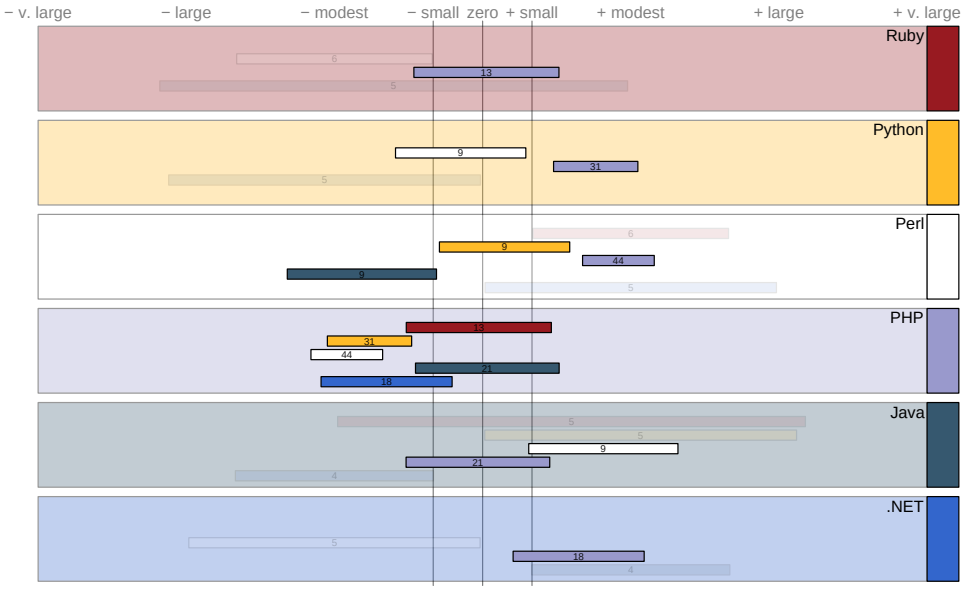

Figure 31: Scalability comparisons (large-heavy weighting)

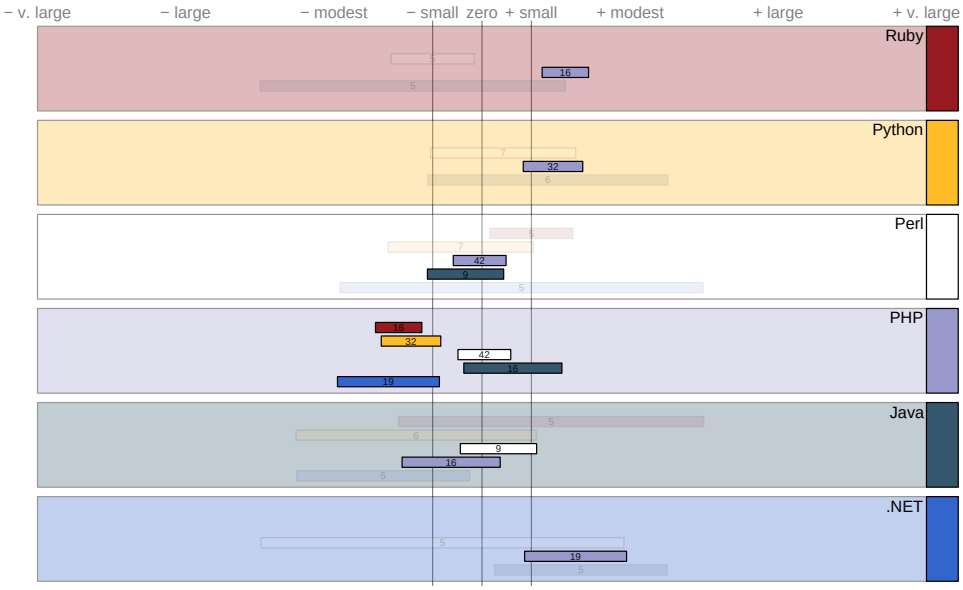

Figure 32: Usability comparisons (large-heavy weighting)

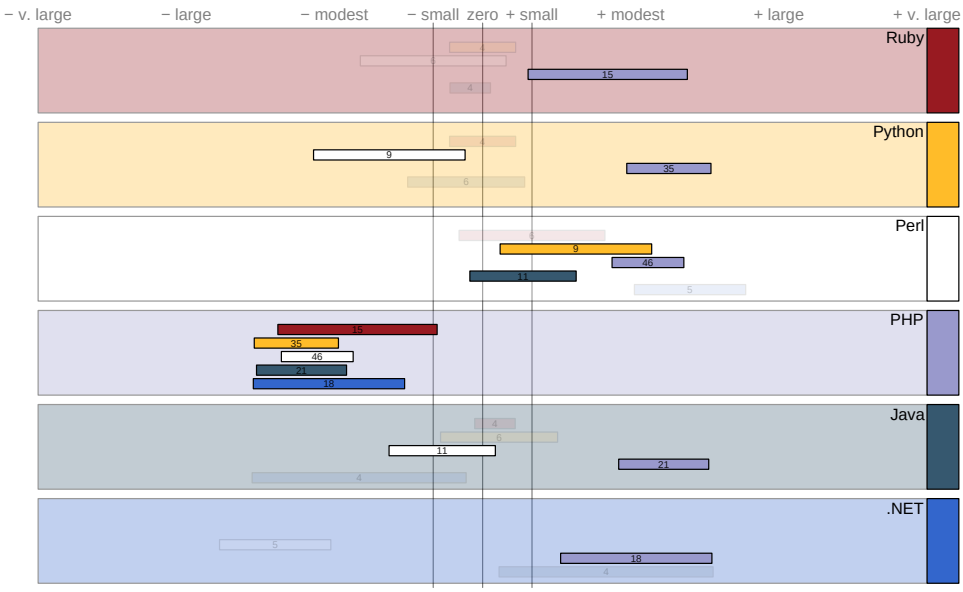

Figure 33: Security comparisons (large-heavy weighting)

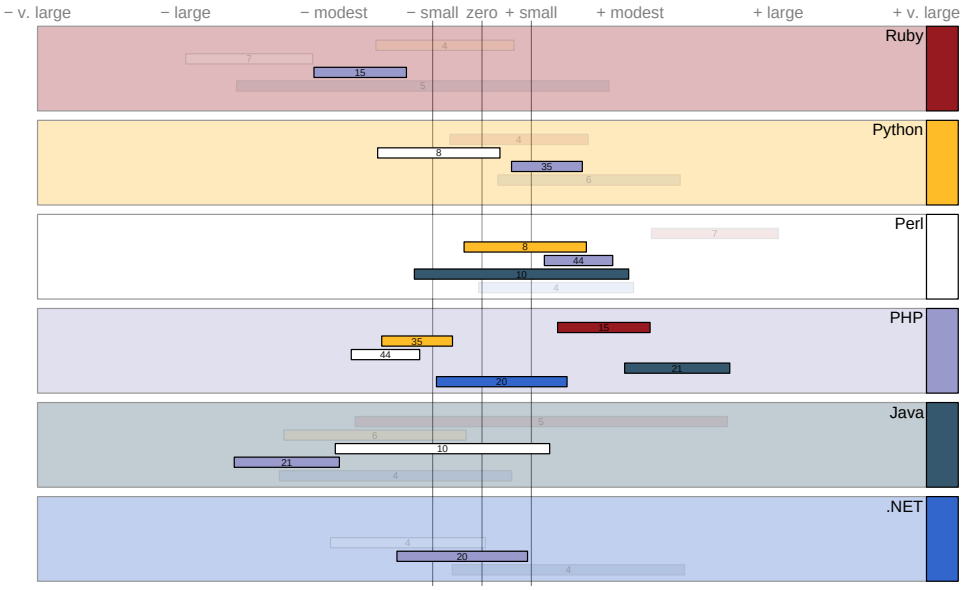

Figure 34: Speed comparisons (large-heavy weighting)

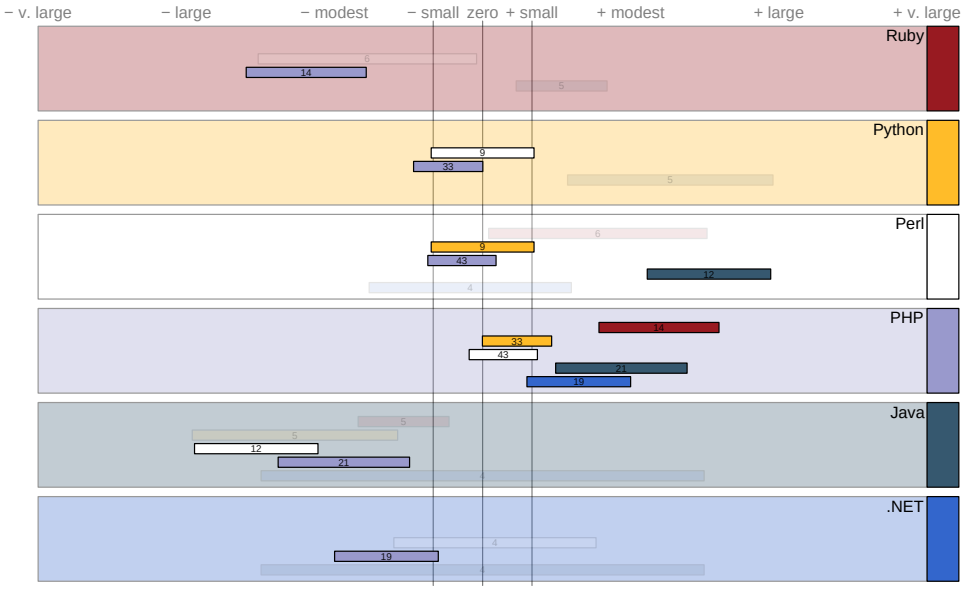

Figure 35: Memory comparisons (large-heavy weighting)

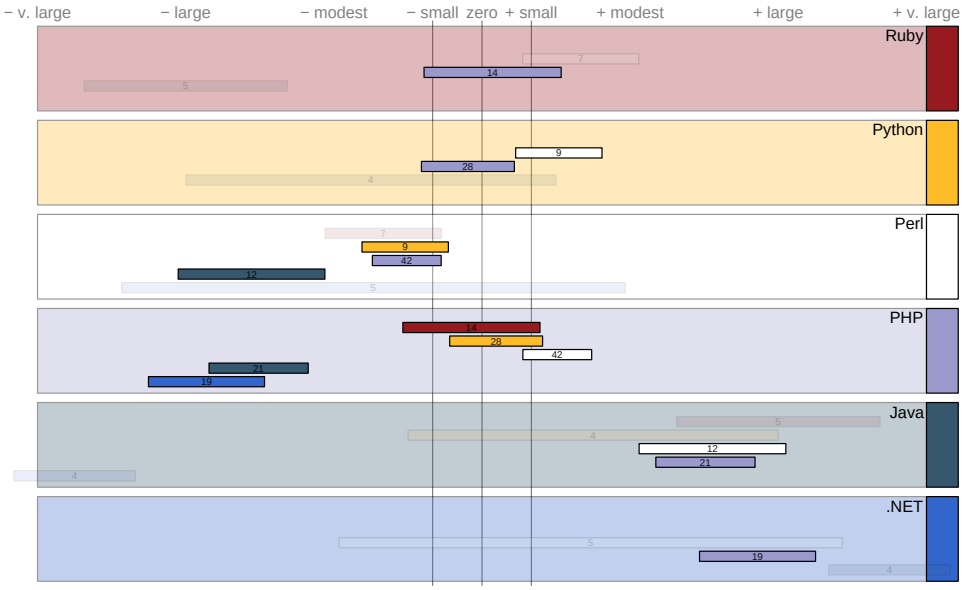

Figure 36: Tools comparisons (large-heavy weighting)

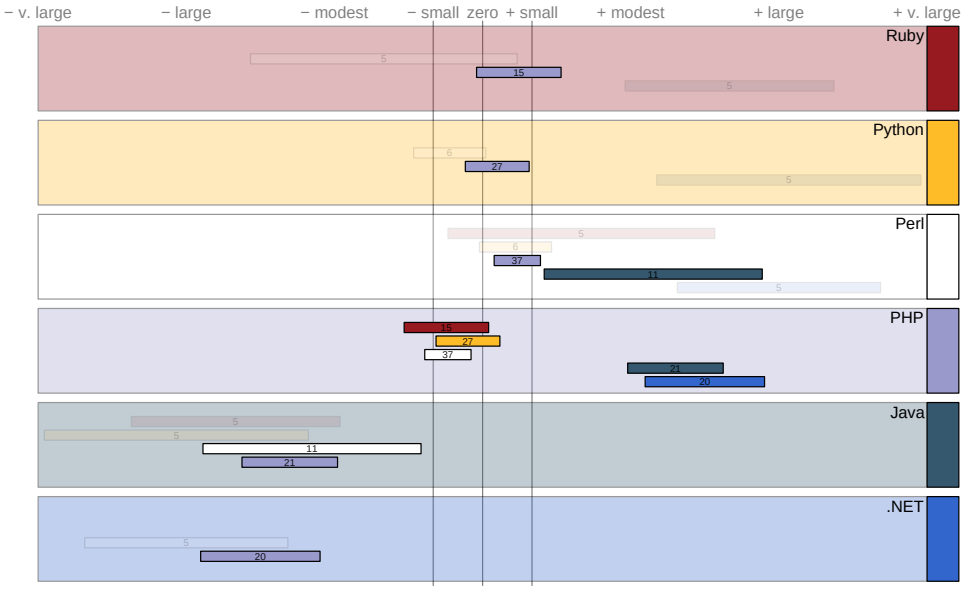

Figure 37: Tool dependence comparisons (large-heavy weighting)

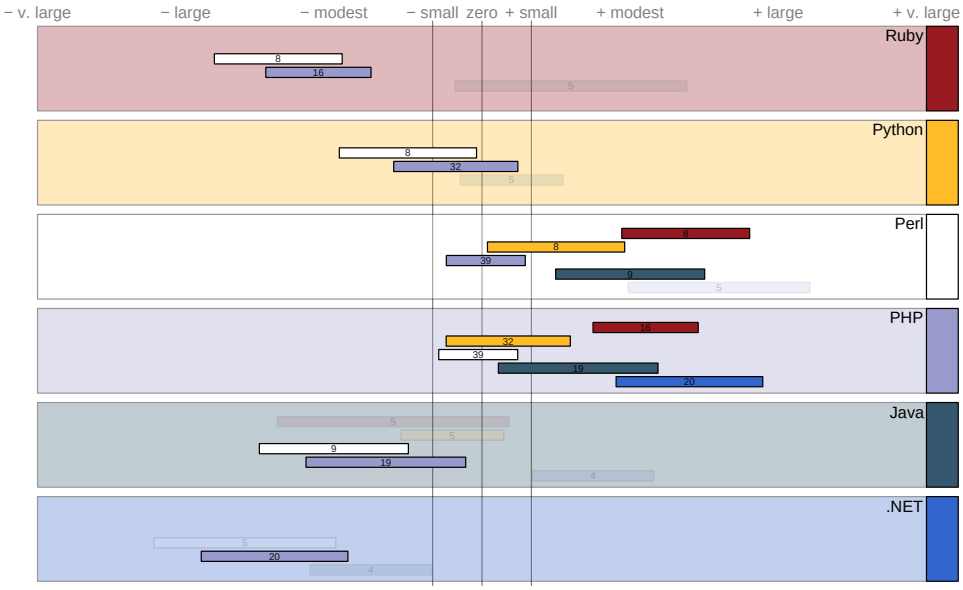

Figure 38: Framework dependence comparisons (large-heavy weighting)

Appendix C Questionnaire

Page 1: Platforms

My most extensively used platform is:

- .NET (ASP etc.)
- Java EE
- Perl
- PHP
- Ruby
- Python

My second most extensively used platform is:

- .NET (ASP etc.)
- Java EE
- Perl
- PHP
- Ruby
- Python

Page 2: Learning

Which platform did you learn and develop with first?

- *Platform A*
- *Platform B*

Page 3: Experience

Compared to the extent of my experience with *Platform A*, I would consider my experience with *Platform B* to be:

- about as thorough
- somewhat less thorough
- about half as thorough
- much less thorough

Page 4: Applications

With *Platform A*:

- I have implemented web applications alone with a total effort of about *XX* person weeks.
- I have implemented web applications with other people, with a total effort of about *XX* person weeks on my part.

With *Platform B*:

- I have implemented web applications alone with a total effort of about *XX* person weeks.
- I have implemented web applications with other people, with a total effort of about *XX* person weeks on my part.

Page 5: Frameworks

On *Platform A*, I am regularly using or have used during some time in the past the following frameworks for writing web applications:

On *Platform B*, I am regularly using or have used during some time in the past the following frameworks for writing web applications:

Page 6: Effort

I believe that a web application built with *Platform A* tends to

- require *more/less* effort

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 7: Readability

I believe that a web application built with *Platform A* tends to

- be *easier/more difficult* to understand for an external programmer

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 8: Modifiability

I believe that a web application built with *Platform A* tends to

- require *more/less* effort to design, perform, and test an unexpected change.

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 9: Usability

I believe that a web application built with *Platform A* tends to

- be *more/less* usable for the end user

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 10: Robustness

I believe that a web application built with *Platform A* tends to

- be *more/less* robust against nonsensical end-user inputs

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 11: Security

I believe that a web application built with *Platform A* tends to

- be *more/less* secure against malicious attacks

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 12: Speed

I believe that a web application built with *Platform A* tends to

- be *better/worse* in terms of response time

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 13: Memory

I believe that a web application built with *Platform A* tends to

- consume *more/less* memory on the server side.

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 14: Scalability

I believe that a web application built with *Platform A* tends to

- be *more/less* scalable to huge numbers of users

. . . than an equivalent one built with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 15: Tools

I believe that web application development with *Platform A* tends to

- have *better/worse* tool (IDE) support

. . . than with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 16: Tool Dependence

I believe that web application development with *Platform A* tends to

- be *more/less* dependent on the use of sophisticated tools

. . . than with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 17: Framework Dependence

I believe that web application development with *Platform A* tends to

- be *more/less* framework dependent — that is, the choice of framework has a great impact on the product quality

. . . than with *Platform B* and I believe the typical size of the difference is:

- very large
- large
- modest
- small
- about zero

I believe the main reasons for the difference are: *reasons*

Page 18: My Background

I am *XX* years old and have years of experience with software development.

Check all that apply:

- I develop software professionally
- I develop software as a student
- I develop software as a hobby
- I contribute to open source software
- I develop software occasionally

My formal education is:

In the past 12 months, I have spent approximately *XX* percent of my work time with technical software development activities (as opposed to project management, non-software-related activities, etc.)

Page 19: My Capabilities

Among all programmers that are creating web applications, I consider myself:

- among the most capable 5%
- among the most capable 10%
- among the most capable 20%
- among the most capable 40%
- about average
- among the least capable 40%
- among the least capable 20%
- among the least capable 10%
- among the least capable 5%

I am regularly using or have used during some time in the past the following programming languages:

I have also at some point tried out (i.e. have written at least one program of at least 50 lines) the following programming languages:

Appendix D Privacy policy

The following privacy statement is the same statement that we displayed on the website and has not changed since it was first published.

The information we collect is to be used solely as part of the Plat_Forms research. No information will be kept that will allow individual responses to be identified. In addition to the responses, your IP address, timestamps and session information will be recorded and will be discarded once the data has been validated.

If you are interested in receiving the results of the research by email, your email address will be stored separately to the given information.

If you have any questions about this research, or would like to offer some feedback, please feel free to send us an email at `hardy@inf.fu-berlin.de`

Appendix E Announcement channels

```
26 June 10:51 rails mailing list
26 June 10:40 django-users mailing list
26 June 17:39 web.py mailing list
27 June 17:42 digg
26 June 18:08 pylons mailing list
26 June 17:59 slashdot
26 June 19:25 comp.lang.perl.misc
26 June 19:40 comp.lang.php mailing list
27 June 10:17 forum.java.sun.com
26 June 18:12 turbogears mailing list
27 June 11:11 all@inf mailing list
27 June 11:54 server side
27 June 12:22 joel on software
27 June 12:46 comp.lang.java.programmer mailing list
27 June 12:54 python forum
28 June 21:35 microsoft.public.dotnet.framework.aspnet
28 June 22:34 msdn channel 9
28 June 23:41 yahoo asp.net2 list
28 June 23:41 forum asp.net free-for-all
28 June 23:49 aspmessageboard.com
29 June 00:20 particletree blog
30 June 09:01 microsoft.public.de.german.entwickler.dotnet.asp
02 July 19:19 Heise forum
02 July 22:00 creativeui.com
05 July 11:51 perl-community.de
```

Appendix F Invitation text

We used one generic template and added a little address to the respective community to get more responses. The text shown below was used for the Java Enterprise Edition developers. All the other texts are similar.

```
Dear JEE developers,

I'm currently part of a research team at the Free University of
Berlin, looking into the ways in which the major web development
platforms differ. In addition to our work with the Plat_Forms contest,
we're now looking for actual professional opinions.

If you have practical experience in the development of non-trivial web
applications with two or more web development languages then we would
like you to participate in a short survey. Three Java teams
participated in this year's Plat_Forms contest but no one used a pure
JEE approach. It would be good to have your opinions and experiences
in the survey.

The questionnaire asks you about your experiences and opinions on two
of the web development languages you have used, and should take about
10-15 minutes to complete. To thank you for your participation, you
will be able to provide an email address and we will send you our
final report when we are finished compiling it.

    http://www.plat-forms.org/survey/

We look forward to your answers and experiences, let me know via email
if you have any questions.

Florian Thiel
Plat_Forms survey team
```

References

[1] Lutz Prechelt. Plat_Forms 2007: The Web Development Comparison — Evaluation and Results, Technical Report TR-B-07-10, Freie Universitaät Berlin, Institut für Informatik, Germany, June 2007.