CHAPTER 9

SUCCESSFUL AND ABANDONED SOURCEFORGE.NET PROJECTS IN THE *GROWTH STAGE*

This chapter undertakes a Classification Tree analysis similar to the previous chapter, but for successful and abandoned projects in the *Growth Stage*. Readers may recall from Chapter 3 (Figure 3.2) that the Growth Stage is the period of time after the project has produced a first public release of its software. Readers are also reminded that the definitions and operationalization of the dependent variable, success and abandonment in the Growth Stage, are discussed in Chapter 6. For example, a project is classified as a Success in the Growth Stage (SG) when it has "produced three releases of a software product that performs a useful computing task for at least a few users." Readers are encouraged to review Chapter 6 (especially Table 6.1) for specifics on how we operationalized this definition as well as the other Growth Stage dependent variable categories (e.g., Abandoned in Growth, Indeterminate in Growth).

Results

The analysis for the Growth Stage produced a mix of simple and fairly complex classification trees that reflected the importance of Page Visits in overall distinction of successful and abandoned projects, and demonstrated the interplay of a few other variables in making finer distinctions. Unlike the Initiation Stage, our definition (and operationalization) of success and abandonment in the Growth Stage does not depend significantly on Downloads or Page Visits; therefore, in this analysis we utilize all numerical variables

(described in Chapter 7) as potential and valid predictors. However, in the analysis that follows, we will examine data subsets that help to mitigate any concerns about the definition of our dependent variable, as well as concerns about missing data.

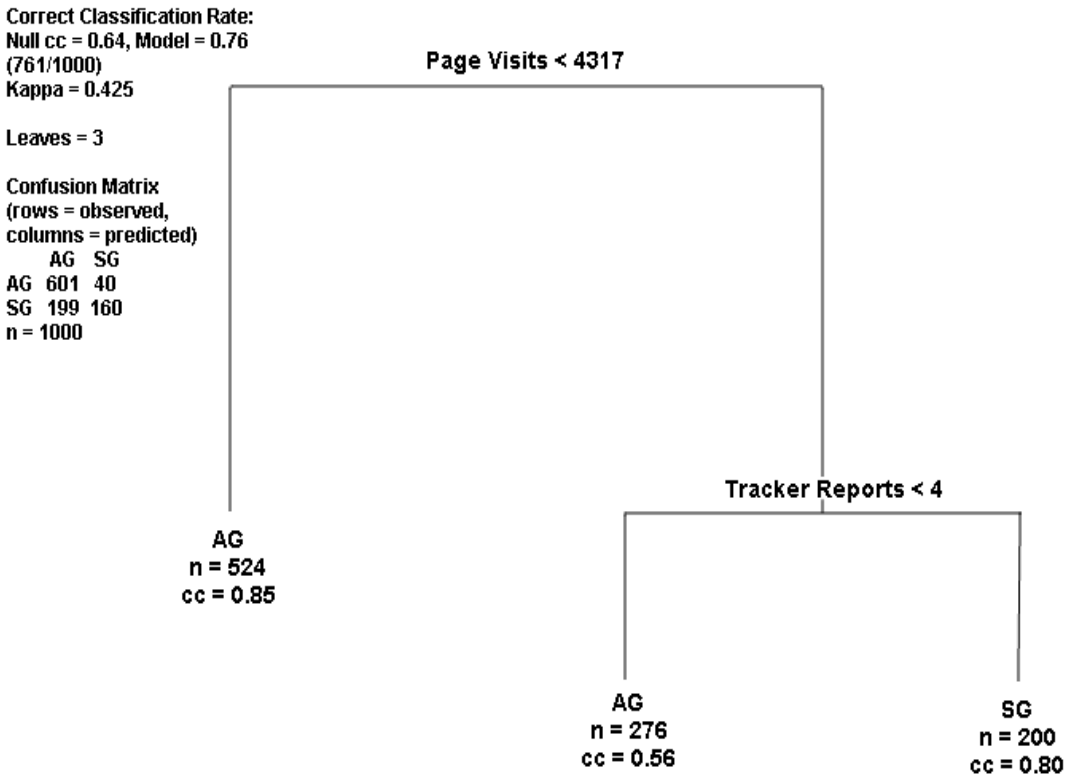

Figure 9.1
Growth Stage Tree using Sampling Strategy #4
(See Chapter 7, Table 7.5)

We developed the tree shown in Figure 9.1 using a random sample (n=1,000) of all successful and abandoned projects in the Growth Stage, and we included all independent variables. As shown in the Figure**,** 85% of the Abandoned in Growth (AG) projects in the first left hand node were correctly classified (the "cc" indicator in Figure 9.1) by separating the projects that had less than 4,317 Page Visits. It is worth noting that the remaining 15% of the

524 projects in this node that were successful but incorrectly classified as abandoned, may represent the small but useful projects that we intentionally included in our definition of success in the Growth Stage. On the right hand side of the tree (where projects had 4,317 or greater Page Visits), having less than four Tracker Reports further separated successful and abandoned projects. Overall, this parsimonious model in Figure 7.5 is moderately-accurate. It correctly classified 76% of the 1000 randomly selected projects, with Kappa score of 0.425.

The Confusion Matrix for this tree shows that the model classified 601 of 641 AG projects correctly, but only classified 160 of 359 SG projects successfully. In other words, for the projects in our data set, this model predicts abandonment in the Growth Stage much better than it predicts success in the Growth Stage. Page Visits (and Downloads) are associated with the interest of users and the utility of the project (see Table 4.1, H-P3a). Tracker Reports indicates that users are communicating about the project, and suggests that using such a facility to document project information (as suggested by Butler and Schmidt (2007)) is important. As we might expect, projects that have few Page Visits (and thus Downloads) and few Tracker Reports exhibit a lack of community involvement and are very often abandoned. As in the Initiation Stage, categorical variables are conspicuously missing from the tree.

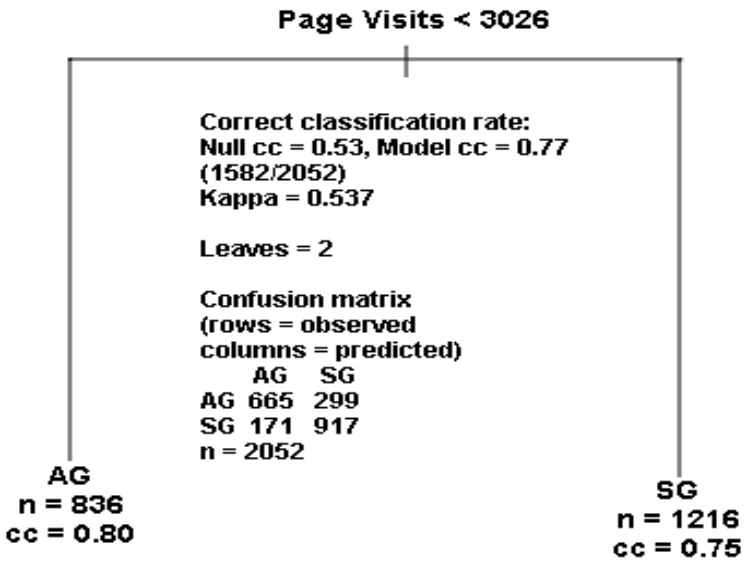

Figure 9.2
Growth Stage Tree using Sampling Strategy #5
(see Chapter 7, Table 7.5)

Next, similar to the analytic process we took for the Initiation Stage data, we decided to look at SF projects with Complete Observations (e.g., at least one value for each categorical variable group listed in Table 7.1) to investigate the sensitivity of the results from Figure 9.1 to missing data. We developed the tree shown in Figure 9.2 using complete observations only (n = 2052) and all variables. This tree is similar to the tree in Figure 9.1, but the first split on Page Visits was at a lower value (3026 visits) and Tracker Reports, as well as other variables, failed to appear in the tree. In order to investigate further, we created a Variable Importance Plot (not shown because of space limitations) for this data set, which showed the most important variables to be Page Visits, Downloads and Tracker Reports in that order. These

results are consistent with the results shown in Figure 9.1 and, like the corresponding example in the Initiations Stage in Chapter 8, support the idea that missing data do not seriously compromise our findings in the Growth Stage.

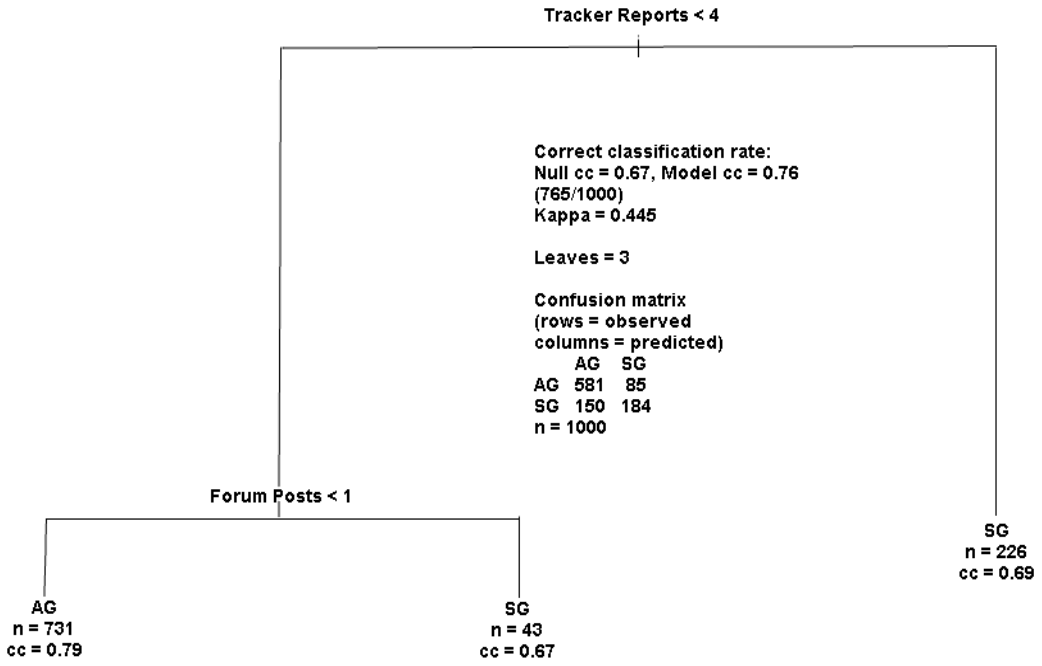

Figure 9.3
Growth Stage Tree using Sampling Strategy #6
(see Chapter 7, Table 7.5)

Back in the "Data Operationalization" section of Chapter 7, we noted that there was a slight connection between the operationalization of the success and abandonment dependent variable in the Growth Stage and the Downloads and Page Visits variables. To address concerns related to the connection of these to our dependent variable, we took a random

sample (n=1000) of all successful and abandoned projects in the Growth stage, but we excluded Page Visits and Downloads and included all other independent variables. This produced a tree (Figure 9.3) with the main split on Tracker Reports; a higher number of reports was a discriminator of successful projects. This main split correctly classified only 69% (cc=0.69) of the projects in the right-hand side (i.e., “Success” node) of the tree. Forum Posts further discriminated abandoned and successful projects in the left branch. Projects that had less than four Tracker Reports and zero Forum Posts were correctly classified as abandoned 79% of the time (cc=0.79). The model correctly classified 76% of the projects overall, with Kappa = 0.445. Like the earlier tree in Figure 9.1, the model in Figure 9.3 is much better at predicting abandonment (581 correct and 85 incorrect) than success (184 correct and 150 incorrect). The accuracy and Kappa values for this tree are almost the same as the values for the tree shown in Figure 9.1, so it appears that eliminating Page Visits and Downloads has little effect on overall accuracy. Furthermore, when Page Visits and Downloads are eliminated, we find that utility and community involvement, as represented by Tracker Reports and Forum Posts are the important characteristics that distinguish success from abandonment. In other words, whether or not we include Page Visits and Downloads, our results are conceptually the same.

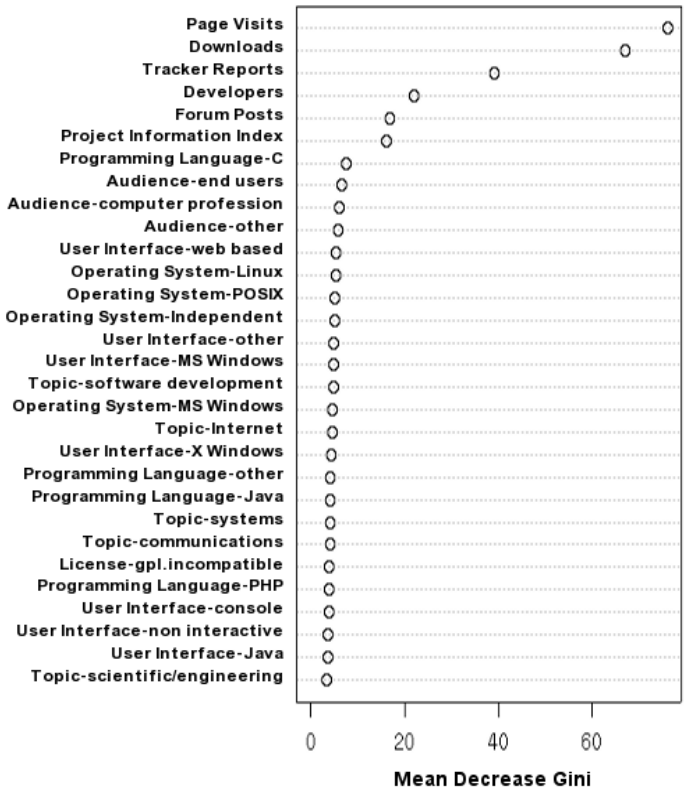

Figure 9.4.
Growth Stage Variable Importance Plot with Page Visits and Downloads Included.
Sampling Strategy #4 (n = 1000, see Table 7.5)

To complete this Results section, we have included a VIP using the same sampling strategy as used for Figure 9.1. As shown in Figure 9.4, and the trees presented above, Page Visits, Downloads and Tracker Reports have the most power to discriminate between success and abandonment in the Growth Stage. Developers, Forum Posts and the PII also show some ability to discriminate. We presented evidence, based on the Complete Observations dataset, that missing data does not substantially change our results, although we recognize that this dataset is a biased selection and does not necessarily prove our case. We also presented convincing evidence that Page Visits and Downloads are valid predictors in the Growth Stage. Overall, the accuracy of the representative Growth Stage model presented in

Figure 9.1 is only fair, which demonstrates the high variability of our data. Lastly, and perhaps most importantly, we found that our categorical variables have little power to discriminate between success and abandonment in the Growth Stage.

Discussion / Findings

So what does the above analysis say about the factors that lead to success and abandonment of open source projects in the Growth Stage? Similar to the Initiation Stage discussion in Chapter 8, we will reflect back to hypotheses and research questions posed in the theory chapters and in the independent variable discussion in the Data Description in Chapter 7, as well as introduce some new ideas.

Growth Stage Finding #1: Page Visits and Downloads lend support to the "software utility" hypothesis (H-P3a, Chapter 4, Table 4.1).

The trees in Figures 9.1, 9.2 reveal Page Visits as the primary splitting variable, suggesting that Sourceforge.net projects with higher numbers of people (at least in the 3-4 thousands) visiting the project page are associated with successful projects. The VIP in Figure 9.4 shows Downloads as the second most important splitting variable, after Page Visits. In the "Product Utility" section of Chapter 4, we noted that in both natural resource commons literature and the open source literature, it is recognized that commons will do better if, following Weinstock and Hissam's (2005: 157) articulation, the project provides "a general community service." Put simply, people find the software being produced as useful. In our previous discussion, we articulated three different "utility-related" hypotheses (see Chapter 4, Table 4.1, H-P3a, b and c). But Page Visits and Downloads are most closely

aligned with the first hypothesis (H-P3a): "Open source commons will be more often successful when the software produced has higher utility as perceived by the end-user community." In short, the larger the number of people interested in the software, the more likely the project will continue to be worked on. Our Page Visit and Download variable findings support this hypothesis.

Growth Stage Finding #2: Bug Tracker Reports and Forum Posts help to distinguish between successful and abandoned projects in the Growth Stage and lend further support to the "software utility" hypothesis (H-P3a, Chapter 4, Table 4.1). Moreover, they appear to be important collaborative infrastructure used in successful projects (RQ-P1, Chapter 4, Table 4.1).

Figures 9.1, 9.3, and 9.4 all highlight the importance of Tracker Reports and Forum Posts for distinguishing between successful and abandoned projects in the Growth Stage. Despite the fact that our Forum Posts data covers only an eleven month period and only 8,335 of the more than 107,000 projects have Forum Posts, a lack of Forum Posts[1] still proved to be important for discriminating Abandoned in Growth projects.

Our discussions in Chapter 7 noted that these variables captured both an element of "software utility," and also allowed us to test whether these particular communication mediums were important. Regarding the "utility hypothesis," described already in Finding #1, our results show that higher numbers of forum posts and bug tracker reports tend to be associated with successful projects. These variables capture effort on the part of end users and developers, and therefore provide an indicator that the project is of some use to them. Projects with no bugs or forum posts may indicate a number of issues, including, potentially, a

[1] We noted earlier that it may often be the case that projects use communication tools outside of SF, so we are not surprised to see such a low number of projects using these facilities.

disinterested user or developer community, poor project leadership, unskilled developers, a lack of responsiveness to users, or “no one home” because of a lack of financial support. The fact that the Bug Tracker and Forum Post variables stand out as useful discriminators of successful and abandoned projects along with our earlier findings regarding Page Views and Downloads, provide strong support for the “utility” hypothesis (H-P3a, Chapter 4, Table 4.1).

Moreover, these results support Fogel's (2007: 16) argument that the use of bug tracking is a useful indicator and signal of project “health.” The importance of Forum Posts could be an indication that the team is using that technology for building support or help documentation as described by Butler and Schmidt (2007), and points made by Ramsey (2007) who argues that one driver of success is that project discussions are transparent and open to everyone to see, and that it is well documented (forum posts are seen as one way to document software). This provides a partial answer to the “collaborative infrastructure” research question presented in Chapter 4 (RQ-P1, Table 4.1): the use of bug tracking and forums are associated with successful projects.

Growth Stage Finding #3. A substantial community of users and developers is associated with the majority of successful Growth Stage projects. There is support for Linus’ Law (RQ-C4, Chapter 4, Table 4.2).

In our efforts to define a robust dependent variable for success in Chapter 6, we worked to make sure the “small team” success cases (refer back to Chapter 3, Figure 3.2) were captured in our definitions. These are cases where there is a small developer community and a small user community, for example, in a very specialized field of software, like bioinformatics. So, how many of these kinds of projects exist in our SF data set? Using a simple database query, we found that only 16% of all SG projects in our Sourceforge data set

have all of the following statistics: fewer than 4,317 Page Visits, less than 2,000 downloads, less than four Tracker Reports, zero Forum Posts, and a Developer count less than four. These SG projects probably represent the approximate proportion of SG projects that are small and useful only to a limited audience—as we mentioned in our description of Figure 9.1 above.

Here's an important point. It could have turned out that the majority of SG projects were small. For example, most SG projects could have been efforts where a few academics worked on very specific software related to a particular field of research, or a few people worked on improving an obscure software game, or a small group of hobbyists worked on writing software that would allow kitchen appliances to be turned on or off via the Internet. However, this is clearly not the case; the majority of SG projects have a fairly large number of page visits to their websites, as well as a substantial number of downloads, tracker reports and forum posts.

In addition to representing the utility of the software, these variables (Page Visits, Tracker Reports, Forum Posts and Downloads[2]) also represent a substantial user and a larger developer community. Our classification tree analysis, along with the download data just presented, suggests a strong case that larger user communities are associated with Growth Stage success (this relates to Finding #4 below, regarding "Linus' Law").

The utility of the software is probably not the only factor drawing these people together.

[2] The "downloads" statistics are particularly interesting. We calculated the average number of downloads for Growth Stage projects along number of release stages (recall, our Success in Growth definition requires at least 3 releases). Growth Stage projects with only 1 release had, on average, 1066 downloads. Projects with two releases had an average of 2172 downloads. Projects classified as Indeterminate in Growth or Abandoned in Growth had an average of 2784 downloads. Projects that had exactly three releases and were classified as Successful in Growth had an average of 6869 downloads, and the average for all Successful in Growth classified projects (three or more releases) had on average, 81,951 downloads!

Building an open source software community most likely requires good leadership, as well as other factors not represented in our SF data. We discussed some of these factors in Finding #2 above. We will discuss these factors further, and our plans for broadening our model to include them, in the Conclusions to Part III.

Growth Stage Finding #4. Larger development teams are associated with Growth Stage success, although most successful Growth Stage projects are still comprised of small groups (e.g., 1-3 developers). Some limited support is provided for Linus' Law. (RQ-C4, Chapter 4, Table 4.2).

In Chapter 4, Table 4.2 (RQ-C4) we asked two questions: "Is there any notable direct relationship between group size and project success in open source?" and, "Is there any support for Olson/Brooks, Linus' Law or the Core Team theories? Recall that the Olson/Brooks theory suggests that small development teams will be more successful than larger ones. Linus' Law suggests that larger groups ("more eyes") will be more successful than smaller ones. Our "Core Team" theory suggests that group size won't matter, since most of the work is done by a small team anyway. We also noted that the relationship between group size and project success can be complex.

In our Classification Tree analysis, we find that the (number of) Developers variable is an important factor helping to distinguish between success and abandonment in the Growth Stage, as illustrated in the VIP in Figure 9.4. To shed more light on this finding, we provide Table 9.1 which reports the mean number of developers for each dependent variable class, and for projects at that have exactly one, two or three releases. Notice that this mean increases steadily as we move from the Indeterminate in Initiation (II) class (1.4 developers)

to projects with one release (1.6 developers; the first set of Success in Initiation projects), and then from the Indeterminate in Growth (IG) class (1.76 developers) to projects having two (1.86 developers) or three releases (2.06 developers) and finally to our Successful in Growth (SG) class (3.66 developers). Based on an Analysis of Variance, these differences are statistically significant (*P*-value less than or equal to 0.004). This increase indicates that as projects move through these stages, the developer counts tend to increase, and thus increasing developer counts occur temporally before projects achieve our definition of Successful in Growth.

In earlier work (Schweik et al., 2008) we published the data in Table 9.2 which shows the results of a simple logistic regression using Developers as the independent variable and the AG and SG classifications as the dependent variable for all AG and SG projects in our database (n=46,374). The coefficient value of 0.21084 in Table 9.2 translates into an odds ratio of 1.24. In other words, for each developer added to a project, the odds that the project will be successful increase 1.24 times. This logistic regression provided the basis of our reported finding of support for Linus' Law over the other two theories, but we also noted in our paper that many other factors, not included in this regression, were likely to contribute to the success of projects. The VIP in Figure 9.4 strengthens support for Linus' Law somewhat because the Developers variable turned out to be relatively more important than many other variables represented in our SF data set.

Based on this combined evidence, a fairly strong argument can now be made that increasing developer counts (group size) is a factor that "causes" success in the Growth Stage because: (1) higher developer counts, on average, occur before projects become

successful; (2) we can easily envision mechanisms that create this success (i.e., more developers = more code and possibly better code through peer review); and (3) success is correlated with higher developer counts. This argument – providing some support for Linus' Law – is strong but not proved because we are relying on averages to establish that higher team sizes occur before a project becomes SG (Table 9.1). Since we only have averages for projects in various stages at one time point, rather than longitudinal data for individual projects, we cannot be sure that *individual* projects that became successful in the Growth Stage increased their team sizes before they became successful (the chicken/egg problem discussed in Chapter 4). This being said, the number of SG projects in our database is large (15,782 projects), and it is hard to imagine that the average increase in developer counts observed for projects as they progress through the Growth Stage would not hold for many of the SG projects. This supports the findings of Madey and colleagues (2002) (noted in Chapter 4) that initial project success breeds more success as programmers join in because of a successful track-record.

At the same time, it is important to keep this conclusion in perspective: relatively few SG projects have large numbers of developers. Table 9.3 shows the distribution of the number of projects across all five of our dependent variable classes for different ranges of developer counts. This table also shows the success rate for projects in the SG and AG classes as the developer count increases. The main point to take from Table 9.3 is that the number of AG and SG projects with developer counts greater than ten is relatively small—only 1,261 of 46,374 projects.[3] Since Linus' Law seems to be based on a large number of

[3] At the same time, 1,261 projects with more than 10 people could be viewed as a fairly large number!

"eyeballs" looking at software code, the relatively small number of larger developer team projects means that although Linus' Law may be correct, it is probably not a major factor in the success of most Growth Stage projects. However, this finding, coupled with previous Finding #3 about user page visits, forum posts, tracker reports and downloads provides some strong evidence that larger developer and user communities leads to success in the Growth Stage. However, the importance of Linus' Law may be diminished by factors not represented in our SF data. These factors, like financial support, developer skill levels and others have been discussed in Part II and are noted in the Conclusions to this section of the book.

Growth Stage Finding #5: Collaborative success and abandonment are widely distributed across SF categories

Similar to our findings in the Initiation Stage, the general lack of importance of categorical variables (VIP in Figure 9.4) demonstrates that successful and abandoned open source projects are widely distributed across all categorical variables – intended audiences, operating systems, programming languages, project topics, and user interfaces, and the two main open source license types. As we noted in Chapter 8 for the Initiation Stage, this finding in the Growth Stage suggests to us that open source as a collaborative paradigm may entering a broader spectrum of software "topic areas," rather than focusing on "traditional open source" technologies, such as software projects around Gnu Linux, Apache, etc. This point is slightly speculative since we have no hard evidence to support it, but it aligns well with the "open source ecosystem" points made back in Chapter 2.

Growth Stage Finding #6: The "Computer Professionals" subcategory of Intended Audience does not help to distinguish between success and abandonment in the Growth Stage. This "non-finding" suggests a broadening of von Hippel's "user-driven innovation" in the more complex open source ecosystem, where it is no longer just programmers developing for their own (more technical) use, but in addition, programmers developing software for use by other types of end users.

The VIP in Figure 9.4 shows that no specific Intended Audience type (e.g., programmer, end-user, etc.) stands out as a discriminator of success in the Growth Stage. This finding is somewhat of a surprise to us; we were expecting projects geared specifically for programmers to stand out as more successful, as suggested by Eric von Hippel's "user-driven innovation" theory (H-C1, Chapter 3, Table 3.2). These results show that while it may still be that projects targeting programmers are often successful, there are many successful cases in the other Intended Audience categories as well. That is, projects specifically targeted for programmer use do not stand out from other categories as more successful.

As we noted in our discussions of a similar finding in the Initiation Stage (Chapter 8, Finding #5), we think this finding isn't a rejection of von Hippel's "user-driven innovation" theory (Chapter 3, Table 3.2, H-C1) but rather captures the impact of the broader ecosystem now involved in open source (Chapter 2), and the idea that "user driven" is no longer just confined to individual programmers developing technical software for their own use, but they are writing software for use by other types of end users. In some cases this can be situations where programmers are writing code, not for their personal need, but for the needs of the organization they work for (e.g., non-differentiating business needs as Perens, 2005, has suggested). A firm contributes or supports a programmer in-house to participate in the development of a project because that product is used by that firm for some business purpose. The intended audience would not necessarily then be Computer Professionals, but

could be in any Intended Audience domain (end users, business, other, government/non-profit).

Growth Stage Finding #7. The "Helping the Open Source Cause" hypothesis is not supported.

Like the Initiation Stage findings, none of the SF variables thought to capture elements of the "Helping the Open Source Cause" hypothesis (H-C5, Chapter 3, Table 3.2) – Operating System, User Interface, Database Environment, and Project License – stood out in the trees or Random Forest VIPs for the Growth Stage. If this hypothesis were correct, we would expect to see specific variable subcategories to stand out. For example, we would expect that operating system subcategories of Linux and/or BSD might help to distinguish successful projects from abandoned ones. Or, similarly, open source desktop interfaces like Gnome or KDE, or projects using open source database environments over proprietary databases, or GPL compatible licensed projects over GPL incompatible licensed projects. However, none of these categorical variables were identified as important variables distinguishing between success and abandonment in the Growth Stage. Hypothesis H-C5 in Tables 3.2 is not supported.

Growth Stage Finding #8. The "preferred technologies" hypothesis (H-P3c, Chapter 4, Table 4.1) is not supported.

No particular Operating System or Programming Language was found to be a factor in distinguishing between Growth Stage success and abandonment. We tested this hypothesis because, as we described in Chapters 3 and 4, some programmers participate in projects because of the learning incentive – they want to contribute to learn some technology that is

seen as "hot" or "cool" or valuable for their future career. However, none of the operating system subcategories or programming language categories helped to distinguish between successful and abandoned Growth Stage projects.[4] Hypothesis H-P3c (Chapter 4, Table 4.1) is not supported.

Growth Stage Finding #9. The "Critical Infrastructure" (H-P3b) hypothesis is not supported.

Recall from Table 7.1 in Chapter 7 that the Project Topic SF variable had a number of subcategories including: "communications," "database," "desktop environment," "education," "formats and protocols," games," "Internet," "security," "systems," and others. Hypothesis H-P3b (Chapter 4, Table 4.1) proposed that open source commons will be more successful when the software being produced is a component of "critical" or "foundational" infrastructure. If this is true, we would expect Project Topic subcategories like "systems," or "security" to help distinguish between successful and abandoned projects. However, in this dataset, neither of these subcategories mattered in making this distinction. Hypothesis H-P3b is not supported.

Growth Stage Finding #10: Growth Stage success and abandonment is not associated with GPL compatible or GPL incompatible licensing.

Back in Chapter 5, we presented a set of hypotheses and research questions related to the institutional design of open source commons (Table 5.5). One of the research questions (RQ-I3) was: "Does the choice in open source license affect the success and abandonment of an open source commons?" We noted in Chapter 7 that the only institutional variable contained in our SF dataset was project license with two categories: GPL compatible and

[4] The programming language "C" was the closest to being useful (see VIP Figure 9.4), although its Gini coefficient is quite low.

GPL incompatible licenses (and some projects could have both assigned to them). However, in evidence even stronger than our Initiation Stage findings (Figure 9.4), we find that neither of the two license categories, GPL compatible and GPL incompatible licenses, was a major factor for distinguishing between project success and abandonment.

Growth Stage Finding #11. The Project Information Index (capturing elements of leadership and utility) "shows up" as a variable that helps to distinguish between success and abandonment, but is much less important in the Growth Stage compared to the Initiation Stage (Chapter 8). The PII lends support for hypotheses H-P1 ("clearly defined vision"), H-P3a (software "utility"), and H-C6a and H-C6b (leadership).

Recall that in Chapter 7 and Chapter 8 we argued that the Project Information Index (PII) variable captured aspects of project leadership (Hypothesis H-P1 "clearly defined vision" in Table 4.1, and Hypotheses H-C6a, "excitement by doing," and H-C6b, "clearly defined goals" in Chapter 4, Table 4.2). While the PII does "show up" as important in the Growth Stage – it is really the last variable in both of the VIPs in Figures 9.4 and 9.5 to have a Gini coefficient that indicates an important effect on model results – it is much less important than it was in our analysis of the Initiation Stage in Chapter 8. There, you may recall, the PII was the most important splitting variable.

The question is why would the distinguishing characteristics of the PII be diminished in this stage, particularly if it is capturing elements of leadership? Looking at the data, apparently the PII gets to relatively high values in the Initiation Stage and then doesn't get much higher, on average, in the Growth Stage, so it doesn't discriminate between AG and SG projects as well as it does between AI and SI projects. While we can't be entirely sure about why this is, our strong inclination is that in the Initiation Stage low and high PII scores help to distinguish between "less serious" (e.g., a student project in a university for a class), and "more serious"

projects where people intend on continuing to work on them. So projects with higher PIIs in the Initiation Stage reflect a more serious endeavor (and an element of stronger leadership). This differs from the majority of the projects in the Growth Stage, where they have achieved a first release. There will be fewer cases in the Growth Stage where the project administrator has not taken the time to fill out the relevant categories of SF project metadata that are used to build the PII metric. While there is still variation in the PII, it is less in the Growth Stage compared to the Initiation Stage and therefore, the PII is diminished as a measure of leadership. In short, the leadership hypotheses of "clearly defined vision," "excitement by doing," and "clear articulated goals," as captured by the PII variable still help to distinguish between successful and abandoned projects – ones with higher PII scores tend to be more successful – but the variable is a less effective measure of leadership compared to the earlier Initiation Stage.

Conclusions

This chapter presented our results and findings of classification tree analysis for Sourceforge.net projects in the Growth Stage (see Figure 3.2, Chapter 3). In the section that follows, the conclusion to Part III, "An Exploratory Data Analysis of Sourceforge.net," we will reflect on these Growth Stage findings, as well as the ones for the Initiation Stage in Chapter 8, and what they suggest in uncovering "design principles" for open source commons. We will also describe what we think are the implications for designing a survey and empirical work for building a more complete model of open source success and abandonment – our goal in Part IV of this book.

Table 9.1 Mean # of Developers		
Project Class	**Mean # Developers**	**# of projects**
Abandoned in Initiation	1.70	37,320
Indeterminate in Initiation	1.40	13,342
Projects with exactly 1 release	1.62	20,347
Indeterminate in Growth	1.76	10,711
Abandoned in Growth	1.78	30,592
Projects with exactly two releases	1.86	10,248
Projects with exactly 3 releases	2.06	5,917
Successful in Growth	3.66	15,782

Table 9.2 **Estimated Coefficients, Standard Errors, z-Scores, Two-tailed p-values, and 95% Confidence Intervals for Simple Logistic Regression of Number of Developers Influence on Project Success or Abandonment (n: 46,374)**					
Variable Coeff.	**Std. Err.**	**z**	**P>|z|**	**95% CI**	
Developers 0.21084	0.00473	44.58	<2E-16***	.202	.220
Constant -1.14993	0.01440	-79.88	<2E-16***	-1.18	-1.12
Significance codes: *** 0.001 C: 0.638; R2: 0.088					

Table 9.3
Number of Projects in each Class for Various Developer Counts

	Developer Count				
	<5	**5-10**	**11-20**	**21-30**	**>30**
Class	**# of Projects**	**# of Projects**	**# of Projects**	**# of Projects**	**# of Projects**
AI	35492	1563	224	25	16
II	13026	273	38	3	2
IG	10120	489	81	9	12
AG	28843	1500	212	25	12
SG	12541	2229	709	164	139
Success Rate SG/(AG+SG)	0.3	0.6	0.77	0.87	0.92
Total # of Projects	100022	6054	1264	226	181