Presented by Charles Alvarenga

UFMG – Departamento de Ciência da Computação

lab-soft
software engineering laboratory

Analyze This! 145 Questions for Data Scientists in Software Engineering

Andrew Begel
Microsoft Research

Thomas Zimmermann
Microsoft Research

Proceedings of the International Conference on Software Engineering (ICSE), 2014

Structure

- Introduction and purpose
- Methodology
- Results
- Implications
- Conclusion

Introduction and purpose

Businesses of all types commonly **use analytics** to better reach and **understand their customers**.

Many **software engineering researchers** have argued for more **use of data for decision-making**.

Even **sporting teams** use **analytics** to **improve their performance**.

Bertrand Meyer (ESEC/FSE 2013) **emphasized** the need for the software engineering community to **become more data-driven.**

Introduction and purpose

Results from **2 surveys** related to **data science** applied to **software engineering**.

1° survey: **questions** that software engineers would like **data scientists** to **investigate** about **software**; about software processes and practices, and about software engineers.

1° survey resulted in a list of **145 questions** grouped into **12 categories**.

2° survey asked a different pool of software engineers to **rate these 145 questions** and identify the **most important** ones to work on first.

Introduction and purpose

Results can **help researchers, practitioners, and educators** to more easily **focus their efforts** on topics that are **important** to the **software industry**.

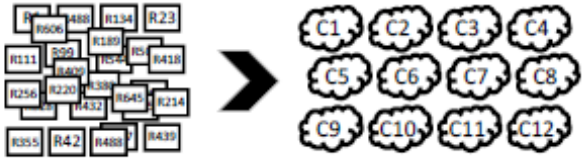
Methodology

❶ SURVEY — 203 participants, 728 response items R1..R728

Suppose you could work with a team of data scientists and data analysts who specialize in studying how software is developed. Please list up to five questions you would like them to answer.

❷ CATEGORIES — 679 questions in 12 categories C1..C12

We used an open card sort to group questions into categories.



❸ DESCRIPTIVE QUESTIONS — 145 questions Q1..Q145

We summarized each category with a set of descriptive questions.

❹ SURVEY — 16,765 ratings by 607 participants

We used a split questionnaire design. Each participant received a subset of the questions Q1..Q145 (on average 27.6) and was asked:

In your opinion, how important is it to have a software data analytics team answer this question?
[Essential | Worthwhile | Unimportant | Unwise | I don't understand]

❺ TOP/BOTTOM RANKED QUESTIONS

❻ DIFFERENCES IN DEMOGRAPHICS

Figure 1. Overview of the research methodology.

Methodology

① SURVEY

- September 2012;
- They asked a random sample of 1,500 Microsoft engineers;
- Chosen by discipline;

- Question:

Suppose you could work with a team of data scientists and data analysts who specialize in studying how software is developed. Please list up to five questions you would like them to answer.

- Additionally asked:
  - “Why do you want to know?”
  - “What would you do with the answer?”

Methodology

① SURVEY

- Pilot surveys to 25 and 75 Microsoft engineers:
  - used the responses to improve the questions.

- Monetary incentives: enter a raffle for a $250 Visa Check Card.

- 203 participants, 728 response.
  - 36.5% developers;
  - 38.9% testers;
  - 22.7% program managers.

Methodology

② CATEGORIES

- Card sorting is a technique that is used to create mental models and derive taxonomies from data;
- 728 -> 679 questions;
- 49 questions discarded;
- 12 categories.

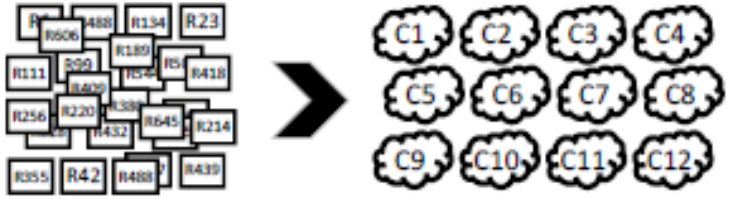


Figure 2. Photo of some of the subcategories that emerged during the open card sort.

Methodology

③ DESCRIPTIVE QUESTIONS

- Summarized each category with a set of descriptive questions.
- Hard questions to understand;
- Same or similar questions;
- Some very detailed questions, some not;
- Described according to the category.

- 679 -> 145 questions.

Methodology

③ DESCRIPTIVE QUESTIONS

Table 1. The 12 categories identified in the card sort.

Category		Cards		Subcategories	Descriptive Questions
Bug Measurements	BUG	23	3%	4	7
Development Practices	DP	117	16%	13	28
Development Best Practices	BEST	65	9%	6	9
Testing practices	TP	101	14%	5	20
Evaluating Quality	EQ	47	6%	6	16
Services	SVC	42	6%	2	8
Customers and Requirements	CR	44	6%	5	9
Software Development Lifecycle	SL	32	4%	3	7
Software Development Process	PROC	47	6%	3	14
Productivity	PROD	57	8%	5	13
Teams and Collaboration	TC	73	10%	7	11
Reuse and Shared Components	RSC	31	4%	1	3
Discarded Cards		*49*	*7%*		
Total Cards		728	100%	60	145

Methodology

④ SURVEY

- Rank and prioritize the 145 descriptive questions;
- 10 components (blocks) of descriptive questions:
  - Bug Measurements with Development Best Practices
  - Productivity with Reuse and Shared Components
  - Services with Software Development Lifecycle
- Little information is lost;
- Each participant received 2 blocks with questions (Q1….Q145);

- Question:

In your opinion, how important is it to have a software data analytics team answer this question?

[Essential | Worthwhile | Unimportant | Unwise | I don't understand]

Methodology

④ SURVEY

- Response scale inspired by Prof. Noriaki Kano:
  - must have (Essential);
  - nice to have (Worthwhile)
  - inconsiderate (Unwise).
- 2,500 engineers (randomly chosen by discipline);
- They not have participated in the initial search;
- Monetary incentives: enter a raffle for a $250 Visa Check Card.

- 607 participants.
- 16.765 ratings.

Methodology

⑤ TOP/BOTTOM RANKED QUESTIONS

- Percentage of "**Essential**" responses among all responses

$$\frac{\text{Essential}}{\text{Essential} + \text{Worthwhile} + \text{Unimportant} + \text{Unwise}}$$

- Percentage of "Essential" and "Worthwhile" responses among all responses (to which we refer as **Worthwhile**+)

$$\frac{\text{Essential} + \text{Worthwhile}}{\text{Essential} + \text{Worthwhile} + \text{Unimportant} + \text{Unwise}}$$

- Percentage of "**Unwise**" responses among all responses

$$\frac{\text{Unwise}}{\text{Essential} + \text{Worthwhile} + \text{Unimportant} + \text{Unwise}}$$

Methodology

⑥ RATING DIFFERENCES BY DEMOGRAPHICS

- Not anonymous.

- Discipline: Development, Testing, Program Management;
- Region: Asia, Europe, North America, Other;
- Current Role: Manager; Individual Contributor (IC);
- Years as Manager or worked in a management role;
- Years at Microsoft.

Results

→ Bug Measurements (BUG)

- Where they are found in code;
- Which bugs are most commonly made by developers;
- Where in the application lifecycle they are found;
- How much it costs to fix bugs.

"*For each bug, at what stage in the development cycle was the bug found and what stage was it introduced?*

"Are there categories of mistakes that people make that result in hotfixes after the fact? For example, date/time mistakes, or memory allocation mistakes, etc."

Results

→ Development Practices (DP)

- Debugging;
- Refactoring;
- Repositories;
- Code reviewing;
- Commenting;
- Documenting code.

“We are often told that code comments are important to assist others understand code, but is this really true?”

“What are the provably best metrics to track during design/planning to determine project complexity and cost? What is the expected margin of error?”

Results

→ Development Best Practices (BEST)

- The right technique for migrating between software versions;
- The best way to track work items;
- The right time to use formal methods for software analysis;
- Which criteria should influence the decision to use a particular programming language or API.

“Which coding guidelines/patterns have the most effect on code quality (e.g. small functions, lower complexity metrics, removal of duplicate code)?”

“What are the best and worst practices [of] teams that lose deadlines?"

Results

→ **Testing Practices (TP)**

- Test automation;
- Testing strategies;
- Unit testing;
- Test processes.

💬 *“What is the cost and benefit analysis of the different levels of testing i.e. unit testing vs. component testing vs. integration testing vs. business process testing?”*

💬 *“How much percentage of development time needs to be spent on unit testing to ensure quality?*

Results

→ Evaluating Quality (EQ)

- Optimization tradeoffs;
- The best metrics;
- Complexity metrics;
- Code duplication.

"How much do existing bugs in a code base affect the velocity with which new features can be added to that code base?"

"How much time does a feature of a certain complexity need before it can be considered certain?"

Results

→ Services (SVC)

- How to change development and testing practices when migrating software to the cloud.

“What are the KPIs that are used in managing our on-line services today?”

“Do distributed systems scale better than centralized systems with vertical/horizontal scaling?”

Results

→ Customers and Requirements (CR)

- The most important features;
- Impact of testing in production.

“How many features of the s/w are used by people and at what percentage?

“How do we determine what problems our customers are having with our software?

Results

→ Software Development Lifecycle (SL)

- Time as a vital factor in designing the software lifecycle;
- How development time should be allocated between planning, design, coding, and testing.

“Do short release cycles result in a better quality product?

“How detailed should design documents be before starting to code?”

Results

→ Software Development Process (PROC)

- Agile is better than Waterfall?
- Benefits of pair programming;
- Methodologies that work better for cloud services.

“How is Agile process model more or less efficient in when it comes to number of bugs?”

“Which software development model(s) are most successful in producing great software?”

Results

→ **Productivity (PROD)**

- Different metrics for measuring the productivity of software developers;
- Relationship between individual productivity and team;
- How to monitor their own or their team's productivity.

"How do we measure the productivity of our engineers? Which engineer is more productive?"

"How does productivity between open work spaces and traditional offices?"

Results

→ Teams and Collaboration (TC)

- How many people of various development roles should be on a team?
- Which practices could improve sharing and collaboration?

“How can we share knowledge more effectively to code faster?”

“How much time/week do you spend updating/sending/reporting status to other people?”

Results

→ Reuse and Shared Components (RSC)

- The smallest category of card sort;

- When should code be written from zero vs. reused from another codebase?
- The cost of searching for the right code;
- Potential impact caused by just linking to it vs. copying.

Results

→ **Reuse and Shared Components (RSC)**

- *“What’s the best way to reuse an existing library if it combines the functionality we are looking for, but does not combines the architecture of our project?”*
- *“When is it best to create your own UI model rather than adopt a standard that might be familiar, but not fit your use exactly right?”*
- *“Do shared codebases actually reduce developer productivity and innovation rates at the company?”*
- *“How can I find out if someone else has already solved a similar problem before?”*

Results

Table 2. Questions with the highest "Essential" and "Worthwhile or higher" percentages.

		Question	Category	Percentages			Rank		
				Essential	Worthwhile+	Unwise	**Essential**	Worthwhile+	Unwise
ıllı	Q27	How do users typically use my application?	DP	80.0%	99.2%	0.83%	1	1	128
ıllı	Q18	What parts of a software product are most used and/or loved by customers?	CR	72.0%	98.5%	0.00%	2	2	130
★	Q50	How effective are the quality gates we run at checkin?	DP	62.4%	96.6%	0.85%	3	6	125
🛠	Q115	How can we improve collaboration and sharing between teams?	TC	54.5%	96.4%	0.00%	4	8	130
★	Q86	What are best key performance indicators (KPIs) for monitoring services?	SVC	53.2%	93.6%	0.92%	5	12	106
★	Q40	What is the impact of a code change or requirements change to the project and tests?	DP	52.1%	94.0%	0.00%	6	10	130
🛠	Q74	What is the impact of tools on productivity?	PROD	50.5%	97.2%	0.92%	7	4	106
🛠	Q84	How do I avoid reinventing the wheel by sharing and/or searching for code?	RSC	50.0%	90.9%	0.91%	8	19	108
ıllı	Q28	What are the common patterns of execution in my application?	DP	48.7%	96.6%	0.84%	9	5	127
ıllı	Q66	How well does test coverage correspond to actual code usage by our customers?	EQ	48.7%	92.0%	0.00%	10	16	130
★	Q42	What tools can help us measure and estimate the risk associated with code changes?	DP	47.8%	92.2%	0.00%	11	15	130
★	Q59	What are effective metrics for ship quality?	EQ	47.8%	96.5%	1.77%	12	7	91
🛠	Q100	How much do design changes cost us and how can we reduce their risk?	SL	46.6%	94.8%	0.86%	13	9	123
ıllı	Q19	What are the best ways to change a product's features without losing customers?	CR	46.2%	92.3%	1.54%	14	14	103
ıllı	Q131	Which test strategies find the most impactful bugs (e.g., assertions, in-circuit testing, A/B testing)?	TP	44.5%	91.8%	0.91%	15	18	108
🛠	Q83	When should I write code from scratch vs. reuse legacy code?	RSC	44.5%	84.5%	3.64%	15	45	37
	Q1	What is the impact and/or cost of findings bugs at a certain stage in the development cycle?	BUG	43.1%	87.9%	2.59%	17	28	68
ıllı	Q92	What is the tradeoff between releasing more features or releasing more often?	SVC	42.5%	79.6%	0.00%	18	64	130
🛠	Q2	What kinds of mistakes do developers make in their software? Which ones are the most common?	BUG	41.7%	98.3%	0.00%	19	3	130
ıllı	Q25	How important is a particular requirement?	CR	41.7%	87.4%	2.36%	20	32	73
★	Q60	How should we use metrics to help us decide when a feature is good enough to release (or poor enough to cancel)?	EQ	41.1%	90.2%	3.57%	23	20	41
ıllı	Q17	What is the best way to collect customer feedback?	CR	39.8%	93.0%	1.56%	26	13	101
🛠	Q3	In what places in their software code do developers make the most mistakes?	BUG	35.0%	94.0%	0.00%	41	10	130
★	Q102	What kinds of problems happen because there is too much software process?	PROC	30.1%	92.0%	0.88%	57	16	116

Results

Table 3. Questions with the highest "Unwise" percentage (opposition).

	Question	Category	Percentages			Rank		
			Essential	Worthwhile+	**Unwise**	Essential	Worthwhile+	**Unwise**
Q72	Which individual measures correlate with employee productivity (e.g., employee age, tenure, engineering skills, education, promotion velocity, IQ)?	PROD	7.3%	44.5%	25.5%	142	141	1
Q71	Which coding measures correlate with employee productivity (e.g., lines of code, time it takes to build the software, a particular tool set, pair programming, number of hours of coding per day, language)?	PROD	15.6%	56.9%	22.0%	126	131	2
Q75	What metrics can be used to compare employees?	PROD	19.4%	67.6%	21.3%	110	112	3
Q70	How can we measure the productivity of a Microsoft employee?	PROD	19.1%	70.9%	20.9%	113	104	4
Q6	Is the number of bugs a good measure of developer effectiveness?	BUG	16.4%	54.3%	17.2%	122	136	5
Q128	Can I generate 100% test coverage?	TP	15.3%	44.1%	14.4%	127	142	6
Q113	Who should be in charge of creating and maintaining a consistent company-wide software process and tool chain?	PROC	21.9%	55.3%	12.3%	93	134	7
Q112	What are the benefits of a consistent, company-wide software process and tool chain?	PROC	25.2%	78.3%	10.4%	79	72	8
Q34	When are code comments worth the effort to write them?	DP	7.9%	41.2%	9.6%	141	143	9
Q24	How much time and money does it cost to add customer input into your design?	CR	15.9%	68.2%	8.3%	124	111	10

Results

Table 4. Statistically significant rating differences by demographics.
The demographic with the highest rating is highlighted in bold. Questions that are also in Table 2 are shown in *italics*.

				Discipline		
	Question	Category	Response	Dev	Test	PM
Q5	How many new bugs are introduced for every bug that is fixed?	BUG	Essential	27.3%	**41.9%**	12.5%
Q10	When should we migrate our code from one version of a library to the next?	BEST	Essential	**32.6%**	16.7%	5.1%
Q20	How much value do customers place on backward compatibility?	CR	Essential	14.3%	**47.1%**	18.3%
Q21	What is the tradeoff between frequency and high-quality when releasing software?	CR	Essential	22.9%	**48.5%**	14.5%
Q42	*What tools can help us measure and estimate the risk associated with code changes? (Essential #11)*	DP	Essential	34.5%	**70.6%**	40.4%
Q121	How can we make it easier for people to find and use commonly used tools?	TC	Essential	27.3%	**48.6%**	20.0%
Q24	*How much time and money does it cost to add customer input into your design? (Unwise #10)*	CR	Worthwhile+	62.9%	**88.2%**	60.3%
Q113	*Who should be in charge of creating and maintaining a consistent company-wide software process and tool chain? (Unwise #7)*	PROC	Worthwhile+	60.0%	**71.9%**	40.4%
Q132	How should we handle test redundancy and/or duplicate tests?	TP	Worthwhile+	48.6%	**81.1%**	47.4%
Q134	Should we develop a separate test suite for servicing a product after we ship it?	TP	Worthwhile+	32.4%	**67.6%**	**67.6%**

				Management Role	
				Manager	Ind. Contributor
Q29	How much legacy code is in my codebase?	DP	Worthwhile+	36.7%	**65.2%**
Q31	When in the development cycle should we test performance?	DP	Worthwhile+	63.3%	**81.4%**
Q70	*How can we measure the productivity of a Microsoft employee? (Unwise #4)*	PROD	Worthwhile+	57.1%	**77.3%**
Q120	What are the most commonly used tools on a software team?	TC	Worthwhile+	**95.8%**	67.8%

Results

				Region		
				Asia	Europe	North America
Q70	*How can we measure the productivity of a Microsoft employee? (Unwise #4)*	PROD	Essential	**52.9%**	30.0%	11.0%
Q104	How do software methodologies affect the success and customer satisfaction of shrinkwrapped and service-oriented products?	PROC	Essential	**52.9%**	10.0%	24.7%
Q128	*Can I generate 100% test coverage? (Unwise #6)*	TP	Essential	**60.0%**	0.0%	9.0%
Q129	What is the effectiveness, reliability, and cost of automated testing?	TP	Essential	**71.4%**	12.5%	23.6%

Implications

RESEARCH

- Help guide **academic researchers** towards novel research **problems** to be **studied and analyzed**, as well as algorithms, processes, and tools to be built and tested.

PRACTICE

- Any **preexisting tools** that could solve these problems may have been unknown to or **unusable** by some of those respondents.

EDUCATION

- Students undergraduate and graduate, even beginning professional software engineers **try to adapt the practices and processes** they **learned** in **university** to the realities they **find** in **industry** .

Conclusion

- **Inspire** similar research projects;
- **Growing** demand for **data scientists**;
- **More research** is needed to better **understand how people make decisions** in software projects and **what data** and **tools** they need;
- **Think more** about the **consumer of analyses** and not just the producers.

Appendix to Analyze This! 145 Questions for Data Scientists in Software Engineering.

Technical Report, MSR-TR-2013-84

http://aka.ms/145Questions

Presented by Charles Alvarenga

UFMG – Departamento de Ciência da Computação

lab-soft
software engineering laboratory

Analyze This! 145 Questions for Data Scientists in Software Engineering

Andrew Begel
Microsoft Research

Thomas Zimmermann
Microsoft Research