PRACTICAL COMMUNITY MANAGEMENT

# LESSONS LEARNED WHILE WORKING ON STACK OVERFLOW

NICOLAS CHABANOVSKY

Practical Community Management

# Lessons Learned While Working on Stack Overflow

Nicolas Chabanovsky

Contact information:
nicolas.chabanovsky@gmail.com
practical-cm.com

ISBN: 9798323978502
First Edition: April 2024

## Dedication

To volunteers, caring people who spend their lives on doing good for others without expecting anything in return.

# Contents

# Introduction

My story related to online communities began at the end of 2010. When I was working as a software developer at Motorola, my friends and I launched HashCode.ru, a Q&A website for Russian-speaking developers inspired by Stack Overflow. Initially, we launched the site out of curiosity, but after a couple of months everything changed and we began to work seriously on building the community. A year and a half later, HashCode had become one of the most popular Q&A sites for software developers who speak Russian. In 2014, the site was acquired by Stack Overflow and became one of the four international Stack Overflow communities. I continued working on Stack Overflow in Russian as part of the company. The Russian-speaking community continued to grow, growing faster than any other Stack Exchange community at the time. Subsequently, among all Stack Exchange communities Stack Overflow in Russian became third by the number of questions per day, the main metric of activity. After that, I continued to help the community management team, first as a leader of all international sites, and then I became fully focused on community data analytics and community strategy.

While working on Stack Overflow communities, I read every book on community management I could find. I noticed that they all shared one common trait: the ideas presented in them are general. The materials I found mostly dealt with abstract concepts and did not explain in any way how to apply the presented knowledge in practice. Let’s face it, there’s a big difference between knowing that there is such a thing as a sense of belonging and being clear about the initiatives you need to run to impact the sense of belonging among users in your community. Books on community management were interesting to read, but the practical advice extracted from each of them was minimal. It seems that now, 10 years later, the situation has not changed. The community manager still has the same three main sources of professional applied knowledge: literature on generally accepted disciplines such as sociology and social psychology, colleagues, and trial and error. I will assume that not everyone has the time to acquire knowledge from an abundance of books, nor does everyone have access to experienced colleagues.

Community management is not rocket science. I don't see why community managers should all gain experience from making the same mistakes. Progress is only possible when we learn from each other. This book is my contribution to our collective step in that direction. The true value of specific knowledge is revealed only in the context of its application. Therefore, the basis of the book is the most important and obscure applied tips, which, in my view, helped the community management team to grow the Stack Overflow communities the most.

The book consists of three parts.

When you launch a community, your main goal is to attract first-time users. The problem is that everyone only wants to participate in successful communities, but for a community to be successful, it needs people to participate in it. This is a typical chicken and egg situation. In the first part of the book, we will talk about how my friends and I launched HashCode.ru, a copy of Stack Overflow for Russian-speaking developers and how we attracted the first users to the community when no one knew about it.

In the second part of the book, I will succinctly, without further ado and avoiding fluff, talk about what community management is. Basic definitions, how to work with volunteers, what approaches to community growth there are and how to use them, how to build communication with the users, and much more. If in the first part, we talked about how to attract first-time users, then in the second part we will focus on how to make sure that those who come, stay and help you attract more new users. Suppose you have ever thought about creating your own online community, or have already created one, but are having any difficulty with it. This chapter will most likely be useful to you.

Activity in the Stack Overflow community has been declining over the past few years. At the time of writing (late 2023), activity in Stack Overflow is comparable to the level back in 2011 (the site was launched in late 2008). In my view, the reasons for this lie in how currently the management of community processes has not been organized wisely. In the third part of the book, we are going to look at some of the most critical mistakes that, in my view, led to the current state of affairs in the community. This section may be of interest to leaders of community management teams who are involved in process design in companies whose main product is the community itself.

While at Stack Overflow I was working on Meta posts, I usually managed to write about ten pages of text per day. Then I spent another two to three days cutting down ten pages to one and a half pages, eliminating all but the absolutely essential thoughts. This book was written on a similar principle. I tried to take only the most important and unobvious, excluding common knowledge to make the book as short as possible, but still interesting to read.

In the book, I tried to reflect on the result of my 14 years of experience in building and growing online communities. Everything I say in the book, I say on my behalf. My views may differ from those of other members of the Stack Overflow community management team, the company’s founders, and the current management. I see great value in this.

My personal goal, as an author, is to make sure that after reading this book, you fall even more in love with creating and growing online communities, and your users participate in them with even greater pleasure.

I hope you enjoy reading the book.

# Part 1

## To do pull ups, you need to do pull ups

Nobody knows the future because it is not defined. People who have experience and knowledge can make hypotheses about what the future will look like, but no one can say for sure whether your community will be successful or not. The optimal strategy in this case is to think about whether the community you are thinking of is worth your time, effort and money, and if it’s worth it, then launch and grow the community as efficiently as possible, as quickly as possible, paying minimal attention to other people’s opinions.

## How it got started: Doing the right thing is more important than doing things right

I started my professional journey as a software developer in a small company consisting of 5 people. The company’s main product was a distributed program that could perform various computing tasks on a large number of machines simultaneously. It was a kind of advanced technology for that time, so the work was very interesting, but not stable, since this was not the primary business for the founders. After working there for about a year, I decided it was time to move on and get some real experience working for a big company. My choice fell on Motorola which had a software development office in St. Petersburg consisting of more than 700 people.

Working for a large company like Motorola has significant advantages like being introduced to refined industrial software development processes and the opportunity to work alongside a huge number of experienced colleagues. At the same time, you suffer a huge disadvantage, which largely determined my future fate. Any employee of a large company is just a small cog inside a huge mechanism. An employee’s task is to do their highly specialized part of the job well, forgetting about everything that goes beyond this scope. In order to somehow continue our professional development, several guys and I began to meet regularly to discuss new technologies after work in the office kitchen over tea. At that time (2009–2010), a lot of interesting things were happening. Android was starting to gain momentum. Google began to actively develop its Chrome browser and an API for it, machine learning began to penetrate everyday life. At some point, we decided to transform our tea discussion club into an applied research club. We started doing small projects on weekends using technologies that we talked about during the week.

*Note: Subsequently, a few commercial projects were launched based on our research. The ability to experiment with new technologies is important even in large companies.*

Studying new technologies involves a constant search for information about them. The realities of 2010 were such that it was difficult to find information about anything. For example, if you were faced with a problem compiling the code for your project, finding a solution to the problem could easily take a whole week. During this time, you usually read the compiler documentation, email correspondence from other developers having similar problems posted online, and several dozen threads on online forums. Finding information was a real manual art. If you found something once, there was no guarantee that you could find it again. Information about new technologies was even more difficult to find. Therefore, within the framework of our discussion club, we agreed that when someone found something interesting, they would save this information to a file on their local computer so that we could further exchange the accumulated knowledge.

After about six months of our discussion club's existence, we began to notice that when we were searching for information about new technologies of that time (Android, WPF, GWT, etc.), we always ended up on the same website. It was Stack Overflow. At one of our tea meetings, we thought, why do we exchange files with our personal knowledge bases if we can launch a website like Stack Overflow but just for us in the Russian language and publish all our findings there? After a short discussion, that's what we decided to do. Moreover, at that time, launching a website in itself was an interesting technical task, especially for us since we worked on operating systems for embedded systems. Creating a website was something from another universe and quite enjoyable. It didn't seem realistic to write our own website from scratch, so we found a good open-source engine and after a month of working on weekends, we launched the first version of what today is Stack Overflow in Russian.

Initially, we launched our website just for the sake of sharing knowledge, but at some point, as a joke, we decided to calculate approximately how much a typical online forum about programming in the Russian language earns (and how much we could earn!) According to our most minimal estimates, the income was ten times higher than the salary for a developer in a good company. This was such a huge revelation for us that we discussed this topic for several more weeks. One day it turned into the discussion of another question. Why not try to turn our weekend project into a full-fledged Q&A forum for programmers? We are experienced software developers and we can make a high-quality product. What could go wrong? It seemed to us that if we just tried, the project would become successful, and we would become if not millionaires at least ten times higher paid developers.

*Note: Further in the book I often mention HashCode as an example of an online community. It is important to note that everything that is true for HashCode is true for any international Stack Overflow site and in most cases is true for Stack Overflow in English itself.*

*Note: Chances are, unless you are a programmer, you might not know what Stack Overflow is. Stack Overflow is a Q&A site for software developers, where anyone can ask their own question or answer other people's questions. The Stack Overflow site is part of the Stack Exchange network, which consists of more than 170 sites on a variety of topics. When someone says "Stack Overflow" they default to "Stack Overflow in English". In addition to Stack Overflow in English, there are four more Stack Overflow sites in other languages on the Stack Exchange network. In Portuguese, Japanese, Russian and Spanish. We call them "international sites" or "international Stack Overflows". All international communities are absolutely self-sufficient and do not depend in any way on Stack Overflow in English.*

# 1. Recruiting a critical mass of users

Once we decided to make a full-fledged website, we spent a few more months refining what we had into a version that could be called an MVP. As soon as the updated website was launched, we immediately uploaded the accumulated knowledge base onto it and began to wait for the first happy users of our community… A day, two days and then one week passed, but no new questions appeared on the website, no answers, there were not even any registrations. We began to understand that for the system to work, simply uploading the knowledge base to the website is not enough and we needed to do something else. Thus, one of the most difficult periods in the life of our community began. The period of initial growth, when it is necessary to solve the chicken and egg situation: an online community is successful when there are people who participate in it, but people usually want to participate in communities that are already successful.

## Seed the initial content that encourages users to take some actions

So, by this time we had a working website with an initial knowledge base of about two hundred questions. Our main goal was to attract active users who would ask questions themselves and answer questions from others without our direct participation. To achieve this, we decided to pursue two development strategies: continue writing interesting questions and answers and create opportunities for users to act.

## 1. Create interesting content

Folks at Stack Overflow always openly said that their main source of traffic was search engines, which gave them more than 95% of all pageviews. The economics of the project was such that Stack Overflow consisted of two types of questions. The first group consisted of a huge number of very specific niche questions that on average received only a few dozen views each. Their second group consisted of a small number of highly popular questions that were viewed tens of thousands of times.

In order for search engines to send us users, it was necessary to create the content that these users were looking for. Our plan was to create something that was in great demand. Armed with search query statistics services, we tracked what was trending at that time and wrote questions and answers on these topics.

When we were publishing content on our website, we used a little trick. We posted the content from many different accounts. There were two reasons for this:

1. When new users come to a website without knowing anything about it, they infer the culture of communication and behavioral norms of the community from the existing content. By publishing content from different accounts, we showed what kind of culture we would like to see in the community.
2. Posting Q&A content from one account looks unnatural and can put people off.

Using multiple accounts to populate a website with initial content is a common practice. The most important thing in this case is to be honest with yourself and other people and do not use accounts created to fill the website to gain votes, reputation or support your own opinion.

*Note: Like attracts like. We were very attentive to the quality of the content we created, and this greatly helped us in the future. The first users wrote their posts using a communication style that was very close to ours. There were practically no violations, moderation took minimal time.*

We posted five to ten questions a day. In addition, I asked my friends to ask questions on our website when they encountered any problem at work. At one of these moments, I realized how it is sometimes difficult for a person to create a post on a public website. One day, a close friend of mine, with whom I worked at the same company, was faced with a compilation problem and asked me to help him. I had already solved a similar problem and knew the answer. I told him that I would help him on the website as soon as he asked his question there. I wrote the answer in a text editor and was ready to post it, but the question from my friend did not show up until the end of the day. My friend spent several hours looking for an answer to his problem on the Internet, not wanting to ask it on a website. Finally, in the evening he only spent 10 minutes posting a question and immediately received an answer from me. Why is a person willing to spend a day searching for an answer to their question instead of spending 10 minutes writing a question and posting online? Because asking questions in public online communities is incredibly stressful.

## 2. Create opportunities for users to act

In fact, the goal of any early-stage community is not so much to create content per se, but to attract new users who are interested in the community's topics, so that these users stay on the website and create new content on their own. For this to happen, two requirements must be met:

1. There should be opportunities on the website for users to perform some actions, as a result of which users can showcase their skills.
2. Users should feel the need to take action here and now.

In the context of our community, the questions and answers we wrote brought us real users from search engines, but they did not create the essential requirements for user actions. To start engaging users in the community, we began intentionally posting unanswered questions and then engaging the users in the comments to their posts. We were creating other tasks that required user actions and looked urgent from the users' point of view.

### Incentivize one side if your community is a multi-sided platform

Multi-sided platforms are those systems that require the participation of several types of actors to function properly. The main problem with multi-sided platforms is when you want to grow such a system it is difficult to ensure that the optimal balance of interests between multiple user groups is maintained. If you let such a system grow naturally, then there will always be an imbalance of interests and the system will grow very slowly. The most common approach to growing a multi-sided system is to stimulate the interest of one of the sides.

In the context of Q&A websites, if you let a community grow naturally, you will constantly be faced with a situation where there are not enough questions for experts to answer to keep them interested in the community or when there are not enough experts to answer learners' questions fast enough so they want to use the community again. HashCode was a two-sided platform as well. In creating the final value—a question with an answer—two types of actors are involved (askers and answer givers). In the beginning, to keep our community growing, we incentivized those who answered by asking unanswered questions. When the community had experts who regularly visited the site, we moved on to encouraging people to ask questions.

Be careful, the imbalance on multi-sided platforms is achieved not so much due to the difference in the number of users, but in the level of their activity. For example, on HashCode, one answer giver could help several dozen users with questions, but I rarely saw anyone who had more than one question per a day. As a result, to achieve a balance of interests on HashCode, we needed to have several hundred active askers for every answer giver.

*Note: Online communities do not have to be multi-sided platforms. For example, Wikipedia is a one-sided platform because in order to write an article there is no need to wait for someone to suggest writing it. You can just go ahead and create the article you want to have on the site.*

## One way to get people interested in your community is to tell them how you are working on it

We launched a project blog almost immediately after the launch of the community. Before Stack Overflow, programmers had solved their technical problems online either on forums or through email correspondence. The idea of a Q&A website with a knowledge base optimized for reading was a big cultural innovation. When we introduced HashCode, we were constantly faced with a misunderstanding of how the website works and the rules of interaction on the platform. In order to explain it, we began to post short articles on our blog about the basic mechanics of the website and moderation of the community and then we continued with posts about all the main stages of development that we went through.

Stories about the community enable the users to form a group identity. Users begin to better understand the community structure and moderation rules. In addition, projects that constantly talk about the progress they make inspire more trust and strengthen faith in their future success. In a blog, you can write about any technical innovations, project news and the entire inner workings of developing or choosing your software and creating community rules. Conduct surveys and make announcements. Tell the users about other users and interesting content on the site. Basically, you can write about anything that has anything to do with the community.

There is one topic that should be avoided. Don’t write about the industry itself. Stories about the industry will turn your blog into a news portal. Industry news in and of itself isn’t a bad thing and can be very interesting, but it won’t help you build a community. If your goal is to create a community, then any information posted should relate exclusively to the community itself (the platform and the users).

## Use contextual advertising when no one knows about your website

A minute of mathematics: In 2014, when HashCode was already a popular place for finding answers to programming questions, with 40 thousand questions in the knowledge base, the website received about 20 thousand page views per day on average, which resulted in 30 new questions daily. One user viewed about 1.3 pages on average. In other words, 0.5 views per question in the knowledge base and 1 question per 500–600 unique users. Thus, in order to receive one question a day from real users, we needed to have about 1500 questions in the knowledge base, or attract a large number of people to the website in some other way.

After several months of working on community content, we had about 500 questions in our knowledge base. Search engines sent us real traffic, but it was not enough, nobody registered on the site; we only dreamed of having questions and answers posted by some users. To somehow speed up the process, we decided to use contextual advertising to compensate for the lack of search traffic.

We chose contextual advertising because it is most effective at the earliest stage of an online community, while the community is still unknown. Here’s how it works.

- When no one knows about your community, advertising can be shown to the entire target audience. Everyone who gets interested in the ads will click on it. The cost of attracting users is minimal.
- When part of the target audience already knows about the community, in order to receive the same number of users coming to the site through advertising, it is

necessary to show more advertising or do it in a more targeted way. This increases the cost of user acquisition.

- At some point, most people from the target audience already know about the community and only a small portion of those who see the ad will click on it and even they might already have an account on the website. This is the moment when the costs of attracting new users using contextual advertising become unreasonably high.

By the same logic, the cost of acquiring new users becomes more expensive when you increase the amount of paid traffic you want to drive to your website. Advertising can be a very effective tool but it can also quickly deplete your money without a return.

When we just started, no one knew about HashCode and we didn’t need a lot of traffic. Contextual advertising seemed like a good solution. After some experimentation, we managed to set up advertising in such a way that we got registrations on the site for about $1, while every second person registered in the community posted a question or an answer.

Much of our approach was about promoting specific experiences that we thought would motivate people to join the community. For example, we attracted answer givers with slogans like “new interesting programming questions every day” and askers with slogans like “get help with your programming issue for free”. We noticed that specialized words (such as reserved programming language keywords) in the ad title generated more clicks, as did a specific call to action in the ad (such as “ask a question” or “sign up”). Also, when we sent people from the ad directly to the end page where we expected them to act we got more actions taken.

What to invest in advertising or content creation?
Initial content of the site sets the context that explains what future content and in what form you expect to see on the website. Sending people to a completely empty site is not effective; people simply will not understand what is required of them. So, I would not recommend starting an empty website, turning on ads and expecting people to start filling it. Any website should have some minimum initial content. Moreover, if we had the opportunity to buy content cheaply, I think we would have invested in that rather than in advertising. The truth is that writing more or less good content is not easy. As a result, in real life we need to invest in both.

Writing content and getting it into the index of search engines takes time. In contrast, advertising creates traffic here and now. As a result, advertising can serve as a temporary solution until your site is receiving enough search traffic. You should not rely on advertising in the long term, since the cost of attracting a user will increase with the popularity of your community. The sooner you can stop using advertising, the better it is for the community and your finances.

## Personally invite people to your community

Although we set up the ad to work effectively within the budget we had, it only generated something like 3–5 questions per day. When we tightened up our advertising campaigns to increase paid traffic, the price of a posted question immediately soared. We started looking for other ways to attract people to the website.

At this point, I told my friends that I ran the community quite often and now we had advertising experience. We decided to combine these two approaches and try to invite potential users in person. The approach was to search for those who actively posted something about programming in thematic groups on social networks. The reasoning was that on social networks all discussions are public so it is easy to identify proactive people, there is a private messaging feature, and, in addition, social networks are not designed to solve technical problems. We expected a lot of people to come to our site if we invited them.

We decided to approach the task the same way we approached working with contextual advertising. We created two personas. The first persona was a question asker. The second persona was an answer giver answering. Then we drafted a template message for each person. Next, we went through all the threads in thematic groups on social media that we were able to find and if we saw active users who were posting something about programming, we contacted them. For each user we customized the template, always greeting the person by name and writing why we decided to write to them. It took about an hour to search and send twenty messages, but the result was amazing. At least half of everyone to whom we sent the message registered on our website. The nuance was that these were mostly answer givers who helped others. There were very few people who often asked questions, and at the very moment we contacted them they had no questions to ask on our website.

*Note: Stack Overflow is an applied knowledge base which means that it is almost impossible to ask a question simply "for the love of knowledge." For a person to ask a question on Stack Overflow, they need (1) to work on a real project right here and now, (2) to face a problem that they cannot solve and cannot find an answer on their own, (3) to overcome all psychological barriers and ask the question in the community.*

From time to time I received replies from people I wrote to. The answers were completely different. Starting from constructive criticism and questions about the project, through to obscene language in protest against sending private messages with an offer to join the community. If you ever decide to invite people via private messages to your community, be prepared to encounter everything from admiration to outright hatred.

## Place posters where your target audience gathers

When you’re studying at university, most of what you do is new to you. In addition, many people begin to take their first steps towards their careers while studying. Students always have an abundance of questions, at least in theory. Plus, any interesting information spreads instantly in the youth environment, where everyone is closely connected and in constant communication with each other. Based on those thoughts we assumed that if someone started successfully using our community to solve academic programming questions, it could quickly become a trend and we would greatly increase the level of activity on the website. We decided that we needed to be present in universities somehow.

*Note: Our second guess was that beginner programmers have not yet decided on their preferred place to get answers online. If they become part of our community today and have a good experience on the site, then in a couple of years we will have a lot of experienced developers on the website. As I found out later, in fact, things are somewhat different. Communities at different stages of development are interesting to different types of users. The type of users is determined by the current state of a community and the challenges the users face, and not by the professional qualifications of the users. When a community moves from one stage to another, the most active users at the next stage of development slowly replace the activists at the stage the community is passing.*

Working with universities looked promising, but there was one caveat. Placing an advertisement at a university was not an easy task. The purpose of universities is to teach, not to monetize their students. Usually at universities there is no place for paid advertising at all, there are only bulletin boards on which teachers post useful information for students. A bulletin board is usually a real board, on top of which there are glass doors with a lock, the key to which is kept by the dean who controls everything that is posted on the boards. Although for the most part the bulletin boards I saw contained information about the educational process, from time to time there were also announcements for internships at companies for students. It seemed that there was a chance and we decided to try.

I chose one of the faculties at my university as a test, found the contact details of the dean’s office and wrote an email containing the essence of the proposal and a request to discuss everything in person. Unexpectedly, they answered and made an appointment for a meeting. On the appointed day, I went to the dean’s office. At first everything was quite hectic, but at some point the dean put everything aside and began asking about our community, basic mechanics and how he could help us. Everything went so well that we were allowed to not only place posters for free on the bulletin boards, but also they offered to place the logo of our community on the main page of the faculty website with a hyperlink! After such success, we decided to scale the approach to other technical universities in the city. I found the contact details of the faculties where they taught programmers, wrote to them, went to personal meetings with them and, with their permission, placed our posters on all bulletin boards that I could find.

A few words about our posters. Contextual advertising on the Internet is a small text block consisting of a title, a short description and a call to action. Banner advertising offline is a completely different story. If you limit a poster to just text or a slogan, the poster will be forgotten as soon as a viewer takes their eyes off it. Offline banner advertising should be remembered and evoke associations not only when a person looks at the poster, but also hours later (a lot of time will pass between the moment a person sees a poster and the moment when we want them to recall the information from the poster). Over many years of evolution, human visual memory has become geared toward remembering faces. As a result, if you want your poster to be remembered, add a human face to it. For example, for our poster, which we placed at universities, we bought a photograph of a red-haired (in the color of our logo) smiling woman in business attire giving a “thumbs up” to our site from a stock image platform. Then we added a title, a call to action, and a few frequently used keywords from the programming world to the poster.

We placed posters not only at universities, but also in the offices of large software companies. The approach at companies was a little different. Instead of looking for some kind of a decision maker, I wrote to my friends who worked at those companies, met with them and asked them to place a poster on the bulletin board or in the company’s cafeteria, with the permission of the management, of course (Note that I have never heard of management at any company that was against it).

As time had passed, we had reason to believe that several very active users came precisely from placing the posters at universities. That fact made this approach a good investment of our time. I tried to negotiate the placement of posters whenever it was appropriate. For example, one of the corporate events that the company where my girlfriend worked at that time was held in Riga. The employees were allowed to take their significant other with them and I went on the trip with her. After arriving in Riga, the first thing I did was find out where the Riga Technical University was located, printed out our posters and went to the programming faculty.

## Speak about your community at conferences and meetups

When I contacted universities about placing posters, many deans greeted us with great enthusiasm. We were asked many questions about our community and were told about the actual problems of students. At one university we agreed to try to hold an offline event, where students would have the opportunity to ask any programming question they were interested in in real time. Organizing the event seemed quite simple. All that needed to be done on our part was to gather experts who could answer questions. I knew several programming languages well and was aware of the main trends. To cover the rest of the technologies, I invited four of my erudite friends.

For such an event to be successful, it is important to have two things. The first thing is an introductory warm-up speech, a presentation or a story that will make attendees relax and engage in the process, something simple and entertaining. Secondly, one needs to have swag that they give away to active participants. Swag would add a gamification component and stimulate the people to be active.

I prepared swag and an introductory presentation. On the day of the event, I picked up my friends and we went to the university together. It turned out that the event would take place in the main assembly hall with a capacity of about a thousand people. Our team took the stage, and the main page of

HashCode was displayed on a ten-meter-high screen behind us. The event had begun.

I warmed up the audience with the introductory presentation, brought swag onto the stage and invited the students to ask their questions. Oddly enough, the audience did not have any questions about programming itself (remember, you can ask applied questions only if you have a specific problem at hand). Fortunately, we had a “Plan B”, according to which, each of the experts on the panel told several interesting technical stories from their field, and then we moved on to discuss how one can successfully look for their first job in the industry.

Honestly, the event did not go as we expected. After weighing the pros and cons, we decided to abandon this type of partnership with universities. It seemed to us that the format when the audience needs to ask questions is not effective. In parallel, we had worked through several other variations of offline formats. Our conclusion was that being a speaker at an event is worth the effort only if we present the story of our community, its structure and the problem solved by it to the domain experts. The most critical thing in this case is to find a place where you can access those experts. For example, when we launched a community of mathematicians, I contacted one of the best mathematical educational institutions in the city and agreed to present them with the idea of a community.

## Hold joint contests

Seeing the success of working with universities, we decided to try to expand the community with online partnership programs.

The essence of partnerships is that you need to find a partner such that you both get more from the partnership than you invest. To achieve this, you need to look for non-competing projects with a similar target audience, which most likely is unaware of your community, and negotiate with these projects to direct their audience to your website in exchange for something else (for example, you will send your audience to their website or pay for swag for the audience of their project). It is easier to find partners among projects that are trying to grow or very large companies. The reason is that projects that are trying to grow usually use all opportunities they have, and large companies have enough resources and altruism to invest in the development of the industry itself.

Let’s see how this works using the example of a contest that we held with the “Developer Jokes” project (not the real name). Developer Jokes was a group on social networks in which the authors posted various jokes somehow related to programming. (I was subscribed to them; their jokes were funny!) The point of the contest was that for several weeks, every two days we posted a puzzle question on our website under a special user account. The administration of Developer Jokes, at the same time as we did, posted a link to the question on our website in their group on social networks. The winners, people who answered the questions correctly first, received from us a t-shirt with the joint logo of HashCode and Developer Jokes projects. We paid all the expenses for the swag and for sending them.

Partnerships can be completely arbitrary. For example, at some point information partnerships had gained momentum. Most conferences were using this type of partnership. A conference partner needed to announce the event to their audience and in return it received something from the conference organizers. Usually, conference organizers offered to place the logo of a partner with a link to its website on one of the pages of the conference website. We actively used information partnerships to increase the number of backlinks to our website.

Information partnerships did not bring us direct traffic, but in the eyes of search engines our domain’s trust greatly increased, which gave us higher positions in search results and more traffic to existing content.

## In conclusion

Making people aware of your community is one of the most difficult of the launch phase tasks. In addition to everything described above, we tried many other approaches to promote our community, from distributing flyers at the entrances to the business centers of large IT companies to posting answers to programming questions on third-party sites with a back link to our website. No one knows what will work for you; there is no universal solution. There is only one universal truth. No one will ever know about your wonderful community if you don’t talk about it to everyone, everywhere and all the time.

# 2. Onboarding and engaging early users

In total, my friends and I needed to ask a little more than two thousand questions before the questions on the site began to be asked by the users themselves. It’s hard to imagine how exhausting it had been for us to keep creating content for that many months. It was a real ultramarathon for our creativity. We saw every new user in our community who asked or answered questions as a treasure. We did everything possible to ensure that the users got the best possible experience from our community and returned to the website again. Let’s look at the activities that helped us the most in onboarding and engaging early users.

## Provide feedback to users as quickly as possible

One of the factors that critically influence the experience of a user is timely feedback on the user’s actions. There are three types of feedback:

1. *Positive feedback* is when someone reacts positively to a given user’s actions.
2. *Negative feedback* is when someone expresses disapproval of a given user’s actions.
3. *Absence of feedback* is a situation when a user does not receive any reaction from the community to their actions.

Positive and negative feedback are the basis for user engagement since they carry information about a user’s actions. The absence of any reaction to a user’s action or a reaction that is difficult to interpret is something that should be avoided at all costs.

Our standards for feedback to the users that we strived for were an answer within an hour and an edit, comment, and upvote within thirty minutes.

*Note: Since at the very beginning of the community we tried to answer all the questions that appeared from real users on the site ourselves, other answer givers had little opportunity to show off their skills. To compensate for this, we asked questions until the real answer givers began to answer all the questions faster than we did. At this point, we stopped writing content on the website completely.*

## Define and optimize user funnels from landing on the site to creating content

Growing online communities largely comes down to engaging users to create content in the community. The more effectively you convert passive readers into active users who contribute content to the site, the faster your community will grow.

*Note: From a marketing perspective, growing anything comes down to creating effective funnels. The idea of a funnel is to send people to your target page and expect some of them to take the action you want. The more people act, the more effective the funnel is.*

The peculiarity of HashCode was that we had two types of users: those who answer questions and those who ask questions. Each type of user approached posting content differently. The vast majority of those who ask questions came to the site from search engines to a page with a specific question. If none of the answers to that specific question answered the person’s question they came to the site with, the person had a choice: continue searching or ask a question on the site. Thus, for many people, the question page was the main entry point to the community. Therefore, we optimized the question page to contain multiple calls to action for users to start asking questions on our site.

For those answering questions, our user persona of the answer giver looked a little different. These are people who are passionate about programming, lack interesting technical tasks at work and have some spare time to spend on forums. Besides landing on a specific question page through search, we saw two other main funnels for such users: direct visits and visiting the site through subscription to content.

In the case of direct visits, the best way to engage is to provide the users with the ability to effectively find content that interests them. Typically, a full-text search on a site is not a suitable solution for this. Filtering is needed to allow users to create a selection of posts based on some categorical characteristics. On HashCode, we provided users the opportunity to select content based on question types (unanswered, new, last week, etc.) and tags. With the move to the Stack Overflow platform, users got an even more flexible way to select questions based on their attributes (closed or open, time posted, number of answers, etc.) Our rule of thumb was that the more flexible the way one can select content, the more likely they are to find content that they want and will act on it. When a user was creating a selection they wanted, we always gave them the option to save the selection and subscribe to receive updates on the selection via email or RSS. These subscriptions formed the second type of funnel for engaging users.

Not all people who come to your site have something to contribute to the community at the particular moment they come. At the same time, your goal is the same: To engage users by making them start creating content in your community. One popular approach is to welcome each new user and ask them to write something about themselves in a special place for non-thematic activities. This allows users to be able to participate and have a complete and positive experience in your community.

*Note: Overall, we can say that a user has an opportunity to act when (1) they can find content that interests them, (2) the content assumes that some action is needed and (3) they can show off their skills by acting.*

## Strive for high quality content and respectful behavior of users toward each other

Let’s briefly talk about the quality of content and interpersonal relationships, which go hand in hand on the Internet. At the time we launched HashCode, users of the vast majority of online forums did not see any issue in writing posts illiterately, ornately, *etc.* Low-quality posts were very common. The second characteristic of the Internet forums at that time was that some people reacted harshly and negatively to these same low-quality posts, as well as to many other violations. One thing led to another and then both led to constant drama, causing many people to fear the mere thought of participating in online forums.

We looked at the situation and we got the idea that much of the rudeness on online forums happened due to the difference in the view of the reality of different types of users. For example, “oldies” who have been using a forum for a long time sincerely did not understand how one could post anything on the forum without reading the community rules, or asking a question that was almost identical to many existing questions. At the same time, those posting on the forum for the first time did not see anything wrong with not knowing some of the nuances of the rules of a random forum on the Internet, and in general, “they have always communicated this way with friends and colleagues and no one ever got angry at them for it”.

*Note: I have asked questions on online forums myself many times and I know that asking a question is much more difficult than answering it. First of all, psychologically. Imagine that you are in a classroom at school. Posting a question is similar to when a teacher calls on you to stand in front of the class and asks you something you don’t know. I forgot to mention, this is your first day at the new school! On the other hand, posting an answer feels similar to the situation when you raise your hand to be asked about something you certainly know. Those are radically different psychological experiences.*

This may sound counterintuitive, but each side, within its own reality, was right. We saw our task as bringing two realities into one. To awaken empathy and compassion in experienced users towards people with less experience on the site or within the topic, while at the same time maintaining the quality of content in the community at the highest level. To accomplish this, we performed the following activities.

## Moderate all posts from new users

First of all, we improved all posts that, in our view, could be improved in any meaningful way. Grammar, punctuation, clarity of wording, we paid attention to even the smallest details. Our goal was to create an “island of knowledge” in the “noisy ocean of the Internet.”

Editing posts is one of the best ways to help new users adapt to your community’s standards. When you improve someone’s post, (1) the author of the post clearly sees what is required of them in terms of writing, (2) the author of the post also sees that there are people in your community who are ready to help others by taking real actions instead of just criticizing them, and (3) the other users of your community learn how to behave (editing instead of criticizing) when they see shortcomings in someone else's posts.

On HashCode, we moderated the first few posts of each new user. Here is why. At the moment a new user posts a message on the site for the first time, they have not yet formed friendly relations with other users, they do not have any positive experience participating in the community. In other words, there is nothing holding a new user to the site. As a result, new users most likely have no motivation to do anything more than the absolute minimum. Any negative experience can lead to a person leaving and never returning to the site. At the same time, new users do not yet know the rules of your community and as a result, the first posts of new users are the venues where the most rule violations from newcomers and rudeness in response to them from the regular users occur.

It's best to moderate the first posts personally until you find volunteers who have a high level of empathy, the time and desire to help newcomers adapt to your community and understand the mechanics, tools and rules of the site. Those are very special people, and they are not easy to find.

## Use canned comments

Imagine that a person is taking a bus ride and their foot is stepped on. How will they react? Most likely, they will smile and say that no harm was done. Now imagine someone's reaction if their foot is stepped on every day, many times during the trip. What will their reaction be when someone steps on their foot again? I'm guessing we will not see a smile on their face.

Most of the rudeness on HashCode (as in any other online community) came down to a few situations when new users made typical mistakes. These mistakes at some point began to cause an exaggerated negative reaction among regular users, solely due to their regularity and typicality. Moderation of such negativity is mandatory, but only helps to a certain extent. If you simply introduce a rule that prohibits the negative reaction, the open negativity will turn into veiled passive aggression. The reason for this is that it is natural to react negatively to the violations of a community's cultural norms or a misuse of the community.

At first, each user reacts to violations of the community norms in their own way, but over time, naturally, the reactions of all regular users get consolidated into one reaction, which the majority of users adapt to. This adapted reaction will not always be the best or can even diminish respect for community members. Therefore, you either need to offer your own version of the reaction to typical violations that users can use or come to terms with the version that the users will come to naturally.

On HashCode the most typical mistakes occurred in questions and caused rudeness in the comments to those questions. To minimize this rudeness and establish the desired reaction to those typical mistakes, we asked users to use special canned comments, which covered the most common situations.

*Note: Canned comments on HashCode were available to users directly via web interface. Just below the comment input field there was an option to select a specific canned comment from the list. After selecting the template, the site engine added text to the input field, but did not send the comment to the server, and the user had the opportunity to modify the comment if necessary.*

A few words about the canned comments themselves. Canned comments should be written from the perspective of the reader, that is, the person to whom they are addressed. Comments should not contain negativity, judgment or criticism. They should be short and clear. Creating a list of canned comments is a great task to work on with the regular users themselves. Once you have a vision for the first few comments, start a public meta-discussion. In the discussion, suggest your own templates, explain in which situations which template to use, and allow users to suggest their own options and improve yours. The ability to discuss templates and offer suggestions will encourage adoption by regular users who usually moderate the community.

## Storytelling

HashCode was a community of volunteers who used the site because they saw some value in it. It was impossible to make volunteers do anything without personal desire. They either wanted to help us with something on the site, by doing something a certain way, or nothing happened. Creating canned comments and software for using them was only the first step. Next, we needed to convince users to use the proposed approach. Even now I know of no more effective method of motivating people to adopt social and software innovations than through storytelling and we used it a lot.

When writing stories, it is very important to know your audience and write stories that create empathy in your readers. The core of the HashCode community was mostly experts who answered other users' questions. We were writing the stories mostly for them. We tried to show the psychological experience that a new user has when asking a question on the site for the first time and how this experience affects the quality of those questions. We talked about the tools users can use to convey to the authors of low-quality questions what they wanted to convey. At the same time, we explained how the editing mechanism works and that editing others’ posts is not only okay but also the only sure way to create a community in which everyone writes competently and clearly.

*Note: We wrote stories not only about the mechanics of canned comments, editing and also the importance of treating each other with respect, regardless of the circumstances. We tried to document every cultural aspect and every engine mechanic that confused the users.*

## What it takes to launch a new community

A year after launching the software developer community, we followed the Stack Overflow example and started scaling our Knowledge Network. First, we divided the software developer community into three: a community of programmers, a community of system administrators and a community of advanced PC users, then we launched a community of mathematicians and immediately after that a community of Russian language linguists. Subsequently, we launched six more communities on different topics.

My main conclusion is that any community requires a lot of human resources at the startup stage. At some point we introduced a rule for ourselves that we would launch a new community only if (1) we have a person who can create an initial content of at least 500 questions and answers, (2) this person is ready to answer new users’ questions and (3) moderate the community for at least six months after the launch. If at least one of the conditions is not met, the community is most likely doomed.

## Why some communities got off and others did not

In total, we launched 11 communities about completely different topics. Many of them differed strikingly in the pace of growth.

The community of mathematicians got off almost immediately on autopilot and one month after the launch, real people were asking and answering questions on the site. It was something incredible, especially in the context of the community for software developers, which we worked on for more than six months in order to get minimal activity on the site. The linguist community has evolved at about the same pace as the community of programmers. At the same time, in the community about fashion no one asked questions even after a considerable time. A reasonable question arose: Why did some communities immediately become active, while others did not?

We noted the following criteria that influenced the future success of a Q&A community on a given topic:

- Potential users have access to a computer and the Internet at the time they encounter a problem.
- Availability of time restrictions on receiving an answer to their questions.
- Lack of people to answer a question offline.

In other words, a good topic for a Q&A community is a topic that implies that a person interested in it works with a computer on a daily basis, needs an answer here and now, and there is no one to help them offline.

Let’s take a look at the mathematician community. Usually, a person who is doing something related to mathematics needs to solve a problem during a short period while being at home alone (think of students). In the context of programming, the situation will be similar except that the time limits are not as strict and software developers often work in teams, so they can ask their colleagues offline. Therefore, we can expect that all other things being equal, a community of programmers will not develop as dynamically as a community of mathematicians. This is exactly what we saw.

# 3. Development of the ecosystem and monetization of the community

By the time the community was about a year and a half old, all content on the site was created by the users themselves. Our organizational tasks were limited to content moderation and some growth activities for the community. At this point, we were still working on the project in the evenings and on weekends. The costs of running the project were minimal. Just some money, used wisely for the server and advertising only for new communities.

While it was possible to moderate and grow the community without compromising the main work and personal life, it was not possible to develop any larger features. We simply didn’t have enough time. Thus, over the last year of working on the community, we have accumulated a list of features we thought the site should have to grow faster. In addition, we began to think about what the next step for the project should look like: How to achieve profits that would cover the operational costs and could reimburse ourselves for our investment to date and look forward to a return on that investment in the future

I would quit my job and focus solely on the community for six months was one option. In this case, we could add new features to the site and monetize the community. Leaving a stable job for nowhere was scary, since I had no serious savings. Besides, I was not quite finished with my university degree. Writing a thesis project and simultaneously looking for a new job would not be easy. On the other hand, the project I was working on at that time at my 9-to-5 job was not interesting to me, and my professional growth within the company had long since come to naught. The new features that we planned for HashCode looked very interesting from the technical point of view and promising in terms of business success.

It was a difficult choice. Working in a large company provided stability, and working on our community attracted me to the opportunity to try something new and hopefully create a useful product. After weighing all the pros and cons, I wrote a resignation letter and started learning a new programming language, Go, in which we decided to develop our new infrastructure.

## Plans for the following six months

When I left my job, we had specific development goals: improve the site engine, create a site that would connect all the communities into a network and develop a job listing site.

### Connecting communities to a network

Since the launch of HashCode, we have been looking at Stack Overflow’s approach to scale. We looked closely at everything that worked well for them and thought about how it could be reused in our language niche.

In May 2009, Stack Overflow, the community, was divided into three communities: the community of programmers (Stack Overflow), system administrators (Server Fault) and advanced users (Super User). In August 2010, folks launched the site stackexchange.com, which united all the sites in a network. Essentially, it was a site that provided a way to overview all communities. It had a list of all communities, the most interesting questions from across the network in real time, leaderboards for each community, *etc.*

We decided to go in the same direction. A year after the launch, we divided the community of programmers into three, and then began to launch new communities on other topics (mathematics, physics, Russian language, etc.). We needed a website that would unite all communities into a single network and enable users to review all communities easily. We did not expect this site to significantly influence the growth of any one particular community, but it should make the project holistic. Plus it would make it easier for us to monitor our communities and developing that site was a quite straightforward task.

## Job listing site

Developing a job listing website was a much more ambitious task than creating the network site. Developing your own website from scratch, like any other large system, is very difficult. The difficulty is not in writing the program code itself, but in thinking about what the end users need, how to make the site convenient and not too complicated.

An entire department has been working on Stack Overflow Careers for years (Stack Overflow Careers was a job listing board that Stack Overflow had at that time). We saw the only accurate approach was to reverse engineer the site and program its minimalistic replica.

## Improving the Q&A engine

We used an open-source engine to host our communities. When we launched, the engine was pretty much the same as Stack Overflow was at the time. Since then, Stack Overflow has been actively developing their Q&A site but the open-source project we used has not. After a year and a half, the engine we were using began to lag behind in its features. On top of that, since we had several communities in our network, we needed to make the communities know about each other. *i.e.* make it so that the users had a single profile, shared reputation, *etc.* on all the sites.

## Monetizing the community through the job listing site

I left my job in October 2011 and spent the next 10 months writing code without a break. Initially, our plan was to focus on finalizing the technical infrastructure of the project and once everything was ready, I would find a new full-time job. I completed the network site and updated the Q&A engine quite quickly, but the work on the job listing site took more time than we initially expected. Instead of completing it in April, we did not launch HashCode Careers until June.

We never raised funds, building the entire project with our own money. As a result, along with the development time, our own spending also increased, which we now needed a return on. It seemed to us that either we would try to monetize now, or we simply would lose our investment. I decided to put off looking for a job and focus on monetizing the project by selling job listings.

### Why a job listing site?

Stack Overflow was never the first or the only Q&A site for programmers. There have always been other communities. All the alternatives at that time had one thing in common. They had difficulty generating income. Some sites even switched to a business model where visitors pay for accessing site content, which many community users considered unethical. The Stack Overflow founders' idea was to make a site that is completely free for all programmers and monetize it through third-party services. Joel Spolsky (one of the founders of Stack Overflow) told the story many times about how, as an experiment, even before the launch of Stack Overflow, he tried to monetize his blog through a banner with job listings, and that made him $90K in revenue during the first month of the experiment. So, a job listing site was not an accident. In addition, one project complements the other. Stack Overflow Careers was adding a special flavor to the community, and, on the other hand, all job listings were shown to the core target audience. Thus, Stack Overflow Careers has been an indispensable part of the Stack Overflow infrastructure. We were trying to replicate the success of Stack Overflow in our language niche, and a job listing site seemed like a good next step for us too.

In addition to that, around the same time we launched HashCode, another project was launched in our language niche, a job listing site specializing exclusively in IT related jobs (programmers, testers, data analysts, etc.) When that project was one year old, its founders raised $100K in seed money. The project was well known and looked successful.

There were reasons to believe that the market existed and that there was an opportunity for small companies like us to carve out a niche. We saw no reason why we shouldn't try.

## How HashCode Careers turned out

By the end of 2012, six months after the launch of HashCode Careers, 2,000 people had registered on the site, 800 of them created their CVs, and 77 job listings were posted. The site was absolutely free for job seekers; employers paid for everything. The cost of posting a job listing was $15, and the cost of accessing the candidate database was $30 per month.

At that time the cost of getting one CV in the database was about $30 for job listing sites. For us, this cost was reduced to zero, since all we did was advertise the Careers site in our own community. On the other hand, we definitely lacked the scale of HashCode itself and the 800 CVs in the database were of little interest to most potential customers. For the CV database to be attractive to employers, it had to contain about 10 times more candidates, and selling job listings turned out to be very expensive.

## The cost of selling job listings

Our approach to selling job listings was extremely straightforward. We looked at who was posting job listings on other similar sites and called these companies with an offer to post their job listings on our site too. Recruitment managers were very active in contacting us to ask about our job listing site, the community, the prices, and many other things. Usually, a phone call with them took at least half an hour of my time. Not everyone was ready to work with a project in the early stages of development so not all of the clients with whom I spoke purchased job listings on our site. Those who did usually started with a trial job listing to see how it would go. As a result, it turned out that to sell a $15 job listing, we needed to spend several hours of our time consulting with potential clients.

At a certain point, it became clear that things were pretty much the same for everyone in the industry. Here is a small example of how recruitment sites worked. We needed to hire a designer. It was a huge job, so I decided to try using a regular job listing site rather than a freelance site. I registered on the site, entered the required data about our company and wrote a job listing. I got to the very end of the process, all that was left was to pay, but I didn’t have my corporate credit card at hand. It was sad but there was nothing I could do. I closed that job site and posted a free ad on my favorite site for finding freelancers and forgot about the other job site for a while. For the following month, every two or three days, the job site manager called me and asked me to complete the posting. It was a real person. The manager wanted to understand why I didn’t post my listing right away, and offered to consult with me on how their site works and talked about the advantages of working with them. “Is this all for $15, really?” was going through my head and “How much staff do you need to have to compete with these guys?”

I talked to several friends who were involved in similar businesses. In their companies the situation looked similar. It was not clear to me how we could make a profit in this economy.

## An attempt to connect universities and companies

I still had contacts with the administration of various universities. So, we decided to try to enter the market from the other side.

Universities are a source of junior specialists for many companies. On the other hand, people go to universities to study something so that they can find an interesting, well-paid job afterwards. The typical student employment model then was that undergraduate students did internships at companies where they essentially worked part-time. Then, they wrote a thesis on the topic of the project they were working on and then continued to work in the company after graduating from their university.

Students needed jobs, the ultimate goal of universities was to employ students, and companies needed educated employees. This created a unique situation in the sense that if everything goes well, absolutely all parties would get what they need. The nuance was that it was not trivial to complete a “transaction”, that is, to make an agreement between the company and the university to cooperate and somehow tell students about the available opportunities. There was no automation. When a company wanted to employ students from universities, its representatives had to look for contacts from each department at each university and go to negotiate personally which is a ton of work.

Our idea was to improve our job listing site to provide universities and companies with a special module through which companies could post job listings for specific universities, and university representatives could refer students to internships and track their progress in the company. On our part, we would significantly and quickly increase the base of candidates and companies that worked with us.

I contacted two universities and three companies. Everyone expressed real interest, so we put everything aside and finalized the new module on our job site in a month, and when ready, we resumed negotiations on a test project launch. Then the unexpected happened. We and our partner employers were ready to start right away, but universities began to postpone the launch of the pilot. It looked like someone at some high level could not decide when to launch even though the deans saw high value in working with us. When, after a month, we still had not received any response from our partner universities, we decided to abandon the idea altogether. It was time to move on.

## The split of the team

Meanwhile, six months had passed since the launch of HashCode Careers and the reality began to dawn on us that, most likely, with the available resources and traffic on the main site, we would not be able to ensure sales of job listings at a level for the company to break even. When I left my day job at Motorola, I had a small amount of savings, which I planned to live on for the next six months. Due to the protracted development, I had to sell the only thing I had at that time, my car, and based on the results of the six months that we spent on monetization, even the money from the sale of the car evaporated. On top of that, I'd been working 70 hours a week for the last year. My vital energy and my tolerance for failure also started to reach the zero mark.

The other problem was that we began to disagree within the team on the vision for the future of the project. Part of the team insisted on completely stopping work on the community and investing everything that was left in developing the Careers business. Their argument was that HashCode Careers is a real business that can generate income. We had a high-quality engine and first rate clients, as well as an example of a similar startup that raised investments. I had a completely opposite opinion. The Careers site looked really interesting, but I was convinced that without a large community of programmers, a job listing site would not have a chance in the current reality; selling job listings was extremely expensive and required a large number of employees.

*Note: Interestingly, I encountered a similar situation later, when I worked for Stack Overflow. Many executives and product managers at various levels have long prioritized products that directly generate revenue over community efforts. In my opinion, this approach is justified only in one case: When the community is growing and does not require many resources to keep growing. In my view, the logic here is simple: if the community loses its popularity, all other company businesses will almost immediately become ineffective or even unprofitable, since the cost of selling the company's products and the complexity of selling the products will increase by an order of magnitude.*

*Note: In the summer of 2012, for reasons unknown to us, search traffic on HashCode began to grow exponentially. After a month of growth, we were contacted by several venture capital funds who were willing to invest in the project if it continued to grow. Unfortunately, growth soon slowed down and we did not receive any* *investment. Note to self, I concluded that for any startup, the essential activity is growth. There will be growth, just like everything else.*

We couldn’t come to an agreement and the team broke up. I bought the rest of the project and was left to work on the community alone. Around the same time, Daria, the lady of my heart, made it clear to me that it was time for me to stop playing at being a startup guy and start thinking about the future. We had already graduated from university and it was time to start a family, so I needed some kind of financial stability. The issue of monetization had become incredibly critical.

## Monetizing community through advertising

Any healthy project, including a community project, must be self-sufficient, otherwise it will sooner or later close down. At a minimum, a self-sufficient project should cover all operating costs (cost of servers, minimum staff, costs of running a legal entity, etc.). There is another important reason to strive for commercial success. I believe that commercial success of a project is an indicator that what we do is not only important to us, but also is useful to other people.

I was faced with a choice: either bring the project’s economics to break even or close the project completely. I didn’t want to shut the community down, so I created a simple plan. Over the following month, I wanted to close all obligations to clients on the Careers site and figure out another way to effectively monetize the community. Then, no matter what happened, look for a job.

I see two primary types of monetization in online communities. You either monetize the users (that is, you sell something to them), or you monetize the result of user interactions (that is, somehow sell the content created by the users). Basically, with the job listings site we were selling to users, so it was time to try to monetize the content of the main site.

## Banner advertising from advertising networks

On our main site we had an advertising unit in the sidebar with a static banner from the Yandex Direct banner network (the largest banner network in Ru-net at that time), but we never seriously worked with advertising, since our focus was on the job listing site. At the very beginning, we added these banners to the site as an experiment, to see how much money they could generate. On average, we got about $0.3 per thousand impressions from Yandex. It was so little that we postponed advertising until better times.

When I revisited our banner ads again, the earnings were very similar. I felt like I needed to do something more effective than just placing a banner from one of the ad networks to increase the revenue numbers.

*Note: My general conclusion about banner networks is that only the owners of banner networks make money on banner networks. If you have a website that you don’t want to clutter up with a lot of animated blocks that fill the entire screen, it won’t be easy to make any significant money with banner networks. If you want to make money from advertising, the only way to do it is to sell advertising yourself.*

## Pay-per-impression advertising

To understand how and for how much other sites sell advertising, I went through the main websites at that time, which were somehow related to software development. I found out that many popular sites had a media kit with prices for advertising in public access. I also took a look at what companies bought advertising on these sites and in what format.

Logically I divided all advertisers into two large groups. The first group was small companies developing their only product. The second group was large established companies that dominated in their markets, such as Microsoft, Intel, IBM, *etc.* I also saw two big types of advertising. The first type was regular banner advertising. The second type was various special programs. Although most of the special programs were very diverse, the idea behind all of them was the same. It consisted of a special landing page developed for the partner on the site domain and some special element in the site UI (for example, an additional button in the main navigation section or a modified site logo) upon clicking on which a person was taken to the landing page. Banner advertising was mainly purchased by large companies, and special programs were purchased by small companies. Special programs were purchased once, while banner advertising was purchased by the same companies every two to three months.

Banner advertising looked like a tasty morsel. It should have been very easy to work with. All that was required was to add a graphic banner to the site for a certain amount of impressions. At the same time, the average check for the banner advertisement was high. In our language niche at that time, a thousand impressions were traded for $10. To start working with it I needed to find the decision makers among advertisers and invite them to use my site.

I created a list of all the big companies like Microsoft, Intel and IBM which bought advertising on other sites and started looking for contact information of their marketing departments' representatives. Finding contacts was not difficult, it was difficult to get them to reply. No matter how many times and to whom I wrote, no one answered me. One day, I was ready to give up, I had nothing better to do and clicked on a Microsoft banner ad on one of the sites. The site redirected me to a landing page not to the same domain, nor to the Microsoft domain, as usually happened, but to the domain of a large integrator company. I knew that company was selling software to universities. What I didn’t understand was why they used advertising banners on the Internet, because selling software to universities was definitely done in a completely different way.

As it turned out, this integrator handled almost all sales for all large companies. Not only to universities, but everywhere. It was the integrator who was responsible for advertising placement, and not vendors like Microsoft. From the outside, I saw it like this: When the headquarters of a software company somewhere around the world decided that an advertising campaign for a product was needed, they contacted the integrator, agreed on the details like budget and gave the go-ahead to launch the campaign. The integrator’s job was to maximize the use of the dedicated money, ensuring that all the requisite metrics are met. I should have contacted the integrator to sell the advertising, which is what I did after understanding the issue.

## Pay-per-action advertising

In addition to the companies that used banner advertising, I also wrote to everyone who launched special programs on other sites. As it turned out, they were not ready to pay per impression for banner advertising but were happy to cooperate with us if we agreed on the payment for an action. For example, for a lead we send on their site, a user filling out a certain form, downloading their product, *etc.*

As I found out after working this way, pay-per-action advertising is not as profitable as paying for impressions. At the same time, I had full control over the advertising on my site and could set it up quite effectively. I was able to earn about $2 per thousand impressions. Another advantage of this approach is that it is much easier to find companies that are willing to pay per action than companies that are willing to pay for impressions and they do not usually limit their budgets while you are sending them people who are interested in their products.

## Summing up

A couple of months after starting to sell advertising, I had a diversified sales model. For four to six months a year, the site had banners from large companies, which I placed on a pay-per-impression basis. They were usually tied to major product releases. Most of this kind of advertising was placed by the same integrator. In addition, I worked with several companies that offered software products for a specific target audience. For example, an IDE for development in a specific programming language, or a paid widget library. This type of advertising ran constantly, as customers paid for specific actions on their sites like downloading a product. I filled all unsold impressions with advertising from banner networks like Google AdWords and Yandex Direct. On each page of the site I had two or three static banners, which in total brought in an average of $8 per thousand impressions and the community reached financial self-sufficiency.

The average check for an advertising campaign in our small community was about $1,500. Let me remind you that on the job listing site it was $15, since people usually posted only one trial job listing. The cost of selling and servicing the deal was minimal. I just had to find companies that were interested in the type of users we had in the community, negotiate a partnership with them, and upon completion of the advertising campaign, provide a report and an invoice. As a result, to serve one customer I needed no more than a couple of hours of time. In addition, the advertising on our site had good conversion rates and almost all companies became repeat customers.

*Note: Working with large companies has its own specifics, which consist of annual or semi-annual planning. Many companies set their marketing budget at the beginning of the year, and if the company has already set a budget and discussed everything with their integrator, and the integrator has planned advertising on specific sites, no matter what you do and no matter what results you show, they will not work with you, since you are not included in the plan. The only thing you can do is agree on a partnership for the next year or in six months. In other words, selling advertising to large companies has a time lag. Additionally, integrators may be reluctant to work with small sites because they simply may not have the human resources to service these partnerships on their end.*

# 4. Leaving HashCode and then selling it

As soon as I managed to figure out how advertising works and achieve the first sales, I no longer had any critical tasks within the company. Selling advertising did not take much time, and I stopped all activity on the job listing site. It was time to turn the page. I needed to stop working on HashCode full-time and find a job as a programmer. At the same time, a few freelancers continued working for the company. These were mostly people who were involved in creating and moderating content and engaging users within already launched communities.

After a short period of job hunting and interviewing, I settled on LG R&D lab, where I would help develop an operating system for their TVs (to be precise, I was responsible for porting Google Chromium to WebOS). After working at my own company, where I had to solve problems for 10–14 hours a day, 6 days a week, an 8-hour 5-day a week job seemed like an endless paid vacation. No work with freelancers, sales, budget planning, *etc.* Everything was extremely simple—just sit and program.

For the first six months of working at LG, I worked as a programmer during the day, and moonlighting as an entrepreneur in the evenings and on weekends. My focus within HashCode was setting community development goals for the remaining team, selling and servicing the ads deals, and doing a bit of bug fixing. I think that in general, it is possible to work as a programmer and run your own company without compromising the primary work, but you need to be rested and motivated. The previous year had exhausted me physically and mentally. I felt that the time had come to turn over yet another page in my life and to leave the project completely.

I thought that simply closing the project was not the right thing to do. I felt responsible for the time the users had invested in the community, people who believed in me and actively supported development of the project for several years. I couldn't let them down. So I wanted to hand over the project to someone who could take care of the community and revitalize it. I had two potential buyers for the project in mind.

The first one was Rabodubr (not the real name of the company). Rabodubr was one of the most popular news sites in the Russian language at the time dedicated to the IT thematic. It was collectively maintained by the community. The project was on the rise and at some point, they launched a dedicated Q&A site on all IT related topics. I talked to the founders, but we did not find common ground. I think the reason was that although Rabodubr had a Q&A website, working on a Knowledge Network did not fit their profile. Rabodubr was a media platform, not a knowledge base platform.

The second potential buyer was Stack Overflow itself. At that time, Stack Overflow was only present in the English language and proposing to start localizing the site, and especially by absorbing our community, was a venture. Still, I found Joel Spolsky's contact information and wrote him an email. I did not receive a response and two weeks later I sent a follow-up email. Unexpectedly, Joel answered almost immediately. The essence of his answer was something like this: "Thank you for the offer, but if one day we want to have Stack Overflow in other languages, we will do everything ourselves."

"At least I tried…" I thought and continued working on the community. About six months later, something unusual happened. Stack Overflow announced the launch of its first international site, Stack Overflow in Portuguese. The event was unusual for two reasons. First, at that time there was the belief that every programmer in the world spoke English. The founders of Stack Overflow said in one of the project blog posts that they considered English the de facto language of the industry and deliberately did not create localized versions of the community. In addition, from Joel's answer it seemed to me that the folks did not plan to launch a localized version of the site six months before. It was clear that something had changed within the company. I decided to try writing to Joel again.

Imagine my surprise when he replied to me and referred me to the VP of Community, to see if we could agree on something. After that I had several interviews with the VP of Community, and then with Tim Post, Adam Lear and a few other of my future colleagues. Unexpectedly, everything went very well and we found common ground! Folks from Stack Overflow did an audit of my communities and we agreed that the communities of programmers and linguists would migrate to the Stack Exchange platform, the community of mathematicians would continue to live their lives on the old platform, we would close the remaining communities, and I would move to work for Stack Overflow to continue growing communities in the Russian language.

The situation was ambiguous. On one hand, everything was great. The community got a second chance and I got a monetary reward for several years of hard work. On the other hand, I was leaving programming again. I really liked working at LG, I enjoyed every day I spent in the company. On top of that, I had a wife on maternity leave and a three-month-old child in my arms. Working at a startup meant long hours and lack of stability. At that point in time, I did not know whether I should be happy or not. Nevertheless, the choice was made, the card was played. In

November 2014 I wrote a letter of resignation and went to pack my suitcase for a trip to the United States to meet my colleagues.

*Note: When I was already working at Stack Overflow, my colleagues told me their anecdotal version of why Stack Overflow decided to launch international communities. It all started when Joel went on vacation to Brazil, where he decided to meet the community. He was surprised when so many people there either spoke very little English or did not speak it at all. It turned out that English was the language of industry only in an idealized world. In reality, most programmers are passionate about technology rather than learning languages. After this, the team saw great opportunities to scale Stack Overflow. It was decided to launch several international communities for a test. The chosen languages were Portuguese, Spanish and Japanese. Folks quickly found a community manager for Stack Overflow in Portuguese but could not find a person who could speak Spanish and English fluently, had an understanding of the domain (programming) and knew how online communities work. The search for a community manager dragged on for more than six months. It became obvious that finding a community manager to grow an international Stack Overflow is not very easy. When I contacted the folks again, I looked more attractive than before in light of their experience searching for a Spanish-speaking community manager. Moreover, I had a unique wealth of knowledge, real experience in community building and an already developed community. It was definitely a win-win deal.*

*Note: It took one year from my first email to Joel Spolsky until the deal was completed. Which once again proves the rule: Selling something to companies is a lengthy process.*

*Note: Don't be afraid to become "uncompetitive." When my HashCode partners and I discussed that I would need to remain involved in the monetization of the project, I was tormented by the fear that during this time my qualifications as a software developer would get rusty, since for at least six months I would have to deal with sales, not programming. In fact, "loss of qualifications" is only half the story. The second half is that you in return acquire a huge number of other skills that are extremely difficult to obtain working as a software developer in a large company. This includes the ability to communicate with people and sell, first of all, yourself. Here is my personal example. Before starting to work on HashCode, I was thinking about changing jobs and went to interviews from time to time. Although I was able to solve technical problems at home with ease, I failed oral interviews in offices when I had to solve something in front of other people who were right in front of me. I was simply lost due to my natural shyness and made stupid mistakes. When I started going through interviews after leaving HashCode, I received 7 offers that I liked within 5 days.*

# Part 2

## Playing in the Premier League

We were building HashCode using our own meager savings. As a result, I'd spent the last few years cutting my spending on everything I could. The community was literally assembled from matchsticks and acorns. We used open-source projects any time it was possible, developed software ourselves only as a last resort, and hired freelancers only for the work that was not economically justified to do ourselves. In my personal life, I also lived very modestly. I rented a 14-square-meter room, which was for me a bedroom, a home office and a dining room at the same time. Although Stack Overflow was a well-known project and I knew that the folks raised money from a few venture capital firms, somewhere on a subconscious level I somehow thought that Stack Overflow was exactly like me, just living in the United States, working on the community from their bedrooms. Imagine my surprise when, having arrived to meet the team, I found myself in a chic office located in the financial district of New York with a view of the Brooklyn Bridge, with its own canteen and free drinks. After meeting my amazing new colleagues, I got the feeling that I was a guy from a street football team who somehow miraculously won a trip in the lottery to the training camp of the best team in the Premier League.

I joined Stack Overflow in November 2014. We spent the next six months preparing to migrate the HashCode community to the Stack Overflow platform. On March 31, 2015, we set up a redirection from hashcode.ru to ru.stackoverflow.com, sent out physical letters and T-shirts with the Stack Overflow logo to the top users and an email with the news about the migration to all other users, and posted the official announcement that HashCode from now on will be known as Stack Overflow in Russian. Hooray!

One of the main reasons for selling the community was to breathe new life into it, but there were others. I also believed that the deal with Stack Overflow was great because the acquisition had the potential to solve two problems that I felt were holding back HashCode's growth.

The first problem was that some developers looked down on HashCode. These people were convinced that all "real" programmers speak English, and those who do not speak English were simply beginners or just bad programmers. (The fact that they themselves did not speak English for some reason did not bother them in any way.) HashCode was created specifically for Russian-speaking developers. As a result, those people concluded that only beginners or amateur programmers participated in the community. In reality, of course, this was not the case. The second problem was that over the past couple of years, the engine of Stack Overflow had come a long way, while our engine had remained almost unchanged. We did not have a chat system, there were no review queues, no suspicious vote detection system, and a lot of other essential functionality.

I expected that the Stack Overflow brand would solve the first problem by "equalizing" the Stack Overflow in English and Stack Overflow in Russian communities, and the company's excellent technical infrastructure would give the community features that users have been asking for for so long. I thought that after the migration I would see an increase in activity on the site simply due to the community becoming part of the Stack Overflow family.

As it turned out, both hypotheses were wrong. No one began to love our community more simply because we changed the name and provided more features on the site. Obviously, the success of Stack Overflow in English was not in the software or brand itself. I didn’t understand what exactly the root of the success of the community was, but I really wanted to figure it out.

*Note: Looking back, it seems quite logical to me that software does not make a community successful per se. After all, there are many communities using the same software platform under the hood. Some of them are successful, others are not, even within the Stack Exchange network.*

When we launched HashCode, we were interested in trying something new. With a genuine interest in social systems, an engineering mindset and a huge amount of perseverance, we were able to build a small self-sufficient community. Unfortunately, the growth of the community has hit the wall of our unprofessionalism in community management. At some point, it became clear to me that in order to play in the Premier League it is not enough to run fast and hit the ball hard, you need to know where to run and when to hit, in other words, you need to understand the game as a whole and in all its smallest details at the same time. To make Stack Overflow in Russian successful, I needed to improve my theoretical knowledge about online communities. I put everything aside and started studying community management from all the sources I could find access to.

In retrospect, I can say that community management in itself is not difficult, but it is counterintuitive. Throughout the rest of this part of the book we are going to take a look at the most important discoveries that I made while working at the company to grow the Stack Overflow communities. It seems to me that the biggest difficulties with growing communities arise from missing the very idea of an online community, that is, why communities are needed and why people participate in them in the first place. Another big difficulty is that it is hard to realize that users in online communities are volunteers. Volunteers won’t do anything they don’t want to do. At the same time, in developed communities, absolutely everything is done by volunteers. Therefore, without the ability to ask for help from users and coordinate their joint efforts, it will not be possible to grow an online community. The growth of any social system is associated with the need to develop and introduce new rules to maintain a given culture in the community. Thus, it is critically important that the rules you create have support from the community. Conflicts will arise in absolutely any community. You should not be afraid of them, but it is important to know how to resolve them for the benefit of the community. At the end of this section, we will briefly talk about the role of software and what health metrics could look like in the context of online communities.

Let’s get started!

# 1. Community is the solution to a pressing social problem

The first step to solving the puzzle of online community growth for me was mastering the basic theory of sociology and social psychology. It made it possible to decompose the large task of growing a community into smaller building blocks that could be worked on. Below are the main ideas that helped me understand what made Stack Overflow become the largest community of software developers on the Internet.

## There are two main forces that bring people together into groups: interest in a topic and a desire to have close connections with other people

I define an online community as a group of people connected by a common goal, interest, or pursuit who interact with each other in the same environment. All online communities can be divided into two large groups based on the type of force that pushes people to participate in the communities.

## 1. Topic-based communities

Topic-based communities are built on the basis of users’ passion for a particular topic. The following can be said about topic-based communities.

- People associate themselves with a topic through participation in the communities dedicated to this topic.
- The more relevant users’ background or experience is to the topic of the community, the more they like the group.
- The more people identify themselves with the topic of a community, the greater their involvement in the community.

Any hobby community or community of practice falls into this category.

## 2. Relationship-based communities

Relationship-based communities are formed on a desire to build close relationships with other users of the community. Some of the properties of the relationship communities.

- In relationship-based communities users tend to form small tightly connected groups.
- Users feel attached to other people in the group rather than the community as a whole.
- Off-topic conversations and personal messaging increase users' involvement in relationship communities.

An example of this type of community is forums for mutual support where people with similar difficulties help each other not to give up and exchange their experiences in fighting some shared problems.

Although both types of forces play a large role in engaging users in any community, each community uses only one of these two driving forces as the basis for building a group.

For example, the main unifying force on Stack Overflow is the love for programming. People for the most part come to the site in order to satisfy their professional interests, and not to have a chit-chat about life with people they are interested in or find new friends. At least on the main site.

## Successful online communities are meritocracies

Stack Overflow, like most other successful online communities, is a meritocracy. Meritocracy is a way of organizing a group in which people are given a place in the group’s social hierarchy based only on the work they do, regardless of anything else. In such systems, the rules of the game are clear and the same for everyone, and everyone can participate.

Do you want to become a community superstar and gain universal recognition? You are welcome! Here are a couple hundred unanswered questions. All in your hands!

## Extrinsic and intrinsic motivation

There are two types of user motivation in online communities:

- *Extrinsic motivation*. Extrinsic motivation helps stimulate short-term interest among existing users who are already interested in the community. Extrinsic motivation is based on incentives and rewards, which are divided into two main categories: tangible (swag, gift cards, etc.) and intangible (special status, additional privileges on the site, etc.).
- *Intrinsic motivation*. Intrinsic motivation is based on the feeling of satisfaction and pleasure that a user gets from the very process of performing an activity.

There are two broad categories of intrinsic motivation:

- *Personal motivation*. Primary components of personal motivation are a sense of autonomy, the ability to decide what, when and how to do something, and a sense of competence, the ability of being good at what you do.
- *Social motivation*. The most prominent social motivators are the ability to connect with other members of the

community and being able to share your work with peers and help them.

Extrinsic motivation always plays a secondary role. Whether you’re planning to create a new community, increase the number of users, or increase the engagement of existing users, you need to think in terms of intrinsic motivation.

## What an online community consists of: mission, tools and social norms

Any online community can be seen as a product that consists of three main ingredients: the community’s mission, the tools available to the users, and the social norms of the group.

The community mission is the most important ingredient, without which everything else is meaningless. Online communities often arise in response to pressing social problems that begin to concern many people. Social problems, like any other problems, can be solved in several ways. So, a community mission statement describes a proposed solution to a problem that you invite people to solve with you and other users on your site. A mission statement answers the question “why” people should join your community and is the community’s unique value proposition. The community will attract exactly the kind of people who care about the problem being solved and who believe that the solution you propose in your mission statement is worth working on.

Usually, people form groups only in situations when it is difficult or impossible to achieve something on their own. The reason is that the existence of any group is associated with overhead costs for its creation, management and facilitation of the interactions between the users. In online communities, software helps to overcome these barriers. Software tools determine what opportunities users will have in terms of interactions with each other, how they will form and manage groups on the platform to achieve the community’s mission.

Social norms determine the “rules of the game” in a community. The tools and mission of a community do not completely determine the behavior of people in a group. Different communities using the same software may have completely different cultures, which are determined by social norms.

## Attainable and unattainable missions

A mission of a community can be attainable or unattainable.

- Communities with an *attainable mission* have a very specific end criteria. When the mission of such a community gets attained and the original social problem ceases to exist, the community loses its relevance and often ceases to exist along with the problem. They usually are devoted to one-off events. For instance, the community of volunteers who help with hosting the Olympics has an attainable mission.
- Communities with an *unattainable mission* are usually focused on changing people’s behavior, attitudes or other processes that without constant coordination of the public effort, risk returning to their original state. For instance, Wikipedia has an unattainable mission, not because amateurs cannot create an encyclopedia, but because for Wikipedia to exist people need to work on it indefinitely.

## Antagonist project as a brief description of what your community does

Social problems, like any other high-level concepts, are not easy to explain briefly and succinctly. One way to make the mission statement of your community more tangible is to find an antagonist project that currently benefits the most from the existence of the problem you are attempting to solve. If the problem you want to solve is real, there are bound to be specific projects that benefit from the existence of that problem. You can define your community as something that opposes the chosen antagonist. In this case, the simplest measure of community success, especially in the early stages of development, is the measure of the community’s progress against the antagonist position.

*Note: The use of an antagonist project is based on a feeling of hatred. Hate and fear are some of the most powerful emotions that effectively activate people. They should be used in exceptional cases. If you feel that it is the case, try to find topics that are based on the users’ desire either to avoid a future that scares them or change the present reality that they do not like much. Initiatives proposing work on such topics as these are effective in activating users and are compelling in their own right.*

## Stack Overflow is a solution to a pressing social problem

Before Stack Overflow, there were two main ways to ask questions about programming online. The first was to ask a question on one of a dozen online forums. The second was to ask the question directly to the developers by email. Search engines indexed all existing discussions on forums, as well as archives of email correspondence with developers that were publicly available. Thus, subsequent generations of programmers who encountered the problem that had been already discussed had the opportunity to find a ready-made answer fast.

The problem was that the mechanics of both asking questions and searching through the available information on the Internet were not optimal. On forums and by email, people did not always answer the topic of the question, and many new questions were asked in replies to some other questions in a random place in the threads. To understand the context of a thread, it was necessary to read several pages of text before and after the post to which the search engine sent you. On top of that, some people asked the same question on several sites at the same time, which further clogged the search results. As a result, when a programmer encountered a problem that they could not solve, searching for an answer could easily take a week, during which they would read hundreds of pages of text. I think it is reasonable to say that the entire Internet at that time was an unstructured dump of information.

Many companies have tried to solve this problem. One company was Experts Exchange. Experts Exchange provided a platform where people could ask and answer questions more effectively than on forums and developer mailing lists. Everything would be fine if there weren't one "but". Experts Exchange allowed search engines to index all created content on the site, but when a real person visited the site, the site only showed some part of a question, but everything else was blurred. The site required a paid subscription to access the remaining content. This clogged the search results with irrelevant information even more. Dissatisfaction with the situation among developers had reached its limit.

Let's sum everything up. Around 2007–2008, there was a pressing social problem. The Internet was a dump of information, which took a lot of time to search through. The problem worried many developers so much that they were ready to act immediately. Stack Overflow's solution to this problem was to work together on a free community knowledge base in the question-and-answer format, with the end content optimized for reading. Stack Overflow's mission, a unique value proposition and a specific solution to the social problem, is "To build a library of detailed, high-quality answers to every question about programming." (Note that this is an unattainable mission aimed at changing behavior, the way programmers approach problem solving on the Internet, and the community could, in theory, last forever.) Stack Overflow defined itself as the anti-Experts Exchange. Anyone who felt angry about Experts Exchange could help in its disappearance from search results by participating on Stack Overflow. In his blog, Jeff Atwood' wrote "We're like Experts-Exchange, but without all the evil."

Now let's look at Stack Overflow in Russian and see why the community did not grow as rapidly as Stack Overflow in English. The way Stack Overflow in English was launched was very different from how and why HashCode was launched. Stack Overflow in English was launched as a solution to the specific pressing social problem that we discussed above. On the other hand, several years after the launch, at a time when we were actively working on HashCode, the original problem being solved did not look as critical as before. The Internet was no longer an unstructured dump of information, Stack Overflow in English was constantly present in search results, and almost no one remembered about Experts Exchange anymore.

# 2. From marketing to community management

## Community management is all about working with users and not about marketing

After migrating to the Stack Overflow platform, I continued working on growing Stack Overflow in Russian as I understood “growth” at that time. Most of the activities that I did came down to spreading information about the community to the target audience. These were speeches at conferences, information partnerships, joint competitions with partner projects, *etc.* In other words, I approached community development as a marketer and thought of the community growth in terms of “creating funnels” of users. My understanding of what needs to be done to make the community grow began to change when Stack Overflow in Spanish got their community manager, and soon a good friend of mine, Emilio (not their real name), onboard.

Emilio’s approach was very different from mine. I was looking for channels with the largest audience reach, Emilio was making a podcast for his community of dozen people. I spoke at conferences in front of several hundred people, Emilio helped first users organize offline local meetings for a couple of people each. I wrote articles on popular third-party blogging platforms, and Emilio interacted with active community users one-on-one. At the beginning, Emilio’s approach seemed very strange to me. Then Stack Overflow in Spanish started growing like no other community on the Stack Exchange network. Emilio’s approaches definitely worked and were very effective. It made me do the same. I launched a webcast for Stack Overflow in Russian, started organizing offline meetups and completely abandoned the terms like “audience” and “reach”.

After working with Emilio for some time, I came to the conclusion that, in general, marketing is about attracting new users to an empty community, while community management is about working with existing users. Marketing is critical at the stage of launching a community, while no one knows about the community at all. At the same time, marketing will not help communicate goals to the community, nor will it help with engagement or retention of users on the site.

## What does a community manager do?

Let’s talk a bit about what companies usually expect from a person when they hire a community manager. I define community management as a set of activities that make a community work efficiently. What a community manager does at a particular moment largely depends on the state and the needs of the current community, which is determined by the number of active users involved. There are three main stages of the lifecycle of a community during which a community manager needs to deal with different types of tasks:

- *Inception*. At this stage the primary tasks of a community manager are to set up the site for others, start acquiring first users and establish the desired culture by onboarding new users, moderating the site and facilitating meta discussions.
- *Growth*. This stage starts when a community reaches its critical mass of users which means that all tasks in the community are performed by the users themselves. The

more users the community continues to acquire, the more social tension and the need for new rules there will be. At this stage, a community manager needs to change their focus to critical activities like defining new rules, proactively looking for volunteers who want to help organize the community, highlighting desired behavior, *etc.*

- *Maturity*. When it becomes clear that users with different views do not have enough space in the same community, a community manager needs to focus on scaling the community by splitting the monolith into subcommunities and planning their joint integration. After the split, the community manager should start helping each of the subcommunities to grow independently and repeat the cycle.

## Three key skills every community manager should master

To successfully grow a community, a community manager must be fluent in three key areas.

# 1. Relationship building

In sociology there is a term, *social capital*. Simply put, social capital shows the difference between how much you can accomplish without your social connections and how much you can accomplish with your social connections. The greater your social capital, that is, the better your relationships with colleagues in the company and activists in the community, the more you can achieve.

The essence of most community initiatives that you as a community manager work on are requests to users to help you with something. You need to identify a group of users who might want to help you when you ask them and build a good relationship with them. The better the relationships, the more time and resources the users will be willing to contribute. Inside the company the situation looks the same. You will constantly need colleagues' help in order to get done what the community users have asked you for.

# 2. Communication

Inside Stack Overflow, the company, we often said that a community manager is a bridge connecting two banks of a river: the company and the community. The community manager’s job is to represent the interests of the company among the community and the interests of the community within the company. The task on each side is to gather information and present your most important findings to the other side. To do this, you need to constantly communicate with people who have the context, your colleagues and volunteers in the community.

There are two primary communication skills that a community manager should have. The first is the ability to tell short and engaging stories that ignite users’ imagination and spur their actions. The second is the ability to get others to talk. The more frequently others talk to you, the more willing they will be to listen to you when you talk to them.

# 3. Managing conflicts

The biggest danger to communities comes from the community itself. The name of this danger is drama. Drama drains trust and any motivation to do anything together, it polarizes users, making them dislike each other and leave the community.

There can be two sources of conflict in a community.

1. *Community users*. In any community there will be misunderstandings between users from time to time and there will always be difficult people present. The community manager needs to know how to work with both and keep the community productive.
2. *Employees of the managing company*. When working with online communities, your first priority is to avoid making mistakes yourself and keep your colleagues from making mistakes that could lead to conflict situations with users. When someone from the managing company makes careless moves that harm the community (ill-conceived new moderation rules, public statements that offend some users’ dignity, etc.), a conflict between the company and community begins. When a conflict occurs, the energy of volunteers moves from creating value on the platform to an enmity towards the company and its employees. Even the most effective resolution to this kind of conflict rarely brings the community back to its original state. A lot of users lose their trust in the managing company and some users may even leave the community forever.

## A sense of belonging

Developing a sense of belonging among users is often the end goal of many community building activities. I define the sense of belonging to a group as the feeling that one is accepted and supported in the group, they are engaged and active, and identify themselves with the group. Sense of belonging is closely related to a sense of ownership, that is, when community users think that this is their community and they, through their actions or inactions, influence its future.

There are two broad directions of work with the community that impact users' sense of belonging.

## 1. Providing the opportunity to influence the community

A sense of belonging can only be developed by active users and is a combination of activities on the main site and meta activities to maintain and develop the community itself. Meta activities are only possible if a community is in active development and, thus, the users have the opportunity to participate in discussions about the future of the community. This could be a discussion of new features on the site, a discussion of changing the rules, or anything else. Important: all discussions in which users come to a decision must have a result, a change or an explanation of why change is impossible. Without this, at some point, users will cease to be active.

Stack Overflow users have been involved in the decision-making process since day one. For example, the domain name and logo of the future site were chosen together with the blog readers of the project founders. When the community was launched, many of the site's mechanics, even before implementation, were put up for discussion with the community and anyone could offer their own implementation option or comment on the ones that have been already proposed. Of course, not all what was proposed by users were accepted for one reason or another, but users always had the opportunity to share what they think and get feedback from the developers.

When I started growing Stack Overflow in Russian, the engine was almost completely finished and no resources were allocated to improve it. Even though the users did not have the opportunity to make changes to the site's logic or implement new features that our community needed, they could translate the site user interface, create unique help center articles for the community, decide what the topic of the community is and what is not, and much more.

## 2. Getting participants to know each other

We feel the sense of belonging not to a domain name, but to a specific group of people. This is also true for topic-based communities (like communities of practice). As a general rule, the better relationship users have with others on the site, the more engaged they will be in the community. The best strategy for getting users to know each other is to conduct collective unifying activities. These could be joint contests aimed at improving content quality on the site, offline gatherings, interviewing the best community users for a video blog, posting on meta about the achievements of specific users, or any other activity that helps you to share some information about the users from a positive perspective.

# 3. The main driving force of any community is volunteers

While growing HashCode, and later Stack Overflow in Russian, I did a lot in the community myself. I improved the quality of posts, removed spam, and asked questions on topics that had not yet been presented in the community. Things were different on Stack Overflow in Spanish. Emilio was not an expert in programming and even if he wanted to, he could not help the community. Instead, Emilio had a list of tasks that needed to be done to grow the community. Next, he found users who would like to help the community with the necessary activities and asked them for help. It was a fair exchange. Volunteers enjoyed the activities, got gratitude from other users for helping the community and high status in the group. On the other hand, Emilio was gradually gaining a more and more developed community.

As it turned out, Emilio's approach is the only way to develop a healthy community, because the main driving force of any community should be volunteers. Volunteers are the people who care about the mission of the community and the community itself, and who have the time and energy to help with its development. If you want to grow your community, you need to carry out initiatives that are interesting to volunteers and useful to the community. There are three main approaches to creating and running such initiatives.

1. Design activities yourself and convince users to participate.
2. Help users to connect and form groups based on the shared interests.
3. Find leaders among the users and help them with the initiatives they propose.

Let’s take a look at what each approach looks like.

## Designing activities: Be persuasive when inviting users to participate in initiatives you have designed

Once you have an understanding of what improvements you want to make in the community, you can try to convince volunteers to help you make them happen. The main problem with this approach is not that there are no people willing to help improve the community, but that those who want to help are inactive for one reason or another. For example, users may not know that the community needs help, be unsure that they have the skills to help, or be unaware of the importance of each user’s contribution. The job of a community manager in this case is to enable people, make them enthusiastic and convince them to take an active part in one of the suggested initiatives. The key to this is to be persuasive in your requests.

What do I mean by “be persuasive”? Let me explain with an example. At the end of 1978, an interesting experiment was conducted. Psychologist Ellen Langer stood in line to copy some papers by a copier. She asked to skip in the line. She skipped 60% of the time if it was a direct request like “Sorry, I have five pages. Can I use the copier without waiting in line? As soon as she added a reason she should skip, the success rate increased to 94%. Moreover, the reason could be arbitrary, even absent. For example, a reason like “Sorry, I have five pages. Can I use the copier because I need to make multiple copies?” did not add any information, but she was successful in almost the same 93% of cases. In other words, we can get completely different results if we ask people for something in different ways.

*Note: Persuasive techniques are based on exploiting the non-rational component of how people make decisions. This often raises questions about how ethical it is to discuss this topic. I think the issue of ethics is largely a personal matter. We are all different, raised in different cultures. I personally think that healthy communities bring a lot of benefits to humanity (who has never used Wikipedia or Stack Overflow?) and users take pleasure in participating. Building a community is not a trivial task. I personally don’t see anything unethical in sharing knowledge and discussing how to build communities effectively and for the benefit of other people.*

In software development, we have design patterns that must be followed if one wants to make their program code easy to understand, maintain, and improve. Similarly, in the world of online communities, there are rules that must be followed in order to make users participate in ongoing initiatives. Below are the most important rules to help you create initiatives that maximize the users’ likelihood of participating in them.

Goal, feedback and public benefits of an initiative are a sufficient minimum to activate users

To effectively activate users, an initiative that you are proposing must:

- *Have a specific end goal*, achieving which will be interesting to the users.

- *Provide feedback* on the progress to the goal of the initiative.
- *Clearly explain why the initiative is important to other people* that the work you suggest will result in high value for the community.

Or simply put, goal-based initiatives that provide users clear feedback on the work they do and its quality combined with a socially useful end result are sufficient for most initiatives to get users to participate.

Users must see the personal value in an initiative and feel that its end result is valued by general public
A required condition to make volunteers engaged in anything is that they should clearly see that their contribution is valued by those they help, as well as by the community at large. If you want to engage users in a collaborative activity, the outcome of the initiative must be seen as valued by everyone involved, including those who get the value from the end result and those who participate in the activity.

Explain the reasoning behind your requests when reaching out to regular users
Regular users do care about the end results. Almost every regular user will assess the initiatives you propose individually. When you want to engage regular users, arguments and rationalization are a necessary part of communication.

When reaching out to casual visitors, rely on heuristics
Casual visitors do not tend to do in-depth analysis of the proposed initiatives. When you want to reach these users, your communication should be as simple and short as possible. Reasons and rationale in this case do not create a better

response. Moreover, they sometimes cause users to perceive the arguments as hidden manipulation.

Additionally, when you talk to casual users, simple and short requests are effective when communicating initiatives related to:

- Something that the users do not have a strong personal view about.
- Something that assumes a small contribution.

## Make the work of volunteers visible to the other users

When you want to persuade users to participate in an initiative, you can use the social proof heuristic. To do this, make sure that users see that there are other users who take actions you are asking for in the initiative. Highlight the most active users who are participating in initiatives in digests, interview them about their personal experiences, share progress metrics with the community, *etc.* The fact that some users are participating gives the others confidence that the proposed initiative has a chance to succeed and creates the desire to join themselves.

## Ask specific people for specific help

To make your requests more effective, ask specific users for help with specific well-defined tasks. The smaller the group of people to whom the requests are addressed, the higher the likelihood that people will respond to the requests. When asking for help, create a logical connection between the success of an initiative and something that the target users value. Ask for help from those users who are interested in the tasks you suggest working on and have all the necessary skills to complete.

## Other things to consider in the context of initiatives

## Help users to connect and form groups based on the shared interests

Connecting users to others is one of the most powerful ways to keep people in the community and stimulate their activity. The more effectively users create connections between each other, the faster the community will grow. Relationships (the connections) are created through shared, hopefully, positive experiences. A user’s feeling of belonging to a group directly depends on what kind of relationship and with what kind of people they have.

Usually, connections occur naturally by chance when users with similar or complementary interests happen to be in the same place at the same time, have the opportunity to act, and are not afraid to do so. To accelerate the growth of your community, you can intentionally bring together users who can positively influence each other and create opportunities for them to interact. This approach is called “network weaving”. Below are the main ideas of how to be successful at it.

### Keep an eye on users’ interests and needs that can complement each other

When you see users who have some needs and other users who have an interest in meeting those needs, and these users do not intersect naturally or intersect but not in a timely manner, you need to help with coordinating. The easiest approach is to maintain relationships with active regular users and know the types of activities they prefer to act on. When these activities appear on the platform, notify the users so they can show off their skills. This can be done manually by you, by volunteers, or via some automatization if your platform allows it.

### Create opportunities for joint activities

When you see users who, due to the patterns of their participation, have little overlap with each other in the community, but who would be interested in getting to know each other, you need to create opportunities for these users to be engaged in a shared experience on the platform.

The simplest approach to organizing a shared experience is to impose time and place limits on some type of activities that the users are interested in for some period of time. The type of activities and the place do not really matter, the goal is to do something useful together.

Typical activities that encourage shared experience:

- *Contests*. The primary requirement is that the contests are interesting to the selected segment of users. Usually contests have a theme, clear rules for winning and a timing. You can use some branded swag as prizes.
- *Seasonal events* are activities dedicated to some widely recognized public holidays or any important dates for users.
- *Improvement activities*. Any activities aimed at improving the structure and quality of content, identifying unfound violations of the rules, removing unnecessary content, creating missing content, *etc.* All kinds of activities that users would like to do, but never got around to.
- *Meta-discussions* about important aspects of the platform and the community that users care about and have personal opinions on.

Tell users about other users

To build deep, trusting relationships, users need to know the others as real people. This requires users knowing some interesting things about other users in a way that causes good feelings and empathy and having a chance to self-disclose about themselves and their interests. Such conversations rarely occur naturally, since this topic goes beyond the scope of most online communities.

Your task is to organize activities that introduce users to each other without harming the community itself. Here are some activities that stimulate building relationships:

- *Highlighting users' achievements in social media*. You cover almost any event that somehow shows users' positive contributions.
- *Hosting offline meetups*. If possible, attend them in person yourself. If you cannot make it, find some volunteers you trust who are able and willing to organize the events. Have some swag prepared. If the meetup is organized by volunteers, send them swag so that they can distribute it to the attendees. The meetup can take place anywhere, from the office of some company to a random cafe. There is no need to have any strict theme for a meetup. Providing users a way to get to know each other is the main goal. Take photos during the event and share a story afterwards.
- *Creating a media channel dedicated to the users themselves where you will post stories about the users*. It can be in any popular format like a webcast, podcast, text blog or something else. Interview active regular users, giving them an opportunity to talk about everything that worries them, starting from the current industry problems up to their pet-projects or their favorite outdoor activities.

## Find leaders among the users and help them with the initiatives they propose

You can come up with initiatives yourself and invite users to participate. At the same time, you can approach the task of growing a community from a different angle and, instead of thinking through initiatives yourself, look for volunteers in the community who would like to become community leaders. (Yes, this is the approach Emilio mostly used.)

The peculiarity of this approach is unpredictability. Essentially, you invite users to realize themselves on your site and then you support their endeavors that benefit the community. At the same time, you have absolutely no control over what exactly the users want to do at a particular moment. As a result, there needs to be flexibility in your growth plans. At the same time, this approach is one of the most effective, since any initiatives activate the community more successfully if they are proposed or led by the users.

*Note: Initiatives led by users who have popularity, authority, or some other special social status in the community, as well as by users who actively help other users, will be especially effective in activating the community. The rule of thumb here is that the more users feel empathy towards the author of an initiative, the greater the activation will be among them.*

Here are the most important tips for working with leaders from the community.

### Look for volunteers proactively

There are two primary approaches to the search.

1. *Manual search*. Look for users who are already doing what you expect your future volunteers to do. Contact

each of them individually, talk with them about the needs of the community, and offer them an “official position” as a volunteer. Users might consider a place in the social hierarchy of the community as an additional reward for the work they already do.

2. *Ask for help publicly*. Reach out to the community in need of volunteer help. Describe specific tasks to be done, your expectations for future volunteers and what those willing to apply should do to express their wish.

Encourage users to become volunteers

All users in any community have unique talents. Your task is to provide them with opportunities so that they can show off their talents to the best of their ability. Users may be shy, afraid of criticism, or don’t know where to start. You need to support the users morally, instill in them the faith that they will succeed and should try.

The ability to positively impact many people through initiatives in your community is an effective motivator that may encourage users to become more active.

Make it so it is safe for volunteers to make mistakes

Never criticize or give your personal assessment of the initiatives volunteers propose. Volunteers are not professionals; it is fine for their initiatives to look amateur. You need to give volunteers the opportunity to try whatever seems appropriate to them. There is only one case when it is worth asking volunteers to postpone launching an initiative, when the initiative is outright harmful to the community. If you can make sure volunteers enjoy working on their initiatives, it is already a win.

Volunteers must be autonomous

Avoid initiatives that require a lot of time investment on your part. If you need to spend a lot of time on providing support for some initiatives, you won't be able to help many other people and won't have time to plan the next steps to grow the community. Your time is limited, this approach is not scalable.

Try to provide volunteers with access to all necessary tools and information they need. If this is not enough and you are required to do some repetitive activities that take a lot of your time, kindly ask volunteers to adjust the proposed initiatives.

Recognize volunteers' contribution to successful initiatives as much as you can

Social recognition and praise are positive feedback. You need to regularly tell the community about successful initiatives and their authors through all media channels you have. The more volunteers feel that their contribution is recognized fairly the more likely they will propose and participate in new initiatives. Visible examples of successful initiatives and public recognition of the contributors motivate other users to become volunteers themselves.

If something goes wrong with an initiative, cover for the authors and volunteers

Some users feel a sense of fear of making an error and are afraid of situations where they potentially can be publicly criticized. This keeps the users from becoming volunteers and prevents users from revealing their talents. If users can sense even the slightest confirmation of their fears, it will be very difficult for you to persuade them to become volunteers. Whatever happens, always support existing volunteers, cover for them publicly and coach them in private conversations. Volunteers should never experience any negative feelings of guilt if initiatives they participate in fail.

One effective approach to accomplish this is to create a culture in which mistakes are encouraged. The thing that might help is facilitating public discussions of situations where something went wrong, without mentioning specific users, with further proposals for improving the initiative.

## Maintain trusting relationships with volunteers

Building trusting relationships involves continuous communication with volunteers and helping them resolve their problems. Here are a few topics one can talk about with volunteers.

- Answer all volunteers’ questions, provide feedback when they ask for it, share your thoughts and experiences with them.
- Give volunteers extra information that is not publicly available.
- Talk with volunteers about the community needs and new opportunities for initiatives available.
- Ask volunteers for feedback about the community and what future they see for it.

At the applied level, trust takes the form of actions, on both sides. If you trust your volunteers you will find time to help them with what they need. If you are trusted by volunteers, they will become more engaged in the community and help you when you ask them for help.

# 4. Communication

## Two ways to approach meta discussions with users: Helping and persuading

Communication is the means by which everything else is achieved in online communities. On Stack Overflow, most of the communication with volunteers interested in the community itself took place on Meta. On Meta we discussed the mechanics of the site, moderation rules, we shared our ideas about the vision of the future for the community and received feedback from the most engaged users. We had two main approaches to working with users on Meta.

- *Supporting users by answering their questions*. The practical purpose of answering users' questions is support. At the same time, answers to users' questions are the most effective way to convey your vision for the community to the users. The point is that until you are asked, all your thoughts are considered unsolicited advice. On the other hand, the ideas expressed in your answers will look reasonable and most users will follow your advice, if you are persuasive in your answers. Your authority as a person who knows how communities work, is earned precisely through answering users' questions. For the community management team at Stack Overflow, answering users' questions on Meta has always been the top priority.
- *Writing proposals for initiatives and announcements of technical and social innovations that encourage action*. It's

> great to have a large, active community where users ask a lot of questions about the community and your only job is to guide them in the right direction. In reality, this will not always be the case. At the beginning of community development, while there are not many proactive users yet, you have to propose initiatives yourself. In addition, in developed communities, community managers constantly need to share with users information about new features and make users' want to adapt them, which is also done by proposing initiatives on meta.

From my personal experience, of all types of communications, writing posts about initiatives is the most difficult for community managers. At the same time, the ability to communicate what you think the community needs and be persuasive is one of the most critical skills for a community manager. We have already considered the question of how to be persuasive, so in this section, we will look at what a post that suggests working on an initiative should consist of.

## How to write posts that propose initiatives

Technically, there are three main ingredients that each post about an initiative should have: a story, a theme and a question or call to action.

### Story: If it is worth communicating, it is worth a story

Humans are narrative creatures, we perceive the whole world around us through stories. Stories allow us to experience information, as opposed to just consuming it. Stories can influence our beliefs, views, and behaviors. This will happen if the author of a post manages to evoke empathy and compassion in the readers through which they begin to relate with the emotional experiences the author offers to the readers in the story.

Here are the best practices for writing stories in the context of community management.

- Stories set the prism through which a reader sees a post. When working on a story for a post, we choose the narrative according to what we want a reader to feel and think after reading the story. Before starting to write, clearly define what you want readers to think about.
- In posts about initiatives, stories play an auxiliary role to set the context, while the primary focus should still be on the initiatives themselves.
- When sharing a story that has an example, share a story of one person. It is hard to imagine an “all-population’s problem”.
- Keep a story short, one paragraph at most. The best option is to make it in the form of a metaphor.
- Introduce the story at the beginning and refer to it throughout the whole post, if needed.
- A story should be positive, inspiring, and unifying. It may have a picture to illustrate the narrative.
- In a perfect world, all the stories you share somehow correspond to the primary mission of the project.

Theme: The most important thing you want to share

The theme of a post is what we wanted to communicate in the first place. It is the body of your proposal. Usually, it is in the form of a statement of a problem that you want to address. The theme may take up to one page of text, but the shorter, the better.

- The most crucial part of writing is to present the problem from the readers' perspective and highlighting the benefits that they will get out of helping you solve it.
- You need to make the readers feel empathetic and supportive. They should never be presented as the root cause of the problem. Moreover, there should not be any blame placed on anyone in the post. You should present the problem through possibilities that the readers can accomplish together as a group.
- Whatever the end goals of the current initiative, the problem that you want users to work on should be presented as a step towards completing the mission of the community.
- If you are suggesting solutions, you need to show why the solution is a viable approach.
- There should be just one theme per post.

Question: Narrowing the discussion
Often in online communities, a discussion starts with a talk about a certain problem, but quickly turns into a talk about some off-topic theme with all the consequences it brings. To keep the discussion healthy, you need to set the boundaries of what is on-topic for the current discussion and what is not. The best way to do this is to ask very specific and valid questions at the end. Vague or open-ended questions allow users to steer the conversation and feedback away from what we want them to focus on. The more specific your questions are, the more likely you will get the results you were looking for.

- Phrase questions in a way that makes users provide positive answers.
- Questions should ask for solutions to problems we are facing, rather than just ask for opinions.
- Questions should make users be able to show off their skills while answering the questions.
- Keep the questions relevant to the users so that they have expertise and knowledge to answer them.
- Do not ask more than three closely related questions per post.

Other best practices for writing posts about community initiatives

# 5. Community culture

The culture of an online community is a set of beliefs, values and norms collectively held within the group. Culture is passed on from one user to another through their interactions on the site, through the stories you tell the users about the community, and through public discussions about the rules themselves. Culture is one of the three main aspects, in addition to mission and tools, that determines how your community will look as a final product to the user. Communities with the same mission and using the same software can be radically different from each other if they have different cultures. For example, Stack Overflow in Russian is much more friendly than Stack Overflow in English. Stack Overflow in Japanese is more open to joint growth activities than Stack Overflow in Portuguese. Users on Stack Overflow in Spanish consistently showed more leadership in the context of initiatives than users on Stack Overflow in Russian. The reason is that the culture of each Stack Overflow community was unique.

While working on Stack Overflows, I came up with two main best practices for creating a community culture.

## 1. Like attracts like

No one starts participating in a community by reading the help center. People come to your site looking for the content they need. By reading existing content on a site, people unconsciously infer the basic rules of communication in the community, which are then complimented by their personal experiences of interacting with other users. It is not enough to simply describe the desired standards of behavior in the help center and expect some kind of culture in your community. You need to constantly moderate the site content and monitor how users in your community treat each other, so that everyone who likes the designed culture of your community remains on the site and those who do not, leave.

*Note: Besides moderation, one of the best ways to communicate desired behavior in a community is through storytelling. For example, at HashCode our goal was to create and maintain the culture that we described as the "atmosphere of knowledge". To implement this for our community, besides the constant moderation of the content, we told stories to users, explaining the culture of our community through different metaphors like "Everything is acceptable on the site as if you were talking with your fellow students in front of a university professor." Through those stories, most users easily understood what was acceptable and what was not.*

## 2. Leadership by example

All successful online communities are meritocracies, where authority is earned through action. If you want someone to do something in a certain way, start doing it yourself or find someone who can take your place as a leader and focus the community’s attention on the leader in every possible way.

In the case of HashCode, “atmosphere of knowledge” was a cultural innovation for online communities at the time. It took some time before we found those who shared our views on things, during which we very actively moderated the site. Seeing that we walked the walk, most users adapted the approach we proposed.

### The only source of new rules is the real cases in your community

Community culture is inextricably linked to community rules. Community rules define how users are expected to behave in a community. When we had just started HashCode a few rules were enough to make the group of dozen people follow some designed behaviors. As the size of the community grew, new types of interactions appeared and brought new behavioral patterns, including bad ones. As a result, we started to introduce new rules that regulated new behaviors that had caused repeated violations.

When I was growing Stack Overflow in Russian, I had Stack Overflow in English as a reference. At some point, it seemed reasonable to me to take time-tested rules from the English-speaking community and adapt them to our site. As soon as I introduced the first of such rules, I received a lot of negativity from regular users who might be affected by the new rule. (The point of most rules is to prohibit something under certain circumstances, and no one likes to be prohibited from doing something that they like to do!) Users saw adding new rules “for the future” as unfair and were very skeptical about them, people resisted tne change. I think some users were also frightened by the very idea of “adding a rule because another community had it” and the uncertainty that this approach brought. As a result, I scrapped the rule and have never tried anything like that again.

As I see it now, the working approach to creating community rules is to have a minimum set of rules when launching a community and then add new rules only after something has happened that has already caused some damage. In this case, there will be users who do not like the situation and they will help you promote a particular rule and ensure that it is adopted and followed by the entire community.

## Hosting public discussions of the rules makes the rules legit

In developed and healthy communities all moderation is performed by the users themselves. For any rule to have an effect, you need to make volunteers understand the rule and accept it internally. One of the most effective ways to achieve this is through public discussion. Rules that were created based on public discussions in which any user could participate have great weight and legitimacy. Moderators and users internalize these rules, follow them and compel others to follow them as well without any effort from the managing company. For this to happen, a community manager’s job is to encourage users to initiate public discussions about any violations they see in the community, and then they need to do their best to make sure that the community takes the right direction towards solving the issues.

This is exactly how Stack Overflow worked. Any user could ask a question on Meta at any time about moderation in general or about their specific case. Next, all interested users discussed the problem together and usually came to some consensus, which was then implemented.

Not all discussions about violations need to end with a new “official” community rule. Much of the discussion about social norms should and will remain in the form of discussions. Keep in mind that there will always be users who like a formal approach. They will demand that everything, down to the smallest detail, be described in the official rules. At the same time, if you start documenting every nuance of the cultural design, the number of nuances and exceptional cases that need to be documented will only increase. To prevent this from happening, the “official” rules must describe the general idea in such a way that most reasonable people can easily understand the implications of the rules.

Here is a list of things that usually need to be documented on the site as official rules.

- Rules that should not change over time. This is especially important for rules that are based on community consensus where users’ views are almost equally divided, and all the proposed solutions have their own pros and cons. You need to choose one thing, add it to the help center and then live with the rule. This helps to keep the community out of unnecessary disputes.

- Unique and counterintuitive rules that may cause controversy.
- The most important rules that shape the culture of your community that you want to pay special attention to.

The shorter your community's "official" rules, the more likely it is that someone will read them. After all, community culture exists in the minds of users and is transmitted through their interactions with each other. The help center plays only an auxiliary function.

## If one wants to change a rule, they need to prove that the change has a beneficial effect on the community

Imagine that you launched a new community with only the most basic rules in the help center. A problem arises in the community that requires a new rule. You start a discussion with users and see that there are several similar approaches to solving the problem. What will you do?

*Note: The truth is that almost any problem can have multiple solutions with comparative effectiveness. Therefore, you will encounter the situation described below all the time.*

I assume that you choose the best solution available, commit to it and follow the new rule created based on the suggestion.

The issue is that since there were multiple solutions to the problem, no matter which solution you choose, there will be those in the community who like one of the alternative approaches more. Sometimes they will propose changing existing rules or processes to the alternative they like. The proposed changes will seem logical, since the current approach has its disadvantages and the alternative approach has its advantages. If you agree to the proposed changes, you will open a Pandora's box of endless debates with users and wasted time with no ultimate benefit to the community since each approach initially had roughly the same efficiency.

The only way to avoid this is when you see a proposal to change existing rules, ask the author of the proposal to prove that their proposed solution is significantly superior to the existing one. If this is not the case, do not make the changes, as they will bring nothing but controversy to the community.

## Promote the culture of welcomeness and kindness

There is one universal rule that should be part of the culture of any online community. It usually sounds like this: “No matter what happens, we should always remain human” or “We should be nice to others and assume good intentions”. This rule is primarily expressed when users behave with restraint when something or someone causes them a storm of negative feelings.

Promoting being kind to others even if they did something wrong in one’s view is the foundation without which it will be hard to build a community. Throughout all my time working on communities, I have never seen a single case where anger and intolerance caused anything other than increased anger. At the same time, kindness often causes users to change their behavior dramatically in response.

Kindness between users sets the foundation for an inclusive, safe environment where people want to interact with each other. When users treat one another with respect, it creates positive connections not only between the users who are directly involved in a joint activity, but also everyone who observes the activity.

Here are some suggestions for promoting and maintaining the community culture of welcoming and kindness.

- *Delete all rude content as fast as you can*. Contact the violator and explain why you deleted their content and talk about the importance of treating each other with respect no matter the circumstances. Motivate volunteers to do the same.

- *Identify critical points in the community where most rudeness occurs*. Prepare a set of templates that users can employ when facing a situation that causes them to feel negatively. Templates should be both for content (what to say) and for behavior (what to do). Constantly talk with regular users explaining that rudeness causes new users to leave the community, and never leads to a change in one’s behavior.
- *Help users experience empathy and compassion*. Prepare a series of stories that show how users whose behavior causes negativity see the community. Show that these users have good intentions and would like to learn about the theme of the community.
- *Encourage regular users to make edits* when they see any issue with the content. This helps to show what is expected from users in a particular situation.
- *Initiate discussions about new rules, help pages, templates*, etc related to the welcoming vibe on your site. It is impossible to be too kind, especially online.

Reacting negatively to some content or behavior is as natural as posting content without knowing all the nuances of the rules of a community. When discussing negative behavior, do not blame the users who behaved rudely. Your task is to motivate users to be welcoming and kind, leading by example (i.e. how you treat users in stressful situations) and provide them with the required training programs and tools.

# 6. Having any goal is better than no goal at all

After migrating HashCode to the Stack Exchange platform, the community still needed to pass through the formal graduation process. Graduation on the Stack Exchange network requires that a community meets certain criteria, one of which is the percentage of answered questions. At the time of migration, the percentage of answered questions on Stack Overflow in Russian did not meet the network standards. At some point, volunteers from the community created an initiative whose goal was to increase the metric. The initiative attracted the attention of many users and within three months the community collectively raised the percentage of answered questions, so it passed the threshold.

Recall that a community is a way to solve a specific social problem. People become part of the community because they care about the problem being solved by the group and the mission of the community describes the solution the group is working on. The problem is that the mission is something big, and sometimes completely unattainable. For a community to grow, users need to have something they can achieve here and now. For example, helping their community go through the graduation process. This is why any community must constantly have some goals to work on. We can think of goals as steps that the community takes to achieve the mission or something that sets the context of what the community is working on collectively right now.

*Note: You may probably be confused here. How can we take a step towards achieving an unattainable mission? Very simply, through achieving the current short-term goals. If a community has an unattainable mission, then users are working on some process that needs to be maintained constantly (therefore the mission is unattainable). In this case, any goal that is aimed at maintaining this process is a "step towards achieving the mission." For example, the mission of Stack Overflow is to build a library of all programming questions and their answers. I think we all understand that it is impossible to collect "all" questions about programming, but if we don't motivate people to ask and answer questions on the site, then very soon our community knowledge base will become irrelevant and we will go back to spending days googling for a solution to a simple day-to-day problem. Stack Overflow is developing through achieving goals aimed at maintaining and optimizing the process of collecting its knowledge base.*

## Any goals are better than no goals

A community cannot function properly without goals. Users start losing interest in a community that has no goals and leave it. Another result of a lack of interesting goals is constant drama or constant negativity in the community. Interesting goals focus users' attention on solving specific applied problems, set the direction for collective action, and give people strong reasons to work together and find compromises.

As a community manager, you must set the growth vector for the community either yourself or with the key volunteers' help, but you need to do that. Any goals, even the simplest and not optimal, are better than the absence of them.

Once, Crayfish, Swan, and Pike
Got hired to drag a cart,
Harnessed themselves, all three, and start to move.
It could be pretty light for them,
But to a river Pike is trying to drag,
While Crayfish is pulling back,
And Swan up to the clouds directs its flight.
It Is difficult to say who is to blame.
Which place is now the cart? The same.

– Ivan Krylov

## Look for goals that are achievable, finite, and beneficial to the community to activate and unite users

The community should always have goals that users can help achieve. Any goals are better than no goals at all. At the same time, if you want the community to grow effectively, look for achievable, finite, and beneficial to the community goals that activate and unite the users.

Goals should activate and unite users:

- *Activation*. Essentially, by defining goals you are saying, "It would be great to do this because it will help us get there. Who's ready to help?" People who have personal motivation to achieve the goal become more active and more involved.
- *Unity*. Users perceive the same reality in different ways, placing their emphasis on different aspects of the community. Common goals bring users with different

perspectives together and allow them to move in the same direction as a group.

Goals should be achievable, finite, and beneficial to the community:

- *Achievable*. Goals form feedback loops. When users achieve goals together, everyone experiences a collective positive experience. Users begin to have more faith in themselves, other users and the project as a whole.
- *Finite*. Users must clearly understand if a goal has been achieved or not, how much and what still needs to be done to achieve it. Otherwise, users might soon lose interest.
- *Beneficial*. Our task is to help the community fulfill its mission. Any initiative that the group undertakes must be somehow related to the end mission of the community. Users must have a thorough understanding of how the community as a whole will benefit if they invest their time in achieving a goal.

The choice of goals depends very much on the stage of development of a community. They change dynamically since what the community can and needs to achieve depends heavily on the current state of the community, its active users and their interests.

For example, I once proposed to users of Stack Overflow in Russian translating questions from English into Russian. To do this, I needed to have a consensus among the entire community that it was a good idea. When I started a discussion about allowing translations the first time, the proposal was met with criticism, the idea did not take off. A year later, I proposed the initiative again and it was supported by almost everyone. Users who did not like the initiative were not active in the community anymore, and new users saw a lot of benefit in having some good questions from Stack Overflow in English on our site. Community is a living organism that changes over time.

# 7. Conflict situations in communities

Conflicts for online communities are like athletic injuries for athletes. There is not a single athlete who has not been injured during their sports career. But injuries are very different. Some injuries can be treated, go away without a trace and teach the athletes something. Other more serious injuries, although can heal to some extent, stay with the athletes until the very end and are reminded of them from time to time. There is also a third type of injury that cannot be cured and they force the athletes to retire. The same is true for conflicts in the community. Some pass without a trace and make the community more friendly and united. Others, if not resolved, degenerate into drama and cause irreparable damage to the community, dividing and turning people against each other.

Conflict situations in the community cannot be avoided. There will always be people who are dissatisfied with something about the current situation, those who think differently than the majority. There is nothing wrong with that. Moreover, a natural attribute of a successful community is people whose purpose of participation is to harm the community. You shouldn’t be afraid of conflicts, but you definitely need to know how to deal with them.

## Conflicts in online communities

All people are very different. Each of us possesses a unique set of characteristics, needs and perspectives on things around us which inevitably introduce a conflicting element into our interactions. In well-functioning communities, users do not avoid conflicts; instead, they acquire the skills to address and resolve them. Making conflict constructive plays a pivotal role in enabling the community to accomplish its mission, adds creativity and improves the quality of decision-making.

Usually, conflicts emerge when there is disagreement on an issue that is very important to people, someone fails to meet someone else’s expectations, or deviates from social norms that others follow. In these circumstances, people tend to respond emotionally, which can result in counterproductive actions. Therefore, the fundamental aspect of conflict resolution lies in people’s capacity to manage their own emotions and opt for constructive methods to address conflicts. Poor conflict management reduces community performance as the users tend to focus all their energy on the conflict rather than using it to create something for themselves and the people around them. Conflicts cause stress, which can be so high that users might see only one way out — to leave the community.

Conflict is the gap between what we want to happen and what actually happens. When conflict happens, it generates a significant amount of energy. This energy can be harnessed in two ways: positively, when parties struggle for a shared goal, or negatively, when parties struggle against each other. Struggling against is a destructive process when parties must choose one of two scenarios: either win or lose. To resolve a conflict effectively we need to use the energy in a positive way.

## Destructive conflicts

Destructive conflicts are dangerous as they undermine both formal and informal relationships between users, adding psychological unease to communication and potentially driving users away from the community. Destructive conflicts have never helped resolve any underlying issues.

Mishandling conflicts may lead individuals to stop sharing thoughts and ideas due to a fear of criticism. In this case, the quality and creativity of decisions decrease dramatically. Destructive conflicts may also contribute to an increase of abusive and violent behavior in the community.

Destructive conflicts are also referred to as “relational conflicts”. Relational conflicts are situations when users feel offended, angry and blame each other for the presence of a problem.

When users use negative language to describe a conflict, they often refer to a destructive conflict. In this case, people either try to avoid the conflict or, when avoidance is not an option, they respond aggressively, so called “fight-or-flight” mode.

## Constructive conflicts

Constructive conflicts emerge in communities in the form of disagreement about ideas, values or opinions. Many communities are purposefully designed to foster constructive conflicts. In these environments, users engage in open and respectful discussions to address disagreements and gain insight into each other’s perspectives. Conflicts become beneficial.

Constructive conflicts are also referred to as “task conflicts”. Task conflicts are situations where users come together to solve a problem that is caused by their natural differences. Every user in the group concentrates on solving the problem instead of blaming others.

## Keeping conflicts constructive

Any conflict consists of two parts: an event that has actually happened (fact) and someone’s interpretation of what has happened (fiction). The frustration does not come from the events themselves but rather from the stories people tell themselves about these events. The interpretations, often rooted in emotion, may not necessarily align with the facts.

In order to keep the conflict constructive, it is vital that the narrative people tell themselves leads them to empathy towards the other person. To accomplish that it is critical that one can control their emotions and before losing their composure, take a step back, set emotions aside, and seek to deeply understand the other person’s perspective and what outcome they both want. This is what makes the difference between escalating a situation or contributing to a constructive resolution of the conflict.

It is important that users of your community know all this and know how to control their emotions. Your task is to educate them either in advance by storytelling or when a difficult situation arises by leading the conflict resolution process.

## Drama in online communities

Disagreements between a few users, if not handled correctly, can easily turn into an intense emotional conflict between almost all active users of a community. If this happens we say that a conflict is turning into a drama.

There is a book that helped me understand the essence of the drama and resolve a few of them on international Stack Overflows over the years. The book is called “Conflict Without Casualties” by Nate Regier. Here is the main idea of the approach proposed in the book in a nutshell.

Drama comes from an improper use of the energy that a conflict generates, when users consciously or unconsciously start turning against themselves or others. Any drama has two primary forces. It is either self-justification of one’s destructive behavior or trying to prove one of four following myths: one can make the other feel good or bad, the other can make one feel good or bad. (It is important to understand that while people can affect how others can potentially feel about a situation, the others have a choice to make those feelings a reality or simply let them pass.)

Dr. Stephen Karpman developed the idea that drama is a collective activity that is based on three defined behavior roles: persecutor, victim and rescuer. These roles form the drama triangle. Each role complements the others: victim needs persecutor or rescuer, without persecutor and their victim there is no one for the rescuer to save. When we think of what to do or to say in a particular situation, we choose from behavioral patterns that we experienced in the past. Thus, sometimes when we try to help other users, we unconsciously take the missing role and become part of the issue instead of being a solution. Please note that the role we take is almost fully determined by the others’ behaviors.

- *Persecutor*. When a user criticizes, accuses or insults others, they probably play the role of a persecutor. Persecutors believe they are right and justified, and manipulate others into accepting blame or responsibility. They use the “one can make the other feel bad” myth to justify their behavior. They are convinced that they can do anything they want as long as the goal is achieved. Users who play the persecutor role never take responsibility for their actions and blame others if something goes wrong.

- *Victim*. Whatever happens, users who play the victim role somehow think about themselves as being the root of all problems. These users avoid all conflict situations by agreeing, making concessions, and admitting to any accusations. Victims look like they are repressed or offended most of the time. They are constantly looking for sympathy from others. This behavior triggers persecutors' anger and reinforces victims' belief that "someone can make them feel bad."
- *Rescuer*. Users who are intruding into others' affairs with no reason, giving unsolicited advice, suggesting what to do and the like, usually play the rescuer role. The over self-confidence and honest belief that "they can make others feel good" are the primary driving forces for the justification of their behavior. By trying to be helpful, rescuers aim to build positive relationships with fellow users. The problem is that they fear not being needed, as a result they rarely want others to grow personally.

The book suggests that there is a way to resolve a drama. It is compassionate responsibility. Compassion is the ability to understand and accept others' feelings. Compassion combines caring for others, being empathic and open to different views and values. One has responsibility when they are given the right to do what they say and deliver what they promised. A responsible user fully recognises the consequences of all their actions and behavior, both good and bad.

Responsibility with no compassion leads to rudeness. Compassion with no responsibility leads to inaction.

Resolution starts with openness at all times. Openness is when we approach both our own and others' emotions with an open and non-judgmental attitude, regardless of our personal views on those emotions. Openness helps us build close connections with others. It creates trust that is needed for users to start sharing their feelings and thoughts about the problem. At this step we listen to others, validate their emotions, express our empathy and share what we think about the problem. This step makes users think clearly.

When a list of creative solutions to the problem is created, everyone involved needs to share with others what actions they would like to take to resolve the conflict without giving advice to others on what they should do. Then we need to make users responsible for their commitments by making it clear that everyone involved need to deliver what they have promised.

At this point everyone should be in alignment. If not, you need to go to openness again and repeat the cycle.

# 8. Community metrics

Stack Overflow provides access to almost its full database through the SQL console at data.stackexchange.com (Stack Exchange Data Explorer, or SEDE for short). A similar service, but with access to full data in real-time, is available to company employees. We used our local SEDE to evaluate the effectiveness of ongoing initiatives and conduct research. Over time, we have accumulated a large number of metrics (in fact, SQL queries that calculated some numbers on certain data points). There were many metrics and each of them told its own story. Moreover, seemingly related metrics could show completely different trends. For example, from 2017 to 2023, the number of questions asked and answers posted on Stack Overflow had decreased, but the number of monthly active users had increased during the similar period (at least, according to its definition). As a result, we found ourselves in a situation where, if one wanted, they could arbitrarily prove or disprove any hypothesis they wanted with the help of data.

A complete lack of metrics is rarely a problem today. The most critical question for any modern project that uses metrics to evaluate initiatives is which metrics to use and how to interpret them in situations where there are many metrics and some are showing growth while others showing a decline, or when the metrics show no change but clearly there are enormous changes underway.

As the company grew in 2020, the problem of interpreting metrics became critical and I began working on creating a community health metrics system that could be used by all community managers and product managers working on Stack Overflow sites. Community health metrics is a fairly large topic that goes beyond the scope of this book. Therefore, in this chapter we will look at only the most important conclusions that can give you a sense of what the community health metrics might look like.

Let's start from the beginning. What are the metrics?

## Metrics are the measurements that reduce uncertainty

A metric is a quantitative measurement of something in some specific context. The purpose of any metric is to help decision-makers make better decisions. The difficulty of making a decision is that usually there is some uncertainty associated with the decision. We use metrics to reduce that uncertainty. Metrics do not need to eliminate uncertainty, but they must add some new information about a phenomenon or data that simplifies understanding of the phenomenon.

## Any metric has associated value and costs

The value of a metric is the difference between two decisions: one made with a metric and the other made without it. If it is equal to zero, then there is no point in having the metric, since the metric does not improve (or does not participate at all in) the decision.

The costs of a metric are the costs of thinking through, implementing, and using the metric. Although the cost of metrics in online communities is kept to a minimum, metrics require interpretation and attention by a decision maker or an analyst. Any new metric you add creates yet another data point and one more thing to look at. If metrics are not created wisely they add more noise to the data than there was prior to adding them and overwhelm the decision making person, reducing the value of having them in the first place.

If a metric has no value, there is no reason to have it.

We can represent uncertainty as a set of possible outcomes of an event where each outcome has its own probability of happening and an associated value, positive or negative. When one thinks of what metrics they need to create the starting without to look for areas where there is high uncertainty and high cost of being wrong. Those situations are areas where the most valuable metrics exist.

## There are three types of metrics

In a large company that develops its own software to run their community, there will most likely be three dedicated groups of people who will need their own metrics.

### Business metrics

The main consumers of these metrics are department directors. They are usually interested in community dynamics that may somehow affect the company's revenue or expenses and the forecast of these metrics for the future periods.

To get some sense of how these metrics might look, let's examine the value a community creates (the value that could be converted into revenue, I mean). There are two primary types of value that can be created by an online community.

- *User actions*. We expect users to perform certain actions that have value in themselves. For example, buying a product, signing a petition, recommending your service to friends.

- *User generated content*. By interacting on your platform, users leave a digital footprint, the content. This content usually has value in itself and can generate revenue.

To measure business success from the monetary perspective we can use the difference between the value that the community creates minus the costs of creating and maintaining the community itself.

Also, business people usually want to see some high-level metrics on which the value the community creates depends, like the number of pageviews or the number of posts created.

*Note: When calculating the value created by the community, make adjustments for community inertia: even if you stop investing into the community it will continue to create the value for quite some time. The larger the community, the greater the inertia and the longer the community can create value with almost zero costs.*

Success metrics of specific initiatives and overall community health metrics

The consumers of these metrics are product managers and community managers (and probably most of the readers of this book). Typically, they want data on the health of the community and how the specific initiatives they are pursuing impact it.

The first step in this is to define what health means for your community. In general, we say that a system is healthy (or an initiative is successful) if it efficiently does what it has been designed for. In other words, with health metrics we measure efficiency. To start defining health metrics for a system or an initiative we need to ask ourselves the following questions.

What does health or success mean for the system? If everything goes well, how will the system look?

The answer to these questions is what needs to be measured. The resulting metric is the health metric for your system or an initiative.

For example, let’s look at an abstract community as a whole and define its health metric. A community is a group of people united by a common goal or interest. For a community to exist, people must participate in it. At the same time, constant churn of users is an integral attribute of any online community. If it happens that more users leave the community than join, then the community will sooner or later cease to exist. So that we can conclude that a community is healthy if it has more people joining it than people leaving it.

### Success metrics of the current community goals

The main consumers of these metrics are the users of a community. It is critical for users to know where the community is in achieving goals of the specific community initiatives that users are working on and the end mission as a result. The main purpose of this set of metrics is to provide feedback to users on their actions.

When users have access to data using which they can create metrics, then users are more likely to create initiatives that they believe will help the community and track progress, doing everything on their own. In other words, metrics intended for community users should be publicly available. The more metrics and data you can give users access to, the more proactive they will be.

## Measure independent subsystems separately

An online community is a complex system consisting of several independent subsystems. In order for measurements to be interpretable and have practical meaning, it is necessary to measure each independent subsystem separately. Before creating community metrics, especially product related metrics, we need to identify all independent high-level subsystems. To figure out which parts of a particular setup of a community are independent and which are dependent, we need to try to replace each part one by one with their analogues. If it is possible to replace a part without significant changes to other parts it is independent and must have its own dedicated set of metrics.

For example, Stack Overflow consists of three main independent parts:

- Software
- Community
- Contents

Here is why those are independent subsystems. One can migrate a community (the users) and content to a completely new platform without impacting the project much. One can work on new content with an existing community or create a new community around existing content. At the same time, many different communities with different content can use the exact same software. Those three parts are interconnected and form a single whole, though they are not tightly connected to one another to form a system with clear feedback loops.

## Do not be overwhelmed by looking for meaning of Return on Investment

There is one more important topic to discuss which is Return on Investment (ROI) and how to calculate it for an online community. There are a lot of articles that contain thoughts explaining that is something that any project must have but none of the experts tells us how exactly to calculate ROI. The reason is that ROI can only be calculated if community activities and associated measurements have a link to financial metrics either directly or indirectly and it is rarely the case for online communities.

When it is impossible to calculate ROI, you can try to calculate ROnI, the Risk Of not Investing. In an applied sense, you can find all the costs that a company would incur to achieve the same results as you get with the community now. For example, we can calculate how much it would cost someone to create and moderate a knowledge base about programming of the same size as Stack Overflow has and then compare it with the expenses of having the community management team.

# 9. Software

In today’s world, developing your own social software is a special case, which I deliberately leave outside the scope of this book. At the same time, I would like to emphasize two very important points.

## In online communities, software plays a secondary role

The most important thing to know about software in the context of online communities is that the success of an online community does not depend on the number of features the software on which the community is run has. The primary reason for that is that the value that people get from participating in an online community usually lies outside the physical world. These are a sense of belonging, feeling of unity, the pleasure of being able to share ideas with same minded people who appreciate it, etceteras. To grow a community, one needs to maximize that value, which has very little connection with software per se. At the same time, without proper software it will be hard to get the job done for both the community and the community manager.

## Social software is about efficient groups

Social software is all about efficiency of human groups and usually has three essential goals:

1. *Providing tools to achieve the mission*, a set of possible actions users can perform on a platform. Without tools, users will not be able to do what they are invited to do on the site.
2. *Providing tools for coordinating the common effort*. The more people involved in activities in a community, the more resources needed to be spent on coordinating the group so that it does not turn into chaos. This requires providing users with tools to coordinate collective activities and self-govern the group.
3. *Providing tools for creating and maintaining groups*. Tools aimed at enabling individuals to connect, communicate, and engage with others and foster their relationships.

Everything that does not fall into the three categories above is good to have but not essential for a community to operate and does not contribute to the success of the community *per se*.

## What to look for when choosing a software platform run a community on

It is difficult to predict what tools a community might need in the future, but there are the most important aspects that one should pay attention to before choosing a platform to launch a community on. Here is my list.

### The three places

Sociologists distinguish three distinct social environments where people operate. The first place is home, where we spend most of our free time. The second place is the workplace, the place where we go to do something. The third place is the place of socialization. In the context of online communities, the first place is the main site, the second place is a site for meta discussions, the third place is a site for off-topic conversations. Even though the end goal of community building is to grow the first place, our main site, without the second and third places, the growth of the first place is limited. Be sure to provide your users with all three places at some point of the community development.

*Note: Those places might be parts of the same platform and in the same format (for example, separate sections on a forum), or on different platforms in different formats.*

## The number of customizable parameters

One good way to engage and retain users in a community is to give people the opportunity to adjust the environment in which they interact with one another. The more settings to meet the specific needs of users a platform has (and the more often you change them), the greater the engagement of the users will be.

### The set of content moderation tools

Content moderation is all kinds of actions whose goal is to keep the content clean and relevant. Any user should be able to take part in content moderation to some extent, as long as users want to volunteer their time and you trust them. Usually, content moderation does not require any special knowledge besides good writing skills, understanding of community standards and basic common sense.

The higher the number of users who have access to content moderation and the more gradual the access to moderation tools is, the better.

*Note: Let me emphasize that moderating content is not the same as moderating relationships between users. Moderating relationships is about dealing with interpersonal misunderstandings or even conflicts. It requires special skills such as empathy, active listening, conflict resolution, etc, and implies the need to make decisions on very subjective topics. This type of moderation should be restricted only to a small group of selected people.*

### The content health analytics

In large communities it is impossible to view all content posted on the platform due to its volume. In order to have an objective estimate of the quality of the content, the platform should provide some tools that can help evaluate the quality of the content on the platform.

### The ability to create subcommunities

Online communities grow by splitting a larger community into smaller subcommunities and growing each of them independently. If you are planning to have a large community you need to find a platform that provides some ways to create isolated community spaces based on an existing group and connect new sub groups into a single whole.

Any subcommunity is a community in itself and, as a result, the dedicated subcommunity space should provide users the ability to customize it. The more customization options for dedicated spaces the platform provides, the more effective the growth of subcommunities will be.

### The number of community roles

People start voluntarily performing some activities if doing so brings them pleasure in itself. Separating an activity into a dedicated official role with a special social status is a way to reward volunteers for their contribution to the community. The more different dedicated roles you can have on your platform, the more active users will be in helping you grow your community.

### The type and the number of digital awards for volunteers

In any community, there are always short-term or one-time activities that do not require a dedicated volunteering role. Like helping to facilitate a meetup, speaking at a conference on behalf of the community or participating in a seasonal initiative. At the same time, any such contributions should be rewarded in some way. One of the most efficient ways to do that is to reward volunteers with some special visible digital awards dedicated to specific initiatives that make the participating volunteers distinguishable from the rest of the community. The more customizable digital awards for users the platform provides, the better.

### Automatic backups and data dumps

Before launching a community on the chosen platform, be sure that you can download all the data from the platform you chose and re-create the community based on the data dump. Technical glitches happen all the time and you risk losing everything in an instant if you don’t have a backup. Additionally, in some rare cases you may need to change the platform the community operates on. It is possible only if you can download the full dump of your data.

## Accessing community data using SQL queries

There is no platform that has a user interface for everything you would want to know about your community. If the platform provides a way to run SQL queries and download the results for subsequent analysis, this will make your life much easier and improve the quality of your decisions.

# 10. The rise and fall of International Stack Overflows

## If you do it well, it will be well: Stack Overflow in Russian is one of the fastest growing communities on the network

There is a big difference between playing football in the yard with friends and professional sports. Likewise, creating a chat or Facebook group for friends to organize a birthday party is not the same as building and managing Stack Overflow, which is visited by hundreds of millions of people every month. To play in the Premier League, you need not only to constantly practice, but also to know what to do, how to do it and what you should not do at all.

The main activity metric for Q&A sites is the number of questions asked per day. We launched HashCode at the end of 2010 and by the end of 2011 users asked about 35 questions a day on the site. Due to our erroneous (marketing) approach and misguided priorities (on software), this metric did not grow until mid 2015, when we migrated to the Stack Overflow platform. Over the following three years of managing the Stack Overflow in Russian, I posted about 200 questions and 450 answers on Meta, held a dozen offline meetups in different cities in Russia and abroad, recorded a dozen episodes of the community webcast and spent 0 cents on advertising. 35 questions per day in March 2015 on HashCode turned into 250 questions per day in May 2018 on Stack Overflow in Russian. Over three years of active development, the community grew sevenfold and continued to grow. Stack Overflow in Russian was one of the fastest growing sites in the Stack Exchange network.

If someone asked me today what the secret of the success of Stack Overflow in Russian is, I would answer that there is no secret, but there are key activities that one should focus on. If you create the right foundation for your community, everything else will follow.

Here's the foundation of Stack Overflow in Russian.

- *Kindness*. Without treating each other with kindness and respect, nothing else matters. We constantly promoted the idea of being kind and respectful to each other no matter the circumstances and asked users to assume the good intentions of others no matter what happened.
- *Socialization*. All processes in the community will proceed faster and more efficiently, and users will be more active if they have good friendships with each other and with you. In truth, the social capital we talked about earlier is a characteristic of the entire group, not just one individual. The higher the social capital of your community, the more effective the community is.
- *Communication*. One of the most critical applied skills of a community manager is the ability to engage with a compelling story, showing users the desired future that the community can achieve if we all work together.

Let me emphasize that kindness, socialization and communication are not attributes that can be achieved, they are processes that must take place in a community on an ongoing basis.

## The end of growth of international Stack Overflows

From the moment I joined the company until mid-2017, each international community had its own dedicated community manager. In the second half of 2017, two international community managers left the team, and in November of the same year, the company laid off about 20% of its employees. After that I first started helping other international Stack Overflows whenever possible, and in 2018 I officially started managing all international Stack Overflow communities.

The essence of the approach to growing multiple communities is no different from growing a single community, but the method is slightly different. Basically, one needs to take a step back and "reduce the focus." Here is what I mean. When you manage a group of communities, you need to work even more with the most active volunteers and less with the users of each of the communities. In our team we called this approach "moderator management" as opposed to the standard "community management". I began to move in this direction as well. Together with users of the communities, we created manifestos for international communities, began to collect a list of initiatives held in each of the communities to share the experience, hold offline meetups and online contests, and much more. My colleagues and I truly believed in the success of international communities. We expected that our four international sites, under certain circumstances, could grow to account for 30% of all activity on Stack Overflow in English together.

Unfortunately, our plans to grow international communities were not destined to come true. A period of great change began in the company. During 2019, almost the entire top management team of the company and many heads of departments were replaced. This was accompanied by great internal and external upheavals. It was impossible to do anything new or non-standard in such an unstable situation. In the first half of 2020, after cutting another 15% of the company workforce (and irreparable losses in the community management team), work on the international sites officially stopped, and my focus became the moderators of the Stack Exchange network.

## Why international sites haven’t taken off fast

### How we calculated the volume and size of language niches

The number of questions asked per day is one of the main activity metrics for Stack Exchange communities. Everything else depends on the number of questions in a community. Without questions there will be no answers, without questions and answers there will be nothing to comment on and nothing to vote for, *etc.* Here are two facts. There was a 98% correlation between the number of questions on the site and incoming traffic, and over 95% of traffic on Stack Overflow comes from search engines. Therefore, we guessed the expected maximum number of questions on Stack Overflow in a specific language niche through the number of questions per day on Stack Overflow in English and the proportion of traffic on Wikipedia in that language.

The idea was that we believed that the number of people who use Wikipedia in a particular language niche should be proportional to the use of Stack Overflow in the same language. If this is true, then the following equation is true as well.

*QPDSOen TraffWikiEn = QPDSOLang TraffWikiLang*

Where:

- *QPDSOen* is Questions per day on Stack Overflow in English.
- *TraffWikiEn* is Traffic on Wikipedia in English.
- *QPDSOLang* is Questions per day on Stack Overflow in a specific language.
- *TraffWikiLang* is Traffic on Wikipedia in a specific language.

From there:
*QPDSOLang = (TraffWikiLang / TraffWikiEn) * QPDSOen*
At the time of the active development of international sites, about 8 thousand questions were asked per day on Stack Overflow in English, and traffic on Wikipedia in English was about 50% of the total traffic of the project. From here we get the following results for the expected maximum QPD on our international sites.

| Community | Traffic ratio | Expected QPD |
|---|---|---|
| In Japanese | 2 * 0.072 | 1152 |
| In Spanish | 2 * 0.069 | 1104 |
| In Russian | 2 * 0.053 | 848 |
| In Portuguese | 2 * 0.022 | 352 |

By 2017, when we suspended work on international sites, they were all just at the beginning of their growth curve.

Why didn’t international sites grow as fast as we expected?
There were no silver bullets that could help us, but there were the most critical issues that have clearly hampered the growth of international sites.
**Lack of a clear powerful mission and an antagonist for international sites**

By the time international sites launched, the Internet was very different from when Stack Overflow in English was launched. In particular, Stack Overflow in English had already replaced most forums in search results for many new technologies and no one even remembered about Experts Exchange. Although all international Stack Overflows have a similar mission, it was quite difficult for users of international sites to feel the problem the way early users of Stack Overflow in English felt it. International Stack Overflows were launched for people who have a language barrier and those who would like to create a knowledge base in their native language. These are very worthy goals, but their driving force, which makes people join the group and be active, is noticeably weaker than the intolerance of Experts Exchange and cluttered search results.

*Note: In my view, the current Stack Overflow in English has lost its main driving force and is in approximately the same state as international sites. It seems that many people contribute to Stack Overflow in English largely out of inertia.*

**Competition between Stack Overflow in English and international Stack Overflows**

Each international Stack Overflow was launched as a completely independent community and was not connected in any way with other international sites and Stack Overflow in English as well. Some studies say that 75% of all clicks in search results come to the first three links. Our sites competed with each other for the top positions in search results and those clicks. Stack Overflow in English with 10 million questions often won the competition from international Stack Overflows and thereby taking all search traffic, consequently cannibalizing the international communities. If the knowledge bases of all Stack Overflows were linked, we could manage traffic more efficiently internally, redirecting it to the correct language site when necessary.

**Lack of platform development**

The site engine of Stack Overflow was developed based on specific problems arising when growing the English-speaking community. After the launch, international sites had their own unique needs in the form of tools for interface localization, help center synchronization, knowledge base integration, etcetera. Most of these requests were not implemented due to lack of resources. Therefore, on international sites, a lot was done manually, or not done at all. As I wrote above, software as such will not make the community successful, but if you have good software, it will greatly simplify life for the community. In addition, users' sense of belonging and sense of ownership arise when a company and users work together to solve emerging problems. This cut off the growth of international sites as well.

# Part 3

## The fall of The Star

I was very lucky. I had the opportunity to go through a journey with Stack Overflow that was almost nine years long, plus another four years of working on HashCode on my own. I had the opportunity to look from the inside at the most important stages of development of the company and the community. First, I went through the period of launching and early growth of HashCode, then the rapid growth of Stack Overflow in Russian, followed by the work on international sites and then on Stack Overflow in English, when the community reached a plateau in the number of questions and answers. I saw how we slowed down community development as the company tried to find its way to profitability. Then the period of growth gave way to a period of serious changes in management and the subsequent acquisition of the company. Then I left the company when it became, at least in the context of processes within the company, a corporation.

In my opinion, the growth of the community was most tragically affected by changes in the management and the subsequent transformation of the company. The company, whose main focus has always been the community, began to move towards a SaaS-oriented company with a line of software products for knowledge management: Stack Overflow for Teams and Stack Overflow Enterprise. The changes affected all parts of how the company was functioning. At some point, all the employees who made decisions, including product managers in specific teams, were experts at developing SaaS solutions, and there were only literally about a dozen people left with real experience in community development for the entire company.

The way we developed software has also changed. When the project was just launched, the original Stack Overflow Q&A engine was created to solve a specific social problem. When working on the engine, the developers proceeded from the problems that arose during the use of the software on a large scale. Most decisions about the architecture of the engine and the set of available features were made based on user feedback and public discussions with the community. When it was time, we adapted what was rightfully the best Q&A engine in the world for use within companies, resulting in the products Stack Overflow for Teams and Stack Overflow Enterprise. In the new paradigm, everything was turned upside down. Stack Overflow for Teams and Stack Overflow Enterprise have developed by leaps and bounds, bypassing the public site. While on the public website, since the acquisition of the company, I cannot name any innovations that have had any positive impact on the growth of the community. In other words, the company's SaaS products were actively evolving, while the problems of the public platform were not addressed at all.

There is an axiom of community management that says that any social system has two states: growth and death. The Stack Overflow community has been stagnant for a long time. As a result, starting from 2020, the decrease in the number of questions asked per year has been about 15% and by 2023, there were only 45% questions asked on the site compared to the number of questions asked in 2016, In 2023, on Stack Overflow, which every developer in the world knows about, with a knowledge base of more than 20 million questions, there were

16% fewer questions asked than were asked on Stack Overflow in 2011, when few people knew about the community.

These and other figures clearly indicate that something wrong has been happening in the community. In this part, we’ll look at two of the most critical problems that I think modern Stack Overflow suffers: how decisions are made within the company and how the company interacts with the community.

# 1. Product and community management

There are no good or bad approaches in business, but there are suitable and unsuitable ones for solving a specific applied problem. Like there are nails and bolts. It seems that at some point something went wrong at Stack Overflow (the company) and we started hammering nails with electric screwdrivers. As a result, at the end of the day we did not get the nails driven in and had screwdrivers that are broken.

## It is extremely difficult to grow an online community with a top-down approach to management

In 2019, almost the entire top management team of the company changed. Until this point, the company was run by Joel Spolsky, one of the company’s founders. Joel preached the “servant leadership” approach in management, the applied meaning of which in our context boils down to the following (in my interpretation, of course). The leaders of a company choose the direction of development and communicate it all the way down to individual contributors. Then individual contributors need to understand and internalize the chosen direction. Further, since individual contributors have more information about the state of affairs on the ground, they choose the most effective way to achieve the goal set by the leaders and implement the chosen approach. Managers, in turn, help individual contributors do what they do by removing obstacles on their way. Simple and effective. As the company grew, the approach to organizing teams and how resource allocation changed, but the original idea remained the same: we always followed the idea of servant leadership.

It is worth noting that servant leadership is an ideal management approach for community management teams because in this case the internal processes of a company match are the same as we usually manage communities: The management of the company makes sure that community managers succeed in their endeavors. At the same time, community managers do their best to support volunteers and make sure they have what they need. So, if users need anything, it is the top managers' goal to provide them with it. In our case, when we were managing Stack Overflow communities, first, we communicated to the community the goals that the company was pursuing at a particular point in time, and then we found out the needs of volunteers and their general feedback about the planned activities. If the community management team could not help the users with their needs on its own, we communicated the needs up to the product development team, and even higher, if the product development team could not help us.

Like all approaches, servant leadership has its pros and cons, but in general this approach worked for us and community managers acted as a bridge, representing the interests of the company within the community, and the interests of the community within the company. With the change of the management team in 2019, the approach also changed. Servant leadership has been replaced by a standard managerial approach, which I personally call "I'm the boss, you're a fool." The essence of this approach was that the company's top management chose not only what we would do but also how without any room for discussion. The task of middle management was simplified to communicate specific tasks to individual contributors. The task of individual contributors was to carry out the assigned task the way it was defined. There was complete one-way communication from top to bottom. This style of managing the company affected the way the community was managed. Community managers kept representing the interests of the company within the community, but in the opposite direction, the bridge led to an abyss. Most likely, the “I’m the boss, you’re the fool” approach may work in some industries under certain conditions, but definitely not in an IT company where incredibly smart people work. Especially not in a company whose task is to help the community be effective, because the whole point of growing a community is to help the users to achieve the mission which assumes two-way communication by default.

The “I’m the boss, you’re the fool” approach was felt almost at every single turn. For example, at some point we needed to write a new help center page. This is a typical task for a junior community manager, no big deal. When the task arose I didn’t even pay any attention to it. The page was written by someone on the team and I was asked to help translate it for international sites. When I read it, I was surprised that it was written very poorly: there was no logic in the narrative, there was repetition of ideas and expression of the author’s personal views. While translating I corrected the original text, leaving the general idea the same, and sent the new version to my colleagues so that they could review it and make changes based on the review. Unfortunately, some of the managers forbade making any edits to the text. It turned out that the page was written by a top-level employee and making changes to their work did not correspond to the company corporate culture. Why it was impolite to improve someone’s work as well as why a top-level employee spent even a minute of their time on doing junior community management work is still a big question to me.

The company did not become like this overnight, the changes have been happening gradually. With the new top management, the middle management also changed. Then people who were accustomed to the servant leadership approach, which implied autonomy in decision-making and critical assessment of approaches to achieving goals, regardless of who proposed them, began to leave the company. Not everyone left, the company still had a lot of talented people, but the number of folks who asked “uncomfortable questions that did not correspond to the new corporate culture” became fewer and fewer.

Let me reiterate. The main task of a company that manages a community is to make the community work effectively. To do this, the company has community managers who monitor community dynamics and communicate with most active users, finding out their problems and, based on this, recommending developers to make software changes, if necessary. When decisions are made in a closed circle by top managers and get sent down for implementation, it becomes impossible to receive any information from the outside. Having abandoned servant leadership, the management has built a series of blank walls around themselves, and the company. On top of that, I personally do not see how top-level employees, middle management or even product managers who never talk to users and do not have domain expertise can make even an educated guess about what the community needs. This approach is doomed to fail.

*Note: It's worth noting that Stack Overflow, the company, actually has two types of businesses: public communities and SaaS products. I am sharing my experience in working in the department that was involved in the development of public communities and I have very little idea of what kind of experience my colleagues had in the development of SaaS products division. It is quite possible that for them the chosen management approach seemed more rational than to me.*

## The product team and community management team should have a shared roadmap

Changes in the company's management in the first half of 2019 affected the structure of the teams. In particular, the team developing the public website got their own director. This was a product manager who managed the resources dedicated to the team. The community management team also got a dedicated director. In the beginning, everything seemed to be going well. Each team was engaged in planning, setting priorities, drawing up a roadmap for the upcoming year and a detailed work plan for the next quarter. Suddenly problems began to appear because each team had its own roadmap, the teams had constant conflicts of interest.

The community management team needed developers to implement features that were critically needed by the volunteers and the team itself, but no one allocated these resources, since the development team planned all its resources in its roadmap. At the same time, the product development team approached development in a way that was counterintuitive to the community. People somehow made their own decisions about what needed to be developed, developed what was planned, and delivered the result to users who had completely different needs and priorities. In other words, the product development team critically lacked domain expertise, but community managers were busy implementing their roadmap.

The conflict of interest grew and it became obvious that the current scheme was not working and it was necessary to unite the teams somehow. The question became how to do this. In my view, the choice of who should help whom is quite simple.

- The product development team should support the community management team, that is, develop software according to the community management team's roadmap, if the company's end product is the community itself.
- The community management team should support the product development team, that is, help to prioritize, consult and ensure adoption of the software according to the product development team's roadmap if the company's end product is software.

The reasons behind the approach where the company's product is a community, the ultimate goal and the main value is the developed community, the software itself plays a secondary role. If your main product is software, then most likely you do not have any applied community that works on its own, only a community of your engine's customers, the community that does not generate revenue per se.

Stack Overflow, the company, definitely has a community and that's where the company's greatest value lies. Unfortunately, when we united the teams, the community management team began to support the product development team. Although this was much more efficient than working separately, it was still a mismatch because the company continued to implement initiatives that had been planned internally, rather than what users heeded and had asked for. To understand why it was the case, let's look at the structure of the resulting teams.

## The development of a community platform should be managed by a person with domain expertise

In June 2021, Prosus bought Stack Overflow. The company acquired the resources to grow and we began actively hiring people to develop all the products the company had, including the public communities. Up to this point, the product development team who worked on the public platform consisted of only a few people, but now the team had grown to several sub–teams. Each sub–team consisted of people with different skills: developers, designers, product managers and community managers. Within the company, we called these sub teams "cross-functional" teams. The leader in all groups had to be the product manager, who made all decisions on the initiatives that the team worked on unanimously (adjusted for the fact that many of the tasks that we had to work on came to us from the top). This method of decision making, as I see it, is another important reason why the company failed to develop the public platform.

Let's get back to the question from the previous section. What was the end product of Stack Overflow? Were we developing a SaaS solution? Or were we developing a community? In my view, the only correct answer in the case of Stack Overflow (the public platform) is that we were working on a community. In this case, only the community management team had context regarding the current needs of users and the overall situation in the community. Honestly, community managers were hired to do those jobs in the first place. When community managers were excluded from decision making, no one brought that context to the product development team. As a result, the product development team worked on initiatives that didn’t gain any traction and didn’t help the community to grow. Also, guess what will happen if you don’t address users’ current needs for tool A, and instead you offer them tool B, which has nothing to do with A, and then try to convince users that what they really need is actually B, not A? Yes, correct, all you will achieve is aggressive behavior and a ton of negativity towards you and the company.

This was exactly what we experienced: the constant negativity of users on Meta and the low end benefit from the features being created. I personally can’t think of a single big feature we developed between 2020 and 2023 that had any significant positive impact on the community. Don’t get me wrong. All my colleagues were incredibly talented, and it was a pleasure to work with them. The problem was precisely in the approach itself, when the product managers were required to make decisions in areas about which they had no experience at all. Once again, if you can’t play with the big dogs you’d better stay on the porch.

In general, Product Manager and Project Manager do not need to be the same person. I know many successful teams where the product manager is engaged in market research and helps the team with the business component of product development, while they are not the same person who makes the final decision. I believe that in successful teams, the decision should be made by the person who usually has domain expertise. In companies where the main product is community, this will most likely be a person from the community management team who has solid project management skills.

## Community managers must peer review each other’s work

When I started working at Stack Overflow, it was standard practice throughout the company to review the ideas and work of colleagues. Typically, everyone formulated their proposals in the form of “Request for Comments” documents (RFC for short). After a discussion in RFC format, where anyone could leave a comment and receive feedback from the author of the document, an initial specification was written, which, again, was reviewed by all interested parties. Even all the announcement posts for Meta were reviewed by the team. I really liked this approach because it allowed us all to learn from each other and reduce the number of mistakes we made.

After another series of layoffs, in 2020 the community management team consisted of only five people, one of whom was our manager. There was much more work than we could do. Therefore, we simply selected the most important tasks, distributed them among the team members and tried to do everything as well and quickly as possible. In that reality, we had to abandon RFC and peer review. If anyone had questions or requests for colleagues, we discussed them on the weekly team calls. All the folks on the team were highly qualified and understood the nuances of our community. Basically, the approach worked.

By the second half of 2021, after Stack Overflow was acquired by Prosus, the team had grown in size to 13 people. While scaling the team, we made a grave mistake by continuing to work on initiatives individually without peer reviews, and, unfortunately, this culture became entrenched in the team. In subsequent years, until I left the company, all community managers worked on initiatives on their own. To understand the depth of the problem, imagine for a moment what a world would look like in which developers did not do peer reviews of each other’s commits? That’s right, it would be a world in which we had buggy software, and the speed of the personal development of engineers would be reduced to a minimum. Although I worked on the community management team and not a software development team, the result of the lack of peer-review was the same.

When I worked as a developer at Motorola, the company constantly hired interns who had to be trained and adapted to work for a big company. Therefore, for a group of two or three interns, one senior developer was allocated. The senior set tasks for the interns and reviewed everything they did. In this way, interns learned development processes, how to give and receive feedback from colleagues, as well as programming. Because of this process the result of interns’ work could be used in the real product. Most of the colleagues who joined in 2021 were new to community management, and some had never even participated in Stack Exchange communities before. In my view, they critically needed the same care we gave interns at Motorola, with a mandatory review of the work done by one of the senior community managers, without which none of their work could go into the final product.

Stack Overflow had a process called “onboarding buddy” which meant that everyone new to the company had a dedicated colleague they could approach with questions if they had any. But here’s an axiom: The difference between a senior and a junior is that the senior understands perfectly well what they don’t know, while the junior thinks that they know almost everything. Additionally, new employees may be embarrassed to ask questions. In other words, a junior will rarely approach anyone with a question and will simply do their work to the best of their ability. Now, add to all this the absence of peer review and we get what we have on Stack Overflow today, a serious decrease of user engagement in the community.

Each member of your team has a unique experience. Peer-review is a perfect way to share this experience within the team and as a result improve the final product.

## Communities do not grow by coincidence, to grow a community one needs a plan

When I started working on HashCode, I had minimal funds for the software development and community building. The entire starting capital was very modest savings, which I was able to set aside by working as a programmer for a few years. At the same time, in some incredible way I managed to create one of the most popular communities of software developers in our language niche. Now in front of me there was Stack Overflow, where since 2020 user activity has been constantly decreasing, while the company had more resources than ever before. I was constantly tormented by the question: How can this be possible?

Let’s try to answer this question.

As I see it, the most striking difference in the approach I followed to grow HashCode and the one that the community team used on Stack Overflow in recent years was that while we were working on Stack Overflow, we did a lot of things, but none of the initiatives were aimed at growing the community per se. At the same time, when I was working on the HashCode community, I only did things if I expected them to increase activity on the site or the number of users in the community.

On international sites the situation was similar. My fellow community managers and I were solely focused on growing our communities. Before we started anything, we made plans and only then got down to business. The basis of planning was to assess how the possible end result of a particular initiative might affect the growth of the community and what the likelihood of the initiative being implemented successfully would be. To evaluate initiatives, we used a rule-based system through which we prioritized initiatives and filtered out anything that was impractical to work on. Here are the most important rules.

### Planning rule #1: Almost all activities a community manager does should be aimed at growing the community

Any social system has only two states. It either grows or dies. If you do not work on growing your community, it will begin to stagnate, then degrade and will eventually die one day. Running initiatives aimed at growth is the only way to keep a community healthy. Everything you do as a community manager should always be viewed through the lens of community growth.

Please, note that community growth activities and marketing are two different things. Marketing is more about attracting new users. Community growth activities are about working with volunteers within the community itself.

When planning your work, look at all you do through the lens of community growth and filter out everything that does not affect growth.

### Planning rule #2: Look for AND instead of choosing OR

I think everyone has at least once encountered a conflict of interest when it was necessary to achieve many goals at once, but they had time and resources for one goal only. The standard solution (the OR approach) in these situations is to weigh the alternatives and choose the best option.

There is a better way, actually, at least in the field of community management. A more effective solution (the AND approach) is to try to find an initiative that allows you to achieve results in several directions at the same time where the sum of many not-the-best results exceeds any of the individual alternatives.

When planning your work, choose the initiatives that give the greatest result in terms of the sum of progress towards many goals.

## Planning rule #3: Take on initiatives that you can actually deliver

From my personal experience of working on Stack Overflow, most initiatives that took much longer than they should have or did not reach the end user at all have one common characteristic. There were too many cooks in the kitchen when working on them. In other words, the successful completion of the initiative depended on too many people working together successfully.

A lot of delays in collaboration occur because different departments have different priorities for a given initiative. Programmers, designers, and community managers may think of the same task with different priorities. Reducing the number of people who work on an initiative, especially from different departments, is the easiest way to increase the likelihood that your work will reach the end user.

When planning your work, prioritize those initiatives that require minimal help from outside of your team to complete.

## Planning rule #4: Improve rather than create from scratch

Working on something new feels good. Our mind draws an optimistic forecast for the future success of the endeavor. But when working on something that already exists, our mind limits the forecast within the existing numbers. At the same time, the truth is that a completely new feature can be used by a lot of people or by no one. Working on something new you risk wasting all invested resources, whereas the improvement of the existing system will definitely be used by the current users.

When planning your work, if possible, look for something that users are already using and you just need to improve the system. Consider creating something completely new as an investment that may not return any value.

### Planning rule #5: There is always one thing that is more important than everything else

When resources are scarce, you need to think very carefully about how you spend them. Identify the most important thing and work on it until you get it to the product. Look at anything else only if there are resources left. Multitasking works only in a narrow set of cases, so we can practically think about multitasking as a myth.

### Planning rule #6: Separate making plans and executing them

Back when I was growing HashCode, I noticed that when I was working on certain tasks, I unconsciously optimized the task itself to complete it as quickly as possible. The problem was that in the long run I needed to implement the original task, not the optimized one, no matter how long it would take me to implement it.

There are two easy ways to avoid getting into this situation. The first is to delegate planning and implementation to different people. The second, if you are a team of one and wear multiple hats, then you need to spread out the planning and the implementation of the plan to different days. During planning, it is necessary to record in writing everything that needs to be done, what you want to get as the final result and reasoning why you are working on the task. When you start executing, turn off the engineering instinct to optimize the task and stick to the pre-written plan.

Do your planning on designated days and implement the plan on other days.

## Community grows by dividing into subcommunities

The number of questions and answers on Stack Overflow grew until 2014. In 2014, growth stopped and activity on the site reached a plateau. If you go to Meta and start reading the discussions for 2012–2014, you can see that with the growth of activity in the community, there was a growth of questions about the welcomeness of users and the difficulty of finding interesting questions to answer. To understand the possible reasons why this was happening and how these things are connected, let's take a look at the design of Stack Overflow.

The design of Stack Overflow is that the entire site is a single knowledge base, without explicit separation into subsections. When someone asks a question on Stack Overflow, they ask it to the entire community. Users who want to help other people can choose which questions to see and which to ignore by using tags, the main content filtering tool. In comparison, a standard non-Q&A online forum consists of subtopics. Each subtopic can also consist of subtopics, a typical tree branching structure. When a user asks a question on a forum, they ask it in a specific subtopic, thereby limiting the number of people who will see the post to only those who are interested in that subtopic.

Tags gave users on Stack Overflow incredible flexibility in the way they could filter the content on the site, but introduced a chaotic question flow and moved all the responsibilities for the customization of that chaotic flow onto the shoulders of the end users. By 2014, more than 8 thousand questions were asked on Stack Overflow per day. That is an incredibly large information flow, which became simply impossible to work with by using tags. As a result, users had difficulty finding questions that interested them.

The second most frequently discussed topic in those years was the increase of cases when users behaved unwelcoming to each other. Let’s recall that one of the two strongest internal motivators of participation in communities is a social motivator, that is, the opportunity to communicate with other users who are interesting to us. The axiom is that most people will never be rude to people whom they know, respect and are interested in. We are all very different, we grew up in different cultures and misunderstandings happen, but in general, rudeness in a community is not normal. There is research done by British professor Robin Dunbar. He showed that historically, people lived in communities of up to 150 people. 150 people is the size of a group in which a person can understand who is who and how each member of the group relates to other members. The professor formulated Dunbar’s number, a measurement of the “cognitive limit to the number of individuals with whom any one person can maintain stable relationships”.

It can be assumed that in online communities, as the size of the group grows, we cease to know the people around us and our self-identification with the group weakens, the sense of belonging fades away, at the same time, the stress of the users and following aggression in the group increase. At some point this usually results in the community reaching its limits in the number of users who can effectively create value together. Once the community reaches its limit, no matter how many new people you add to the community, the size of the community will not change, since for one reason or another some users will leave the group. In other words, a plateau in growth can be an indicator that the community has become “too big” for its users. If you want your community to continue growing you need to do something about the size. In practice, every online community grows according to the same pattern. First, from several people to several dozen people in one group, then the group is divided into several subgroups of several people each, and then each subgroup increases in size to several dozen. Then again, the division and growth of smaller groups.

Let me emphasize that sub groups, or sub communities, are full-fledged communities that exist within a larger community. More strictly, a subcommunity is a group of users who interact in an environment isolated from the larger community where specialized rules are in effect. The rules themselves are determined by the subcommunity users as well as the environment in which the users interact. For a subcommunity to function properly, it should have its own place for meta-discussions and the ability to implement the needed changes that have been discussed.

Let’s sum it up. The first problem on Stack Overflow at that time was the large flow of questions, which became impossible to deal with using the tagging system alone. The second problem was the large number of strangers on the site, which makes it impossible to build any relationships between users. In my view, the solution to both problems described above is to split Stack Overflow into smaller, more specialized communities. One size does not really fit all.

*Note: In fact, Stack Overflow was once divided into subcommunities. At the very beginning of its life, after a year of Stack Overflow's existence, what was supposed to happen happened. In May 2009, two new sites, Super User and Server Fault appeared, based on their separation from Stack Overflow. Subcommunities are not the same as individual communities, but the motivation behind them is identical.*

# 2. Communication

Somewhere inside, at the subconscious level, I associate growing a community with the lean startups methodology. The lean startup methodology is the process of validating ideas by analyzing customers' reactions and feedback on an MVP, and deciding if the company should keep working on an idea or pivot. The goal of the approach is to bring products to market as quickly as possible with minimal risk of them not meeting customer needs. For online communities, lean methodology involves discussing your ideas and working with users at an early stage.

When the founders of Stack Overflow first started working on the project, they recorded oral discussions in which they talked over the basics of design of the future site engine, and then posted the recordings on the Internet. In the comments to the posts, all interested parties could share their thoughts about the discussed ideas. When Stack Overflow was launched, all active development was based on the same principle. Thinking through the dynamics of the system out loud and subsequent public discussions of the ideas for future site tools.

Discussing features with the community not only made it possible to dynamically create the software that users needed to work effectively, but also had a positive impact on three most important aspects of user engagement.

- *Shared identity*. By telling how, what and why you do something, you create the story of your community.

Through stories, users develop a shared identity within the group.

- *Sense of ownership*. When users have the opportunity to share their thoughts on an issue and thereby influence the development of the platform, they feel that the platform is “theirs”. This also affects their sense of belonging.
- *Active participation*. People enjoy thinking about interesting topics, sharing their thoughts, and discussing them with other smart people.

By 2021, we had somehow moved away from the lean methodology in working with the community, and platform development, and community management began to be conducted as if we were developing not a community, but kind of a SaaS product. Instead of domain experts thinking through community dynamics and making assumptions, we moved on to brainstorming feature ideas with a diverse group of employees (note that all colleagues were incredibly educated, yes, but without any meaningful experience in community management). Instead of public discussions with users and analyzing their feedback, we began to conduct user research through interviewing several randomly selected users. Instead of sharing stories about planned tools, their design and how the tools can help the community achieve its goals, we began posting announcements when releasing new software.

There is nothing wrong with brainstorming and releasing announcements. But there are things that must be done according to their own nature. Growing community and building SaaS software require completely different approaches. You won’t run a marathon at the pace of a sprinter.

## Don’t treat a community like an audience. Audience and community are not the same thing

There is a huge difference between an audience and a community. By definition, an online community is a group of people connected by a shared goal, an interest or a purpose and communicating with each other. In other words, a community is about people who want to do something together, interact with each other and are interested in each other. An audience, in turn, is a group of people interested in something or someone. The audience’s attention is always focused on one person, particular brand or other single entity.

Users in a community interact with each other; it is always two-way communication. Community assumes collective action that usually leads to creation of value by the users themselves. The audience does not provide any means to facilitate the exchange of information between people, the audience only consumes. If you work with an audience, your primary task is to create something that is interesting to the audience. If you work with a community, then your primary task is to energize the users and in every possible way help them create what is interesting to them and other community users.

In the case of an audience, the organizers make most decisions on their own and are completely responsible for running the show, the people are passive. It is not the case for Stack Overflow, where users are active, know what they need and rarely silently agree with imposed decisions from the outside. They are used to participating in decision making.

## User feedback will always be about problems. At the same time, it is necessary to distinguish between toxic feedback and helpful feedback

In my last years at Stack Overflow, I began to notice that colleagues, especially those who had not previously worked with communities, perceived public meta discussions of the site design and its features as a negative experience. In my view, one of the main reasons was that every time they posted an announcement of a feature, all users' responses contained descriptions of some shortcomings in the proposed solution, which caused anxiety among many of my colleagues.

At the same time, the fact is that kind words do not provide information that will help make an initiative better. When one shares their work with a community for feedback, users intuitively see it as an opportunity to help the author to improve their ideas. The users begin to look for shortcomings intentionally and propose their own ideas, even if the current version looks fine to them. As a result, any place dedicated to feedback will look as if it is "negative". In reality this is not the case. Most people who really don't like something about the community quit in the first place when they encountered something they didn't like. It is unlikely that they would go as far as posting feedback for a new feature. The users who provide feedback, in most cases, are active and loyal members of the community.

One way or another, most of the feedback in the community will be about issues, shortcomings or possible improvements. Some of it will be helpful and some will be toxic. One needs to differentiate between these two categories.

- *Toxic feedback* is characterized by the fact that it is aimed at the personality of the author of the post, it contains accusations, disdain, or humiliates the dignity of others. Usually, toxic feedback does not contain any specifics.

- *Helpful feedback* is based on verifiable facts, is specific, relevant to the context being discussed, is respectful, and often contains specific suggestions for possible solutions.

Another important aspect to look at is a culture of reasoning in your community. Toxic feedback in most cases comes down to the expression of subjective opinions, which can rarely be supported by logical reasoning. Once you make it mandatory to provide reasoning for any feedback, the number of toxic feedback posts will reduce to almost zero.

## Any communication with the community should allow users to show off their skills

There are three important conditions for this to become possible.

1. It is necessary to share your developments and existing ideas with users as early as possible.
The correct approach to doing something for a community is to never wait for the final version of a feature or an idea and share what you have as soon as you have the minimum. The longer you wait to share about your work:

- The harder it is to mentally hear about shortcomings in your solution.
- The harder it is to implement suggestions.
- The harder it is for users to show off their skills and help you.

It is much easier to listen to feedback when you have invested a minimum of your time in the current solution. Moreover, the less invested you are in a particular implementation, the easier it is to abandon it if the users do not like it or there is an obvious flaw in the approach.

Developing something useful for a community largely comes down to working collaboratively with the users. The more “raw” the ideas you share with the users, the more space there is for them to show themselves off, and for you to add some of the users’ ideas to the final product. If users are involved in anything that makes it to production, then they develop a very strong sense of ownership of the community and want to invest in it even more.

2. It is necessary to clearly indicate what kind of feedback you expect users to provide.
When you post about a new feature or an initiative, you have the opportunity to set the rules of the game. All you need is to add a specific question or request to the end of the post, which will clearly describe exactly what you expect to see from the users. When you ask the users for something specific, you set strict boundaries for further discussion and anything beyond these boundaries will be considered off-topic. You and other active users can easily moderate feedback in this case.

3. You must be ready to make changes based on the results of the discussion.
Meta discussions are meaningless if there is no follow-up action by the end of the discussions.

## Communication is all about telling stories that are focused on users

Colleagues who worked on product development have always shown interest in Meta. Often they themselves wrote meta posts about the work they had done. These posts were informative, detailed with the complexities of the problem being solved and the chosen solution. The writings were good enough to be used as posts for scientific journals, but they were often not suitable for Meta. Here are the two most important reasons.

1. Story should be aligned with the software, and the software should be aligned with the story.
There is a popular expression “if it is worth communicating, it is worth a story.” Storytelling may seem like something out of the world of science fiction books, completely unsuitable for the announcement of software features. In fact, this is not true. People are narrative creatures; we perceive the whole world around us through stories, including software features.

The essence of storytelling for announcements is that we want to guide users through a planned emotional experience, creating a logical connection between the problem being solved and the proposed solution. The protagonist of the story, in most cases, will be the reader themself, so the introduction of the protagonist in this case is not necessary. We can start right away by presenting the reader with a problem that should be familiar to them in order to create empathy and interest. Then we tell the reader about the proposed solution, the implementation of the feature, thereby creating a logical connection in the minds of the reader. Next, we can share how this solution will benefit the community overall.

Storytelling about software also works in the opposite direction, *i.e.* it might help developers understand whether they have chosen the right direction for the product. Before you start developing software, think about what story you are going to tell the users. If there’s something wrong with your story, for example, you can’t clearly describe the problem you’re solving, your solution isn't optimal, or your solution doesn't provide tangible value to users, that's a red flag for development. In this case, most likely, you are doing something that is not in line with the current interests of the community. It will be difficult for you to "sell" your idea to users when the software is ready and gain any traction. If you have difficulty explaining why you are developing a feature, it will be even more difficult for users to understand why they should use it.

2. Stories should always be about users and extremely rarely about the company

Communication can be looked at from two points of view: from the point of view of the community focus of attention and from the position of the final goal of the community.

Through stories, you control the focus of users' attention. Recognition in online communities should go to those who bring more value to the community. The reason for this is that most online communities are meritocracies. Company employees do not directly create value in the community itself, except when they actively participate as users. Therefore, the focus of all posts should be on volunteers regardless of anything else.

The ultimate goal of communication is to engage users. Through storytelling, we develop a shared identity and a sense of belonging to a group. Users should be able to relate to the group you are writing about. The more they associate themselves with the story, the stronger their emotional response to what you write about and subsequent engagement will be. As a result, the shortest path to engagement thought stories is to write about the users themselves.

*Note: Feel the difference between two approaches to communication: "we (employees) decided to do this and did it" and "you (users) have long asked to do this and we did it for you".*

*Note: Stories should be as inclusive as possible, thereby bringing people together and helping them see something in common in each other. Common characteristics bring people together.*

# Closing words

Growing a community is like growing a botanical garden. To cultivate a community you will need time, knowledge and a lot of effort, but one wrong step can ruin years of work. A good botanical garden will have all types of plants from all over the world, even those that can destroy each other and that require completely different living conditions. In exactly the same way, all people, no matter what views on life they have, will find a place in an online community if they are given a chance to select the sub groups they are interested in that are isolated from the rest of the community and have their own local environments.

By working together, people can create amazing projects like Wikipedia or Stack Overflow, projects that would not be possible without the joint effort of many people. Create the right climate in the community and it will begin to bloom, delighting regular users, casual visitors and the general public.

In theory, there is no difference between theory and practice, in practice there is. I truly believe that the only way to learn how to do something is to go out and do it. If you've ever thought about creating a community or making some changes to an existing one, please don't put it off until tomorrow! Do it now. Let's create communities useful to the world, learn from our own and others' mistakes and don't forget to share our conclusions with others.

Thank you very much for being with me to the end of the book! I hope you found the book interesting.

# I will appreciate your help

I will be very grateful for your review on Amazon and appreciate it if you share the link to the book with your friends and colleagues. Please feel free to leave any comments and suggestions about the book at practical-cm.com.

Based on the information presented in this book, I created an online course on building and growing online communities that is called "**Practical Community Management**." The course is available in the text form at course.practical-cm.com and in the video format on Udemy at udemy.com/user/nicolas-chabanovsky. Please check out the course and recommend it to someone who might benefit from it.

For the last few years I have been working on community strategy, conducting all kinds of data research and doing community audits. If you see any opportunities for cooperation, please send me an email at nicolas.chabanovsky@gmail.com. I will be very glad to discuss your ideas.

# Acknowledgments

Thanks to everyone who has been helping me on the way. Particularly:

To Daria Chembrovskaya, Daniil Chabanovskii and Adnrey Chabanovskii for being my lighthouse in all the storms throughout the years.

To Joel Spolsky for buying my company and thus giving me the opportunity to see the world from a different perspective.

To the community, the amazing people without whom this book would not have been possible nor would Stack Overflow itself exist.

Stack Overflow is unique in that the company has an incredible number of talented people with whom I was lucky to work. (Folks, you are all the smartest people I have ever worked with!) I would like to highlight several former colleagues with whom I crossed paths most often and who made the greatest contribution to the development of me as a person.

My managers. Juan Garza, Tim Post, Sara Chipps, Philippe Beaudette.

Colleagues with whom I shared the fate of a community manager. Joshua Maciel, Gabe, Akiko Abercrombie, Jon Ericson, Paula Rosenberg, Joshua Heyer, Robert Cartaino, JNat, Kevin Chang, Cesar Manara, Ana Hevesi, Catherine Kuck, Spencer Vail, Bert Hall, Taryn Pratt, Miles Crist, Sarah Giskin, Vis S., Bella B.

Colleagues with whom I pushed the data endeavor. Michael Foree, Nick Pentella, Kendra Little, Anuja Chakravarty, Constantine Kokkinos, Jessica Clark.

Colleagues who helped me and our communities enormously by writing the code and setting up the infrastructure. Adam Lear, Yaakov Ellis, Marco Cecconi, Samo Prelog, Benjamin Dumke-von der Ehe, Gervasio Marchand.

Colleagues with whom I worked on the product. Jascha Drel, Cobih Obih, Desiree Darilek, Puneet Mulchandani, Jenny Kim, Anita Taylor, Jeff Tan, Mithila Fox, Carla España Lynch.

Thank you very much to all my other dear colleagues at Stack Overflow who taught me everything I know, for their wisdom, kindness and their love for what they do.

I would like to express my deep gratitude for the help in proofreading the book and pointing out errors to Jon Ericson, Anton Menshov, Néstor Tejero Orfila. Guys, your edits were incredibly helpful.

Special thanks to the book’s editor-in-chief and my friend Laine Schaublin. The amazing and inexplicable fact about my work at Stack Overflow is that when I joined the company, I spoke almost no English. Laine took over my training. She was able to do something that no one had been able to do before. After three years of studying with her, I began to speak and write in English no worse than people for whom it is their first language. I am incredibly glad that our English classes have grown into friendship and ten years later we can work on something cool together! I hope one day we will have a tea party again.

This entire book was written in immigration, in the beautiful city of Novi Sad, where I found my second home. Thank you to all my Serbian friends for their kindness, warmth and care, for lending a helping hand in difficult times, for giving shelter and bread.

“Tvrd je orah voćka čudnovata,

ne slomi ga, al’ zube polomi!
Nije vino pošto priđe bješe,
nije svijet ono što mišljaste.”

— Petar II Petrović Njegoš

# About the author

Hello my dear reader! My name is Nicolas Chabanovsky. I am an engineer, serial entrepreneur and online community expert with more than a decade experience in building and growing online communities, developing software to host them and measuring the health of the whole. In my free time, I enjoy reading, cycling with my boys, and doing CrossFit with my friends.

In 2010 I launched a Q&A community of Russian speaking software developers, hashcode.ru. By 2012 HashCode grew to twelve communities on different topics and formed the Knowledge Network. In 2014 the Knowledge Network, and hashcode.ru, was acquired by Stack Overflow. In a couple of years after the acquisition, Stack Overflow in Russian became one of the most popular Q&As for the Russian speaking developers. In 2018 I started to lead community growth for all international Stack Overflow sites. Since 2020 I have focused on designing health metrics for Stack Overflow and Stack Exchange sites, data analytics and community strategy.