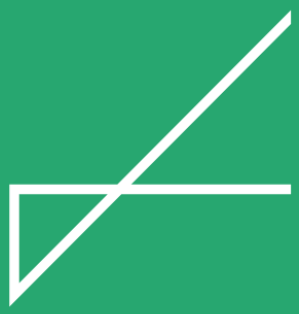
20 years

Project stories. People stories.



RomSoft

20 years

Project stories. People stories.

Foreword

By Iulia Weingold

20 years ago, there were two experienced software engineers who had a clear idea about how a software development house was supposed to work. And that's how RomSoft was founded.

At first, they had only one customer, and just a handful of skilled experts who were confident enough to join them in this adventure.

We've come a long way since then. Today, we are privileged to connect with clients all over the world in a diversity of projects, enjoying the expertise of 100+ extraordinary team of professionals.

This e-book is dedicated to all the people who walk this road with us every day, who live by the same values as we do and work towards our common goal: to make people's lives a bit better.

First in line are all the people who are part of the RomSoft team today. From 20 year veterans with the company to the newer kids on the block, we all function like a big family. You'll find some of their names inside the pages of this e-book, as they decided to share with the rest of the world their most meaningful experiences at RomSoft. This leap of faith is honoring and humbling.

Then, we're grateful to all our customers, who trust us and keep investing in us. Some of these amazing customers have decided to share their thoughts in this e-book, which is actually very cool and reassuring for the years to come.

Last, but not least, there are people who have worked with us for a number of years, and contributed in different ways to building this company. Some of them found their way into this sharing exercise, too. This shows how solid our extended community is, and that once part of the RomSoft team, you'll always be a part of the RomSoft team.

Thank you all for walking along with us. Enjoy the journey for 20 more!

Contents

Project stories

The principles that keep us together

By Nicu Popescu

20 years ago, I and my business partner Dorin decided to leave the company where we were working as software engineers and start our own software development company.

For me, personally, what matters about a company's legacy – more than the brilliant product idea, the carefully drafted business plan, or simply the intuition to make the most of a good market opportunity – is the true motivation that lays behind the shiny slogans and polished facade.

The simple truth is that we weren't business owners back then. We weren't entrepreneurs. We started by being employees, back in the days when software developers in Iasi had a harder life than people today imagine.

As maybe some of you remember, 2001 was the year after the dot-com bubble burst and inevitably, its waves hit our part of the world, too. In Iasi, job opportunities in IT were fewer than ever and developer work was poorly paid. Although we were pinned on the regional map as a vibrant "university town", young IT&C graduates who weren't set on finding a job in western European countries or even overseas, had only two or three local IT companies to choose from.

With such few options, and an employment legislation still in its infancy, it happened more than often for salary payments to be delayed, or for employees to be treated poorly.

This situation led us to see how the company we were working at was slowly drained out of its best people, exceptionally good professionals, who were choosing to leave. It was heart-breaking to see all that talent go.

You know how the business jargon is full of bombastic words sometimes, and phrases like "people are the greatest asset that a company has" certainly sound like that. But I swear it was the moment I realized that this one was, actually, very true. For me, that was the turning point.

A promise that may sound cliché...

So, together with my business partner, we made the decision to start our own company, where developers would be treated like human beings, and we convinced a core team of great people to join us in this new venture.

In the end, I can honestly say, as commonplace as it may sound, that we started RomSoft with this promise: we will continuously invest in people.

We started encouraging people to develop their skills, and at the same time we allocated funds for courses and trainings. We motivated people to get together and exchange knowledge and ideas, and

also granted paid days off when they had important upcoming professional exams. As much as possible, we tried to inspire a continuous learning attitude among our team members.

At the same time, we knew that with learning and experimenting may come certain mistakes that anybody can make. We tried to encourage a constructive attitude towards failure, and support people when that happened, as long as lessons were learned and progress was noticeable on medium-long term.

Later on, came a second promise: **to keep communication channels open at all times**.

We've always been this flat hierarchy company where ideas and initiative are encouraged, but along the way I realized that people need tangible instruments that they can interpret as a "safe place" in order to open up.

To give just one example, a few years ago we initiated a program called MOOOD (management one-on-one day) – a day every month when I clear up my schedule for anybody who wants to come in and have an honest talk about anything career, work environment, or profession related.

A third promise that we made in those "wild-wild west" like days, was to **pursue excellence** in everything we do.

Looking back, none of our titles or certifications were obtained in order to show off, or to check an item on a list. The idea was to use them as much as we could, to improve the development process and to raise the technical level of our team.

We wanted to help our people grow their skills set, so that they could have information at their fingertips. We wanted to be able to deliver the most elegant, optimal solutions to our customers' challenges.

Today we are proud of our 10+ years of Microsoft Partnership that come to validate our continued individual and team efforts that serve both the company and each of its members.

Anyway, it is clear to me that from these honest, gut felt promises that we've made in those early days, we derived our RomSoft values, and the core of our business philosophy.

Our core values

Here's a quick overview of our company values, which you can explore in more detail on our website:

Continuous learning

At RomSoft, we always did and always will encourage continuous learning, and give people the best tools to keep on learning throughout their entire RomSoft career.

Knowledge sharing

We encourage knowledge sharing as an alternate form of continuous learning, as we believe that the best ideas come in collaboration, from people who are not afraid to share their vision and inspire others.

Cultivating quality

I know that quality is a term that can easily become overused, especially in the IT industry. That's why we prefer to measure it through sound reference systems, like our ISO-9001 certified Quality

Management System; or our long-term Microsoft partnership. Either way, we care deeply about quality in everything we do.

Engaged people = productive team = happy customers

Finally, I would add a fourth item on the list. Maybe principle is too much said, but it is something that I believe sets us apart. We encourage **developer autonomy** as much as possible, giving people enough safe space to be creative, to learn, and to find the optimum solutions to challenges ahead.

Following this idea, when we define our development process we take into account the team characteristics, the project requirements and our customer's business culture. We don't blindly apply methodologies and frameworks just because they are popular. In our development process, all elements must come together in a functional ecosystem, where creativity meets performance and professionalism.

To sum it all, we honestly believe that engaged people lead to a productive team and a happy customer.

That's why we prefer to build long term partnerships with our customers. These partnerships are a special breed of business agreements, where risks and benefits are shared equally among the partners. It is the type of collaboration that allowed us to make the shift from being just another outsourcing provider to being a product development company and a research center for our customers.

As a final thought, I don't want to put out there a misleading message. No matter the principles or the business culture, some people will leave the company. And some people did leave RomSoft, in search of a different career, a particular job or experience that we just couldn't offer, or simply to start their own business venture.

But we did our best to stay true to our initial promises and to the principles that emerged from them.

There were quite a few times when those principles were tested, like the 2008 financial crisis, the 2018 fiscal system change, or the more recent 2020 Coronavirus crisis. As bad as these moments were, we always put people first and got to the other end together. And I hope that my teammates can confirm that nobody was ever left behind.

Nicu Popescu is co-founder and Managing Director at RomSoft. Outside of his packed daily schedule, whenever possible, he will happily use his expertise to advise and guide aspiring tech entrepreneurs in the local community. And decorate the company spaces with beautiful home-grown roses from his own garden.

The beginning of a great partnership

By Sebastian Dumitrescu

Sometime between the .Com crash and Second Life, RomSoft was being founded by two ambitious software engineers in Iasi, Romania. The ITC wave was in its forming days and year after year new businesses, from the most diverse areas, found a way to benefit from this (r)evolution.

We started in a period of uncertainty. In 2001 many companies were closing their activities due to the dot-com crisis. Many developers were deciding to relocate to areas where they found more opportunities like Timisoara, Bucharest, or even abroad.

Since 2004 - 2005, opportunities started to appear again, as many international companies were opening subsidiaries in Iasi.

Our first project

In this context, we were more than happy to start the work on our first project, K-Expert - a work area manager for medical laboratories. The customer was Sysmex Europe, part of a Japanese-German corporation, and a top player in the laboratory diagnostics field.

K-Expert was a Win-32 desktop application programmed in C++ that performed numerical and graphical data acquisition, managed patients, orders and tests, processed results, and performed industry specific quality control validations.

I guess it was pure luck that Sysmex Europe was our first customer. Back then, I and my colleagues were making frequent trips to the customer sites, where we had the opportunity to collect direct feedback, learn about what our customer needed and how they pictured our product could help them.

I remember the feeling going to the labs, connecting to the first analyzer, the emotional rollercoaster waiting for data to arrive. That's when I understood that this product was more than some lines of code put together. A medical laboratory meant complex dependencies, hierarchies and workflows.

Understanding our mission

I personally understood for the first time the deeper meaning of our work. We were developing a product that could touch many lives in a positive way. And we wanted to keep on doing that.

That's how our company mission came about. Making lives a bit better. These five little words express just how honored and humbled we are to bring our small contribution to this great purpose: better, affordable healthcare.

On the other hand, working in the healthcare industry is not something that you take lightly. It comes with great responsibility. That's why it's a highly regulated space.

We soon realized that our business model had to be built around quality standards and procedures if we wanted a long-term collaboration with this customer. The next step was to understand as well as we could our customer's business needs.

Finding a soft spot

The objective was to deliver a service that was so good that we would become indispensable to our customer.

At that time, Sysmex was focusing on selling analyzers and reagents. They needed know-how on the software development front. We understood this need and decided to fulfill it.

We realized very fast that good software was not enough. We needed to deliver top-quality solutions. We nailed down our priorities:

(1) One of them was to have topnotch requirements engineering.
(2) Another one was a rigorous documentation process.
(3) And third, we needed a complete testing approach in order to control quality and eliminate bugs before the products being released.

The development process we built in those days was later implemented and further developed by Sysmex within their own technical department.

The turning point

A turning point in our business relationship with Sysmex was a meeting held in Hamburg, in 2005, called by them "The Regional Initiative". At that time Sysmex was having two R&D centers. One of them was a company from Tallinn, Estonia, called Systek. The other one was RomSoft.

In that meeting, the two centers presented their work strategy and business philosophy in front of the Sysmex management.

Our strategy was based on the following key-points:

Development process

We understood that a structured and methodical development process was best suited for our client's needs. Sysmex was catering to medical analysis laboratories all over Europe. These labs were starting a complex automatization process of their functional workflows – so their information systems would revolve around heavy data processing and a highly structured lab hierarchy.

With these prerequisites in mind, we thought that a waterfall development process would be preferable to agile methodologies, with strong emphasis on source code documentation and requirements engineering.

Emphasis on quality

Having an ISO 9001 certified quality management system by Moody International and a TGA-DAR accreditation obtained in Germany were competitive edges that we wanted to use to our advantage.

Solid foundation

We needed to layout a solid base that would sustain an organic development for us as a company, and for the growing software development needs on the client side (both in size and complexity) in the future years.

Sysmex Europe itself was going through some profound transformations. They understood that they couldn't limit to just being a distributer of analyzers and reagents. On those analyzers, they had to incorporate a highly performing software to manage all flows of data in a diagnostics laboratory.

The software that was available on their analyzers at that time was made in Japan, and was less suited for the European market, from the user experience point of view. The user interface was unattractive, heavily packed, less design oriented. In Europe, users wanted an interface with fewer elements, and better suited for their specific workflows.

We realized that Sysmex was going to need software development services long-term, and our goal was to become a trusted partner.

We also recognized that one key feature of Sysmex products was quality. This transpired all over their communication strategy. So we thought that if we were to develop applications for Sysmex analyzers, these applications needed to be very stable, with no bugs, so we started doing complete testing and automated testing. We cared about deadlines, but quality came first.

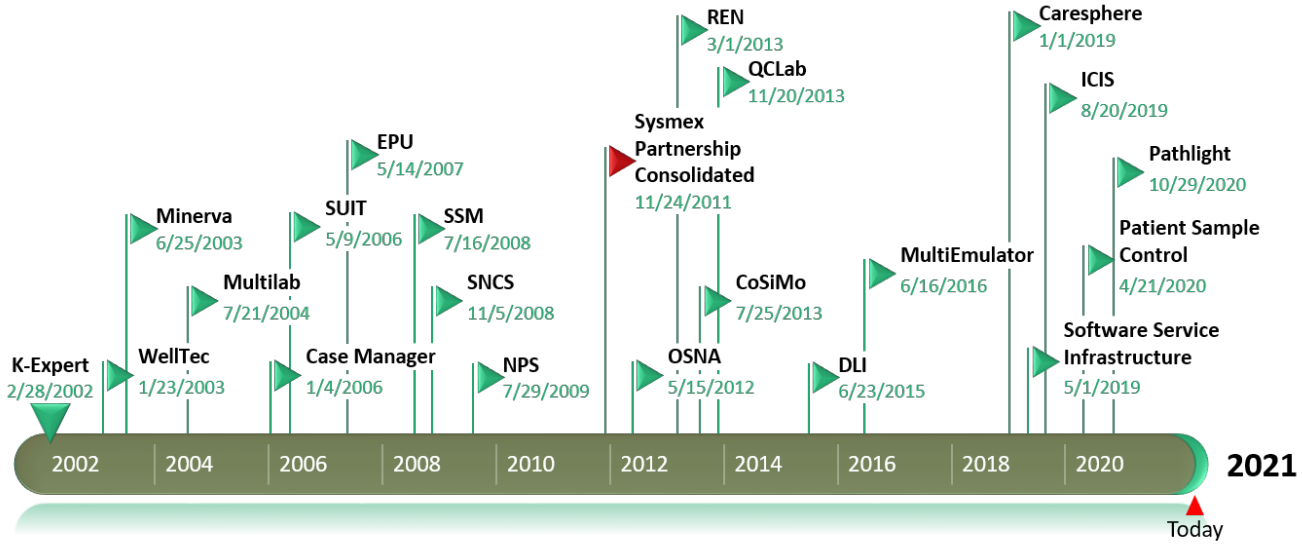
The beginning of a great partnership

At that moment I believed we were in a competition with the Estonian company, and that we were starting from the least favorable position. Tallinn (the alma-mater of the Skype software, among other things) was an important hub for software development in those years, and was definitely seen to have more potential than Iasi.

The Systek team was inclining towards more lightweight development methods such as the Agile methodology. Eventually, the business relationship between Sysmex and Systek was short lived, and ended a couple of years later.

As for RomSoft, I think they appreciated our commitment to a long-term vision and the fact that we understood their needs. This paved the way for our long term collaboration which led us to become their main external R&D center.

In the following years, we developed many more interesting projects together with the Sysmex Europe team. It would take a lot of pages to speak about every product we build together. What I can do instead, is to use another software we are proud to develop – Office Timeline, to showcase how (almost) 20 years of great partnership look like.



Sebastian Dumitrescu has 18 years of experience as project manager, team leader and software engineer and is Customer Line Manager at RomSoft.

“We are so fortunate to have found RomSoft”

Interview by Iulia Weingold

Tim Stumbles & Eddy Malik are co-founders at Office Timeline, a US-based bootstrap startup that entered a strategic partnership with RomSoft in 2012 to develop their product. During one of their regular visits to Iasi when we reconnect, share our visions, sketch our future, and between long meetings and late hours, we couldn’t miss the opportunity to ask them a few questions about this partnership and how things look from their perspective.

Please tell us a bit about how it all started: Why a strategic partnership with an overseas company, like RomSoft? What brought the need?

I love to tell this story because our partnership with RomSoft is simply good luck, very, very good luck! We had prototyped the product but knew it needed professional development to get it stable and ready for release to market. Having worked virtually on international teams for the previous 10 years we were not shy about building an off-shore partnership. Also, as self-funded start-up we needed access to the most talented developers but at slightly better economics than we could find in the US market.

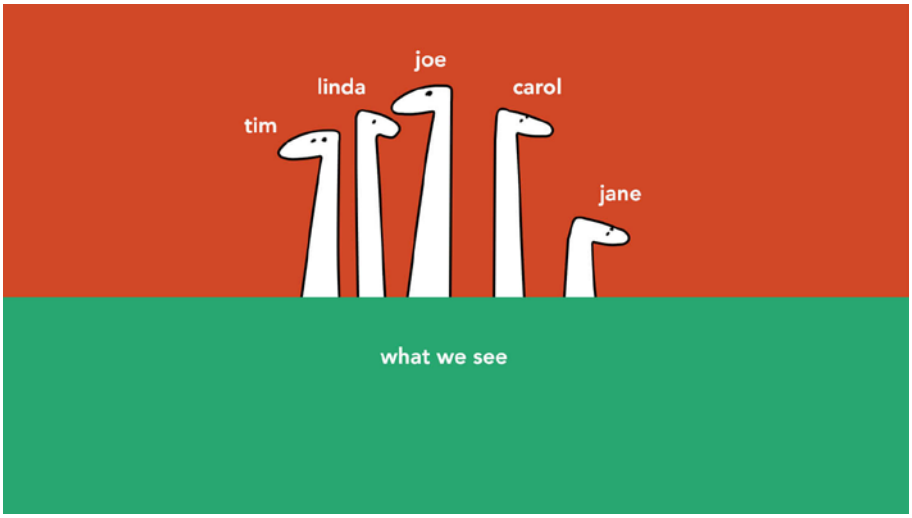
I actually looked in a couple of different places but what was completely unique about RomSoft was their appetite to share the risk in the venture. At that time we were investing from our personal savings and

RomSoft was really flexible with a proposal that allowed us to start slow and grow without having to make a big commitment we could not afford.

What does it actually mean to have a "strategic partnership"? Please give us some details from the concept standpoint.

This topic excites me because in my former career I worked around strategic partnerships often, however I didn't really appreciate what the word meant. Now I see a true strategic partnership is when each party is driving for a 'win-win' outcome like we are with RomSoft. A true 'win-win' can't be achieved unless each party is genuinely working for the other party's interests as well their own. To succeed, strategic partnerships need to be egoless.

There is no-way this can happen without a high degree of trust, mutual respect. We are so fortunate to have found that with RomSoft and it is the reason we are successful and the reason we will continue to build our business around this partnership.



Strategic Partnership Concept (I)

What do you think are the main benefits of this kind of partnership? Are there any downfalls?

We simply could not have accomplished what we have without this partnership. And I don't mean this just from a product development point of view, I mean it as I think of all the things required to incubate, launch and grow a business. The intangible benefits we have received from this partnership have been critical to our success. There are challenges and risks but the partnership is working so well that we have built our long terms plans around it.

Strategic Partnership Concept (II)

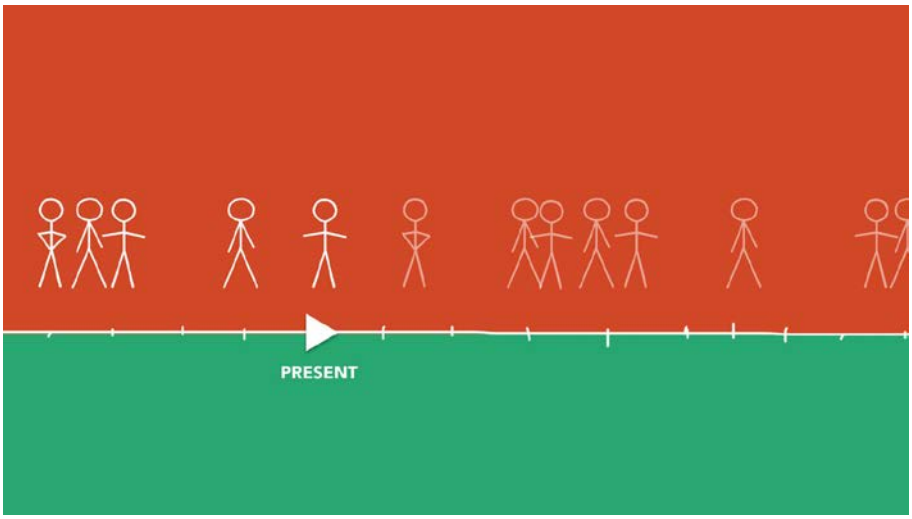
During the last years, the team grew in size, proficiency, but also in skill diversity (by including testing, marketing, design, and even support components). It's like a tri-dimensional growth. Can you talk a little bit about the team development strategy?

This was interesting for us because we only ever intended to build a development team at RomSoft, but later we decided to bring in non-development roles. The reason our strategy shifted was because we found the talent at RomSoft exceptional, and we also found them extremely easy to work with.

When it came time to consider roles beyond developers, the value of having them centralized made a lot of sense. We added and continue to add non-development roles to the team. We knew RomSoft developers were world class, but what has been an unexpected surprise for us is the high level of talent we have found for non-development roles.

What are the main challenges that come with the growth?

There are no shortages of challenges in starting and growing a software business. When you self-fund a business like we are doing, the number one challenge is speed to market for new innovations. Since we can only grow based on the performance of the business it often means there is a lag in time from the moment we need new people to the moment we are actually able to hire them.



Hiring Pattern Prediction

What are the major themes for future development?

Our top-most priority is to please our customers with insanely good software and outstanding service. We don't always get there but we continue to learn and improve with that purpose in mind. Beyond that goal there are obvious management themes that I think about, such as continuing to optimize all of our teams, staying lean and nimble and developing new innovations that delight our customers.

Any other thoughts that you want to share?

Yeah, I love the team. We work together every day and each time I travel to Romania, it is like connecting with old friends. Maybe it is the laid back, non-hierarchical organization that makes is so fun; or because it is exciting to build something new and watch it growth in the market.

Also, the business is reported transparently each month and everyone on the team is empowered to make good business decisions. This kind of flat, empowered structure is perfect for us and something I really enjoy.

Thank you very much Tim, Eddy. Your words always inspire a new vibe to the entire team and give us confidence that we're part of something special, that not so many tech specialists in Iasi are lucky to experience.

Developing an innovative company culture

By Lucian Nita

In the early 2000s we started by developing applications for companies who needed to outsource their software development needs. But within a few years' timeframe, RomSoft has grown a vision to develop its own research and development department. I guess that's how I got hired at RomSoft, in the first place.

From that point on, RomSoft has built its long term strategy around this objective, which then influenced its development processes, choice of projects and management decisions.

To clear out a small detail: research is expensive, and its results - somewhat unpredictable. So if you're a software development SME from Eastern Europe and want to bring your contribution to reshaping the world through technology, where do you start? Launching your ideas on KickStarter? That might help too, but RomSoft decided to explore other options.

The biggest advantage of being a Romanian based SME is to be part of the EU. The European Commission can be the first potential "investor", through its R&D strategic funding programs (dating back in 1984).

What's your area of expertise?

After this revelation, there's a bit of research work to do in order to understand what type of projects should you apply for? The best way to go is to analyze your company's CV and see what your expertise areas are. If your genius idea falls into one of these expertise areas, it's a winner.

At RomSoft, for example, we have solid knowhow in developing software systems with application to various areas of modern medicine, like e-Health, telemedicine or self-health management. These areas are emerging precisely from the opportunities created by the rapidly evolving technologies, addressing problems that could not be addressed a few decades ago:

- Reducing patients' dependency from doctors and caregivers
- Better, improved data on patient history
- Prediction and prevention of acute episodes of chronical diseases
- Less invasive treatments

Is your idea unique?

Having figured out your area of expertise, there's something else to make sure of: your idea must be unique, at least at European level. Also, it has to solve a global problem. The EU doesn't really want to waste its research and development budget on small purposes. Throw in feasibility and you have a good candidate.

Hold on, there's more. The EU may not care about your idea, even if it's unique, global and feasible. It has to fall into a category of interest. In order to check, go to their calls for proposals page.

There you'll find a complete list of European Union's priority research themes. Domains like health, information technology, environment, sustainable development are top of the list. The better your idea is aligned with one of the announced themes, the more chances to succeed.

No place for lone runners

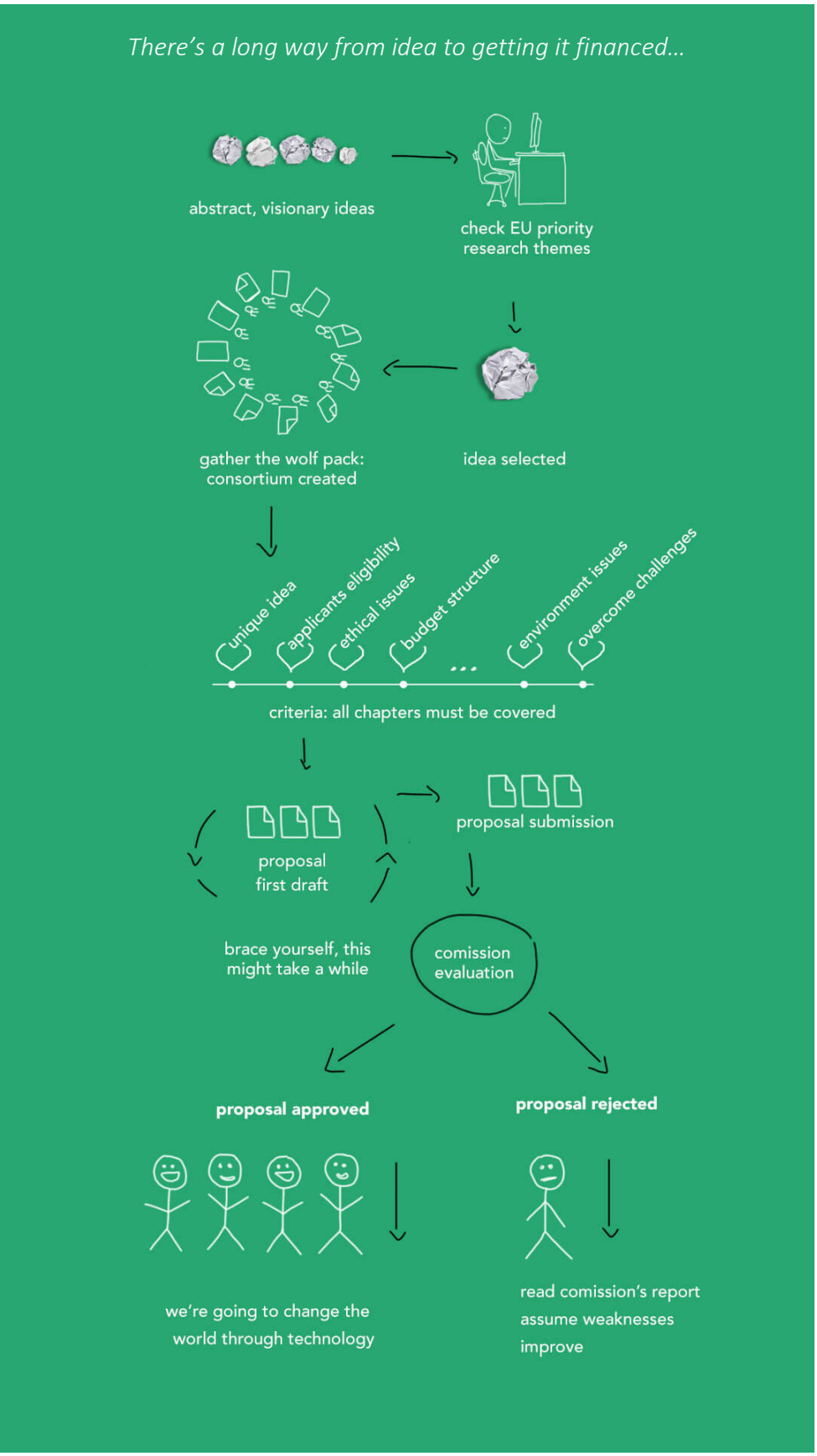
The third rule learned by someone looking for EU R&D funding, is that this is not really a place for lone runners. You'll need a wolf pack. Well, in bureaucratic Brussels terms they are called research consortiums, but you get the idea.

Brussels clerks have a very precise check-list of parameters when they perform the selection of projects. For instance, if you take a university – all they care about is: does the idea have scientific value? If you take a company, they'll ask: does this have a market potential? But from the EU standpoint, a project is analyzed from all these perspectives.

A research consortium must be well-balanced, to include universities – who sell pure research; to include entities that can apply the research, for example, in the medical field – hospitals and clinics that will test the results; and also, to include private companies that will develop a physical product based on the research, a system to be marketed and sold. The more balanced the consortium, the better its performances.

To find partners for your consortium, you don't have to yell in the woods with the hope that someone will hear you. There are specialized entities that you can address to, like the Enterprise Europe Network. EEN is an institution created mainly to help European SMEs increase their competitiveness on external markets. It offers both EU funding expertise, as well as innovation and transfer technologies consultancy. Their services are offered for free by several hundred territorial member-agencies, including chambers of commerce, technological centers, universities and regional development agencies. You just have to identify the agency closer to you and ask for its services. In the North-East region of Romania, you can contact either Tehnopolis or ADR Nord EST.

Or, you can join a research and innovation cluster that fits your targeted expertise areas. For example, RomSoft is founding member of the Imago-Mol cluster, the only medical imaging cluster in both Romania and the EU. These clusters strive to link public bodies, like hospitals, to private enterprises, especially firms in the IT and software development sector.



Expectations vs. reality

Up to this point, things look clean and pleasant. Find an idea, find a consortium, write a proposal and hit the submit button. But reality may look a bit different, as depicted in the roadmap on the left.

A track record

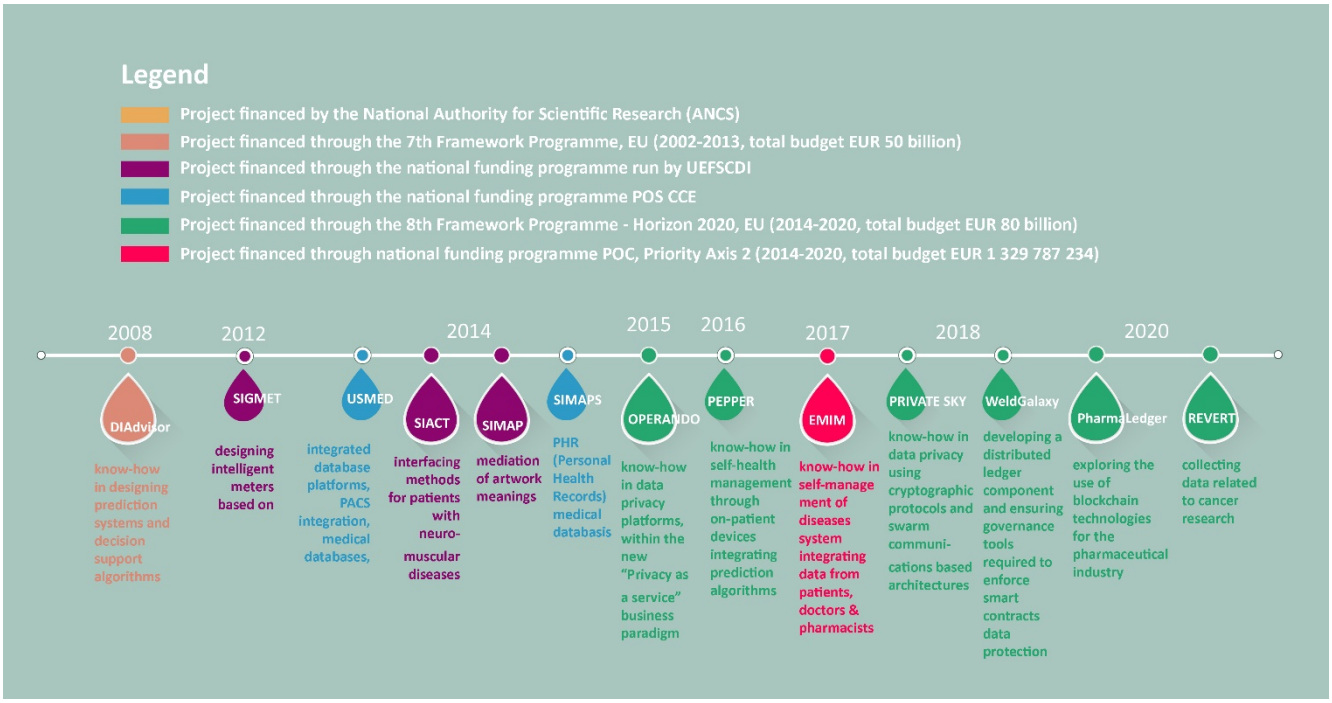
The first project that got financed since I came as a head of research & development at RomSoft was DiAdvisor™. It was a personal blood glucose prediction system, meant to help patients struggling with type I and II diabetes to lead a much closer to normal life.

In a later project, PEPPER, we got to continue developing the innovations from DiAdvisor™, so that the system could be used with an insulin pump. The patients wouldn't have to inject themselves anymore, as the process would be completely automated.

But I could confidently say that the project that came the closest to a marketable product is EMIM – a fully functional online platform connecting patients, doctors and pharmacists.

The integrated pharmacy app of EMIM was featured last year on Euronews, in a piece depicting how European business clusters have been driving innovation despite the difficult context created by the covid-19 crisis.

A visual timeline of all the research projects developed since I joined RomSoft would look like this:



The cumulated financing attracted by RomSoft was a modest EUR 4.006.000, with a co-financing totaling a bit over EUR 1 mil.

Take away thoughts

To sum it up, the most sensitive aspect of the whole process is the way the project proposal is written. Be precise. Be complete. Cover all chapters, no matter how unimportant you think some of them. Act quickly. Calls are active for relatively short periods, and you have to submit your proposal in due time.

If your proposal is rejected, it's not the end of the world. You will get a report from the commission with key aspects you need to improve. This is valuable knowledge, don't ditch it in the trash bin. Some ideas get financed the 3rd or 4th time around. The only condition is that you have to rename your project. And of course, to improve it.

In the end, research and development is not as straightforward as you may want it to be, but here are some thoughts to help you put things into perspective:

(1) Not every research idea will prove a good idea when put into practice. A research project must be thought within reasonable risk limits

(2) Every research project is a learning process and if at the end you can say that you've gained some new insights about how things work, this is a progress

(3) The innovation capability is like a muscle: the more it's exercised, the stronger it gets

(4) There is no longer research for the sake of research. Though we don't necessarily like the word, we must come up with new products, especially innovative ones, with significant added value, so that we can compete (at EU level) with the US or Asian markets, that have a major advantage due to being more result oriented.

Lucian Nita is Head of Research & Development Department at RomSoft and Associate Professor at the Technical University "Gheorghe Asachi" in Iasi.

The benefits of being Microsoft certified

By Iulia Weingold

In RomSoft, the idea to become a Microsoft Certified company came, at first, from a very simple principle: the development tools were expensive, and so many people around us were simply pirating them. Time confirmed that what was at first an ethical problem has now become a serious security issue and a liability that no company can afford.

The day we became a Microsoft Silver partner (in June 2010), we celebrated with enormous satisfaction. The Silver partnership came with 25 Windows licenses and 10 Visual Studio licenses, at a moment when the cost for a Windows license was pretty high, even for IT companies like RomSoft. Six years later, in April 2016, we became a Microsoft Gold Partner for software development, and we've maintained this level ever since.

It's a continued effort for both the company and the team and we are very proud of this sustained marathon that inspires us all. But I often wondered about the story behind the milestones, the titles, and the accolades. So I've spent some time talking to my colleagues to understand what it means to get a certification and why it's important to have one. This is what I've learned.

Why should a software development company be Microsoft certified?

For a company, as more and more developers become certified, regardless of the program – Microsoft, Oracle, and Red Hat – a trend is created. People will have information at their fingertips, and they will know how to prove it. In front of peers, and in front of customers. They will be able to prove that they know, what they know and how they know it.

The customer will see the company in a different perspective. If you want to win over an important customer, especially a customer who puts value on procedures, processes, quality, and a customer who works in a highly regulated industry, then this customer will probably appreciate working with a certified company.

In the same way, our ISO 9001 certification helped us a lot in our collaboration with our customers. As the most prominent name on the list, there is Sysmex, one of the top laboratory diagnostics and healthcare companies in the world – since it operates in such a highly regulated industry.

The 27001 Security Certification is another chapter in the same book – to own this certification means to have a high level of security for your company's Intranet, a well configured firewall, database, and backup system.

Nicu, one of RomSoft's co-founders, concludes it better:

> *Looking back, none of our certifications were obtained in order to show off, or to checkmark an item on a list. The idea was to use them as much as we could, to improve the development process and to raise the technical level of our team.*

Last but not least, these partnerships come with strong financial incentives. Through our Gold Microsoft Partnership we have access to 30 MSDN (Microsoft Developer Network) subscriptions, 100 Windows licenses, 100 Office licenses, support, and more. These don't cover all our software needs, but a fair chunk of them.

What does it mean to study for a certification exam?

Not being a developer myself, I turned to some of my colleagues who took on the challenge and went through a Microsoft certification process.

From their point of view, there are two aspects: The first one is that you learn because you need to know, and to own that knowledge. The second aspect is that you have to learn in order to pass the exam.

On the practical side, people have different learning patterns. They use different instruments and techniques to learn faster and more effectively. From mental mapping to video tutorials, from notepad to blog notes, from PDF reading to old-school highlighter on printed exam courses, everything goes.

But in the end, going through the bibliography at least once and taking some test exams to anchor the new concepts is what seals the deal and helps you pass the exams.

What makes people keep on learning?

The motivation that drive people to this sustained learning effort may be very different, but it seems that there are some common themes that kept on emerging in our conversations:

Preparing yourself to use a new technology

When you study for a certification exam, for a specific technology, you learn a specific way to use that technology. More exactly, you learn how to use it, and how not to use it. What are the most important scenarios? What are the scenarios that were initially projected, and which ones proved viable in production?

In Microsoft technologies you have this situation very often. They come up with a new technology. Then, the technology goes through several iterations, until it becomes truly mature and you can count on it. These certification exams are an opportunity to learn.

Gaining a more comprehensive view of how technology evolves

Occasionally, a certification process helps you find out about concepts that you don't necessarily bounce against in your day-to-day developer work. Some questions help you think about scenarios or functionalities that you wouldn't normally think relevant.

Stefan, who is one of our Microsoft certificates veterans, offered a unique perspective on the AZ-204 certification - Developing Solutions for Microsoft Azure.

"I liked that I interacted with some technologies that I had first contact with during an older certification. In the meantime, the technology has changed a lot. But it was nice to see how the SaaS transition was conceived, from its early developing phase (where one can't argue it was very solid) to the current phase, where you can feel that Microsoft really is interested that SaaS becomes a very important part of their business offer."

Achieving a new career milestone

Being internationally recognized, a Microsoft certification acknowledges a certain professional level that you have for using that specific Microsoft technology. It guarantees that you have a certain technical level to anybody who wants to work with you.

To raise yourself at the team level

A more particular motivation comes from the belief that nothing comes easy and you should always keep yourself in check with the professional level that is required on a specific job market. In IT, the bar is set very high.

"The less knowledge I have in a certain technology, the more I am challenged to prove to myself that I can learn it. It doesn't matter the amount of time that it takes me, I will get there, where my colleagues and peers are."

Irina, Software Developer for the Extended IPU project

A mindset for continuous learning

This way of thinking is very common among IT professionals. It was very well summed up by my colleague Dan, who is a Principal Design Engineer for the Office Timeline project:

"Personally, I like to learn anyway, because software development is my hobby. And I want to do my job the best way I can. There will always be new technologies, many of them solve existing problems. So they help you do your job better, write better code. The applications that you develop will become more robust, and will serve the final user better."

Can you study for an exam and keep working on your projects as usual?

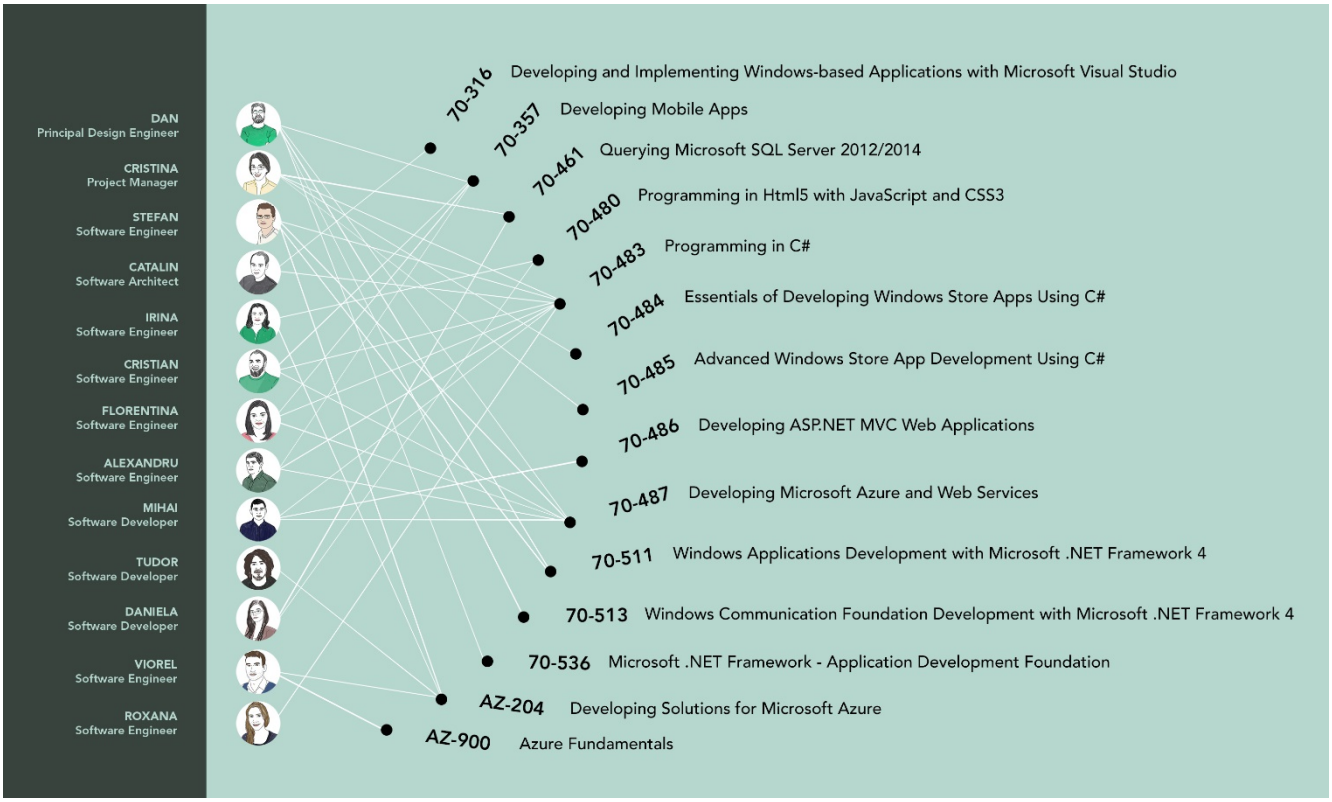
That is another concern that I thought people might have when pursuing a Microsoft certification. As we all know, time is an important asset in our professional and personal lives alike. Sometimes it really feels impossible to allocate the extra hours to study for yet another exam.

The workaround we found at RomSoft was to offer a 5 days study time per exam that can be allocated whichever way the developer feels comfortable. Of course, often times, 5 days are not enough to learn a new technology. But it's a starting point.

The preferred strategy among people I've spoken with was to take a little bit of time every day. Either at the beginning or at the end of the work program, or when they have idle time. It is important to learn in small steps to allow information to be fixed. The (not so) secret recipe is to have a first contact with the information, then try to use it, and then go once again through it. This helps to use as much as possible your logical sense and not learn by heart.

Most popular certifications in RomSoft and key takeaways

In search of some visual gratification, I made the following RomSoft Microsoft Certifications infographic.



The top three certifications by number of people owning them are as following:

70-483 – Programming in C# - 8 occurrences

70-487 – Developing Microsoft Azure and Web Services – 6 occurrences

AZ-204 – Developing solutions for Microsoft Azure – 5 occurrences

In viewing these stats, I can't help but wonder what is it about Azure related certifications that makes them so popular? Following a fascinating conversation with my colleague Dan, I could see that he is kind of the master mind behind this outcome.

As he puts it:

> *"Azure is a set of technologies so new, so powerful, freeing the developer of so much redundant work… I'll always encourage people to be curious about it."*

For example, in many instances we prefer to use Azure Storage instead of a relational database, because it spares us a lot of effort. Similarly, using Azure Functions (Lambda). Or Azure Key Vaults, able to properly secure the application secrets. Or Azure Static Web Sites, greatly improving the app performance, reducing the costs, and diminishing the attack surface.

If your project is suited for Azure technologies only, you can focus solely on your business need, and implement only business logic. The entire project infrastructure, the plumbing part – Azure is taking care of.

However, no matter the certification type, there's an important personal effort behind every one of them. And all of them, together with the non-Microsoft related certificates, all the trainings, and the practical knowledge that is gained in the everyday work field, create the unique technical fingerprint of RomSoft.

Iulia Weingold is public relations specialist and copywriter at RomSoft. She worked for many years as a technical writer which led her to better understand why the IT industry needs a lot more of creative communication through storytelling.

A few thoughts on being Agile

By Dan Vasilov

In software development, we deal with a huge amount of buzzwords. People hear about certain practices and then try to blindly apply them, without going too deep into what they mean.

We did that, too. And, at a certain point, we decided to stop. No matter how revolutionary a technology or tool is, it shouldn't be used just because it's bleeding edge. I learned in time that it is best to apply what is necessary, what solves the problem in the simplest, most elegant way.

I've always been in conflict with people who said that Agile is the best option for every project. Because I really believe there is no such thing as "one size fits all". Actually, in my own experience, and that of many experimented engineers that I know, in order to successfully run many software projects that we've been working on, we had to take them into a minimum Waterfall sequence.

I realize that this is where someone might interrupt me and make the correct observation that for Office Timeline, the project that I'm currently pouring heart and soul into – and have been so for over 10 years, we do use Agile. Yes, that's true. But it's a very personalized Agile. And in order to personalize Agile, or anything else for that matter, you first need to know it inside out.

A bit of context

Since it emerged in 2001, people have written volumes about the Agile methodology and all its instruments. I cannot even begin to cover the subject extensively from a theoretical point, in this article. But a short definition, and a bit of context, may be of help.

> *In software development, Agile practices include requirements discovery and solutions development through the collaborative effort of self-organizing and cross-functional teams with their customers or end users, adaptive planning, evolutionary development, early delivery, continual improvement, and flexible responses to changes in requirements, capacity, and understanding of the problems to be solved.*
> *– Source: Wikipedia*

In 2001, seventeen software developers met in Snowbird, Utah to discuss lightweight development methods. Together they published the Manifesto for Agile Software Development. Based on their combined experience of developing software and helping others do that, the seventeen signatories to the manifesto proclaimed that they value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

So far, so good. There's nothing to come in conflict with my own principles for a successful software development process.

But the problem with values is that they need to be accompanied by procedures and tools in order to be implemented.

For instance, any team needs to measure its productivity. In Agile, it is measured in story points per iteration, which shows the team velocity. For example, one team can deliver 10 story points per one iteration and another team only 5. But can we say with 100% certainty that the first team has performed twice better? Not always, because measuring software development work is not only a quantitative problem, but also a qualitative one.

As I mentioned before, for Office Timeline (OTL in short), we do use Agile and SCRUM. And we've been doing so for many years, to the extent that we came to a point where we considered that we can responsibly tweak some things that just don't apply to us. Here are 7 ways in which we personalized our software development process in OTL:

#1 Assess the team level

A common sense rule is that you have to assess your team level first. The Agile methodology, and moreover the SCRUM framework, may help the dialog and feedback in certain types of teams.

In trusting that we work with people of certain capabilities, and a certain technical level, we also trust that people are able to collaborate beyond many of the Agile/SCRUM enabling techniques.

#2 Developer autonomy

So, now that we assessed the team level, the basic principle is that the developer should have as much autonomy as possible. In OTL, a developer receives the problem and is allowed to come up with a solution proposal. If the problem is a complex one, there may be more than one people designated to come up with proposals.

I can make an analogy with a concept that is used in the military strategy, if I'm allowed – the "spearhead" concept. The idea is to concentrate as much firepower into a small front as possible, so any defenders in front of them will be overwhelmed. As the spearhead moves forward, infantry units following in the gap behind them form up on either side of the line of advance in order to protect the flanks. In a software project, the idea is to allocate the best people that you have to find the optimum solution as quickly as possible, and then move on to the implementation phase.

That doesn't mean that the spearhead always decides what the best solution is. For instance, they could come in front of the team and present the possible solutions and then we decide together. The collaboration is as flexible as possible.

#3 Fluid roles

If developers are permanently caught in a “two-week sprints / user stories” scenario, a certain type of anxiety may surge that they’re on a spinning wheel and there is no exit strategy. As a tweak, in OTL and more generally, in RomSoft, we are allowed to have fluid roles. I’m a principal design engineer. Up to a point it’s similar to software architect. But I’ve been through many roles. Build engineer, release engineer, support engineer, software developer, and even software tester for a very short while, due to a project emergency.

If you think about it, this kind of flexibility really is in the spirit of Agile. A spirit that sometimes feels that we are sacrificing for the sake of ceremonies and rituals. I think any project role can be beautiful, as I consider every one of them an opportunity to learn. And a bit of variation where you can gain a fresh perspective on the project, is also good.

#4 Transparency

Transparency doesn’t just mean open floor space and keeping people in close check at all times.

Sure, at certain times, you may have strong reasons to increase transparency from the bottom up. But at the same time, you should make sure to improve visibility from top – down, as well. What are the objectives? What is the greater purpose of doing things in a certain way?

You should maintain several communication channels opened and make sure that information flows freely as soon as it is needed. One side transparency will not work.

#5 Avoid cargo cult

There were moments at the beginning of my career when I fell victim to cargo cult. I read something in a book and, very proud and excited I tried to implement that solution in a project I was working on. As it later turned out, it was not the best solution for that particular project. And I learned from my own mistakes. Now, if I see this happening in a project that I’m responsible for, I intervene.

As a methodology that is used at such a global scale, Agile hasn’t been spared of this phenomenon - where it has been transformed in a religion and cargo cult.

For example, you cannot blindly implement the stand-up ceremony. It is a short meeting, a daily kick-off that should not take long. If there are more serious problems to discuss, they have to be transferred to a different environment.

The idea to stand up is to enforce in a way the short duration. But we shouldn’t implement it undiscerningly, as we work with smart people who are capable to organize themselves and respect a consensus. They don’t need to be tricked into it. We need to retain the essence of things in Agile.

#6 Pass on the information

What could be of more help than using tricks? Implementing some healthy principles such as a timely distribution of information.

For instance, if I solve an important problem, I don't have to wait for the next day standup. Or if I have a problem that needs to be solved, and I know the person who can help me, I will choose the fastest channel – direct communication. These are common sense.

#7 Maintain the principle, allow the flexibility

Sometimes, it is more important to retain the principles of Agile, but to adapt them to the people you are working with.

For example, as much as we adhere to the principle to maintain iterations as short as possible, we don't have a 2 weeks enforcement rule. And this kind of flexibility is allowed in other areas, too.

In another instance, we noticed that our new engineers experienced a very smooth onboarding process and they integrated quickly to the team. As such, we thought it didn't make sense anymore to get them through those stages of forming a team from SCRUM, forming -> storming -> norming -> performing.

Last but not least, you should never have to be forced to hold a meeting if there's no reason for it. For example, in OTL we have 30 minutes meetings - three times a week. But if before the meeting we realize that there's nothing on the agenda, we ditch the meeting.

As a side note, this personalized system works best when people in your team are responsible, autonomous, guided by strong work ethics, and announce problems as soon as they appear.

Closing thoughts

Agile is subjective. There will always be people who think it's the Holy Grail of development process and there will always be people at the other end, who will think of it as the ultimate productivity killer. But I think moderation is key. And that the secret is in communication.

Taking a step back, I hope this doesn't feel like I'm complaining. In IT we are privileged, and I feel that in the RomSoft – Office Timeline partnership, we have a superior set of privileges. We are lucky to work with a customer who is open to communication, and who would listen to anyone from the team, no matter their role in the project. People will always be heard. We have a flexible schedule. We have access to one of the best project infrastructures in the world, which is Azure, and we have extended support for continuous learning and professional development.

I just needed to make a point that no matter the technology, methodology or tool that we use, we should filter everything through something that we all call common sense.

Dan Vasilov is a Microsoft Certified Professional and Principal Design Engineer for the Office Timeline project.

Programming for medical devices is a whole different game

By Simona Vizniuc

Some developers enjoy working on different projects – the more diverse, the better. And some prefer to focus on a specific type of product. For the past five years (more than half of my software development career), I have been working as a software developer for various medical devices.

During this time, I found this "niche" to be quite rewarding. Here are just a few reasons why:

- First, it feels nice to work in a domain where you know you can make a direct impact on people's health. You feel that your work is useful and relevant.
- Second, it's a great opportunity to get familiar with how embedded systems work. Consider this as your "foot in the door" technique. If you've always been fascinated about what happens down there inside the machine, then a software developer position in an embedded medical project is the best place to start.
- Last but not least, it's an area where you can shine pretty quickly if you have a mind set on innovation.

A few starting tips

OK, let's say that you've made your decision. You want to develop software that commands a medical device and you're wondering how hard it can be. I'm happy to share a few tricks that may help you speed up such a career transition.

#1 Be curious about how things work at a physical level

There's a certain physical, more tangible dimension when developing software for an embedded system. You can see how your software communicates with or commands various physical components of the device. You can feel the direct impact your work has on the device.

In the following video you can see me testing the device commanding a hydro massage bed, which is one of the projects I've been working on at RomSoft.

Also, you collaborate with teams from other disciplines, and get to learn how a software product interacts with the mechanical part, with the electronical part, and so on. You can witness an entire process and this, in my opinion, is the most beautiful part when developing software for embedded medical systems.

#2 Sharpen your programming skills

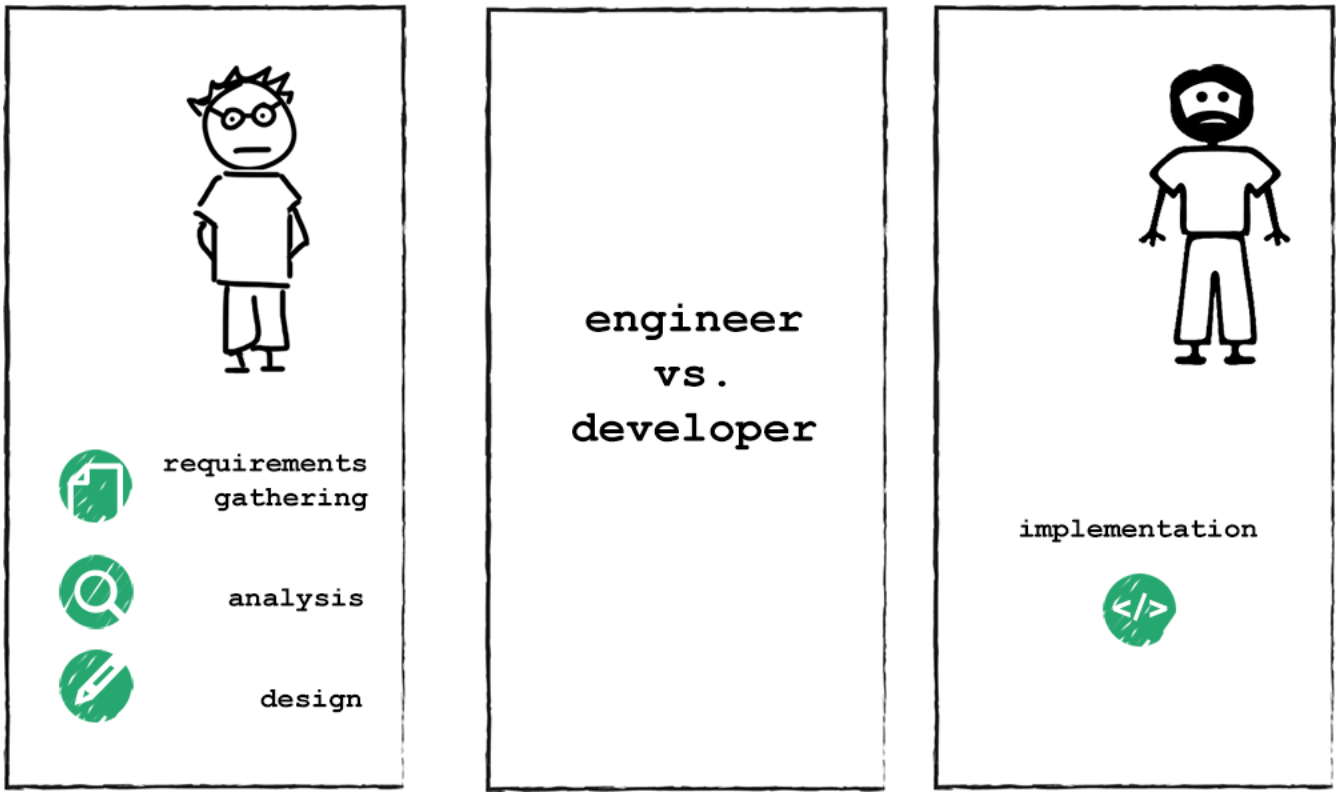
A common misunderstanding in programming for embedded systems is that you have to learn C or C++.

Nowadays, many embedded projects are destined for what we call single board computer systems. These are usually powerful computers on their own, where you don't have limitations on computing power side or storage capacity and you can use all kinds of high level programming languages.

For example, C# and WPF offer enough embedded dedicated libraries and support. There's even a Windows IoT that is destined for device application development.

#3 Think less like a software developer and more like a software engineer

While both software developers and software engineers are highly skilled professionals, there are some differences that are important when it comes to developing software for medical devices.



One important difference is that a developer takes functional specification and delivers the code required within tight parameters, essentially completing the task in isolation. Whereas a software engineer should be able to do everything that a software developer does, but with a bigger picture view. This means that they need to be more focused on structure design and eliminate technical debt, or solve a problem while minimizing the trade-offs to other parts of the system and its architecture.

For example, there are situations where you have to be aware of the environment that your software interacts with and to adapt to them, either it's about the size of a specific display, a touch screen behavior, or some patient-device interactions.

#4 Get ready for some unusual testing situations

There are two testing stages in medical devices, similar to all embedded systems: simulator testing and device testing. The latter can be pretty messy and hilarious, but it will also help you gain additional skills (physical endurance may be one of them).

Medical device testing

I remember an instance when a fellow tester was trying to put the CT injector we were working on in an error state (a CT injector is a device that pumps contrast solution when you do a computed tomography scan). The specifications were that the error would occur after every 8 liters of contrast solution was pumped through the device. So our colleague was holding a bucket full of water with one hand and feeding the contrast solution pump with the other hand, trying to reach that upper limit that would generate the error message. Much different from desktop applications testing. And funnier, too.

#5 Don't use open source software

In programing for medical devices, using open source software is not usually accepted because of the domain regulations. This is why sometimes you may need to write your own libraries or frameworks. But this can be fun, too.

In line with this idea, system upgrades are also a challenge of their own kind. Suppose you are using a version 4 framework, and after 6 months, version 5 becomes available. You have to be aware that any system upgrade may impact your final product in a different way. This may require the manufacturer to redo all tests and clinical trials, not to mention all paperwork requested by regulation authorities. It's like dealing with a whole new product all over again.

#6 Understand quality standards and risk mitigation

Everything that goes into a medical device, whether it's software, mechanical and electronical components, or substances, has to be closely tested and validated.

When you develop software for medical devices you must strictly follow the industry required standards. These don't necessarily limit your work, but help you discover possible problems more timely and efficiently.

For example, when a desktop application freezes, you can press a few buttons and run it once again. But if the freeze happens during a procedure when contrast solution is being injected in the patient… actually, no, this is never an option. The final system should be nothing less than perfect.

#7 Choose a company with a solid record of similar projects

As a final point, I would say that choosing the right company to help you navigate through this career change is important.

In a large company, there's this tendency to split things in very specialized roles or tasks that may limit your possibilities to gain an overview of the entire project and development process.

On the other hand, a team of one or two people could either mean the project is too small or that you'll end up filling too many roles and take the trash out, too. Also, not ideal.

What you want is a company with consistent experience in developing software for medical embedded systems. You want to have people to learn from, but also have an impact on the overall project outcome and positively influence somebody's life through your work.

While working for various medical embedded systems at RomSoft, I've had the chance to branch out into other areas of the product lifecycle, like writing documentation, architecture design, or validation and risk mitigation. Handling a bucket full of water when testing a CT injector or short-circuiting the computer board of a massage bed were some of the side skills acquired while learning the serious stuff.

Will it be worth it?

Considering how much our current lifestyle depends on embedded systems, and how rapidly technology evolves in the medical field, I would say working in this type of project would secure you a highly sought skills set on the future job market.

Medical devices get used more and more every day, and nearly everything is automated, in order to help or facilitate the medical act. It's easy to foresee that this domain will see explosive growth.

Not to mention that you can be fortunate enough to start from a simple device like an insulin pump and implement an idea that could bring a significant improvement. It's an area that allows you to put that creativity to work and build something important. If you're the type of person who enjoys programming as much as helping others, this type of job will make you thrive.

Simona Vizniuc is a full stack developer & Team/Technical Leader at RomSoft, with a passion for embedded systems and single board computer programming.

A journey from frontend-frameworks to native web components

By Rafael Mastaleru

Everybody has at least one horror story about a renovation project that spiraled out of control because of dependencies. You know, when you just want to repaint the bedroom, but it turns out that you also could change the windows, and then redo the electric wiring too... and before you know it, your entire house turns into a construction site for months on end.

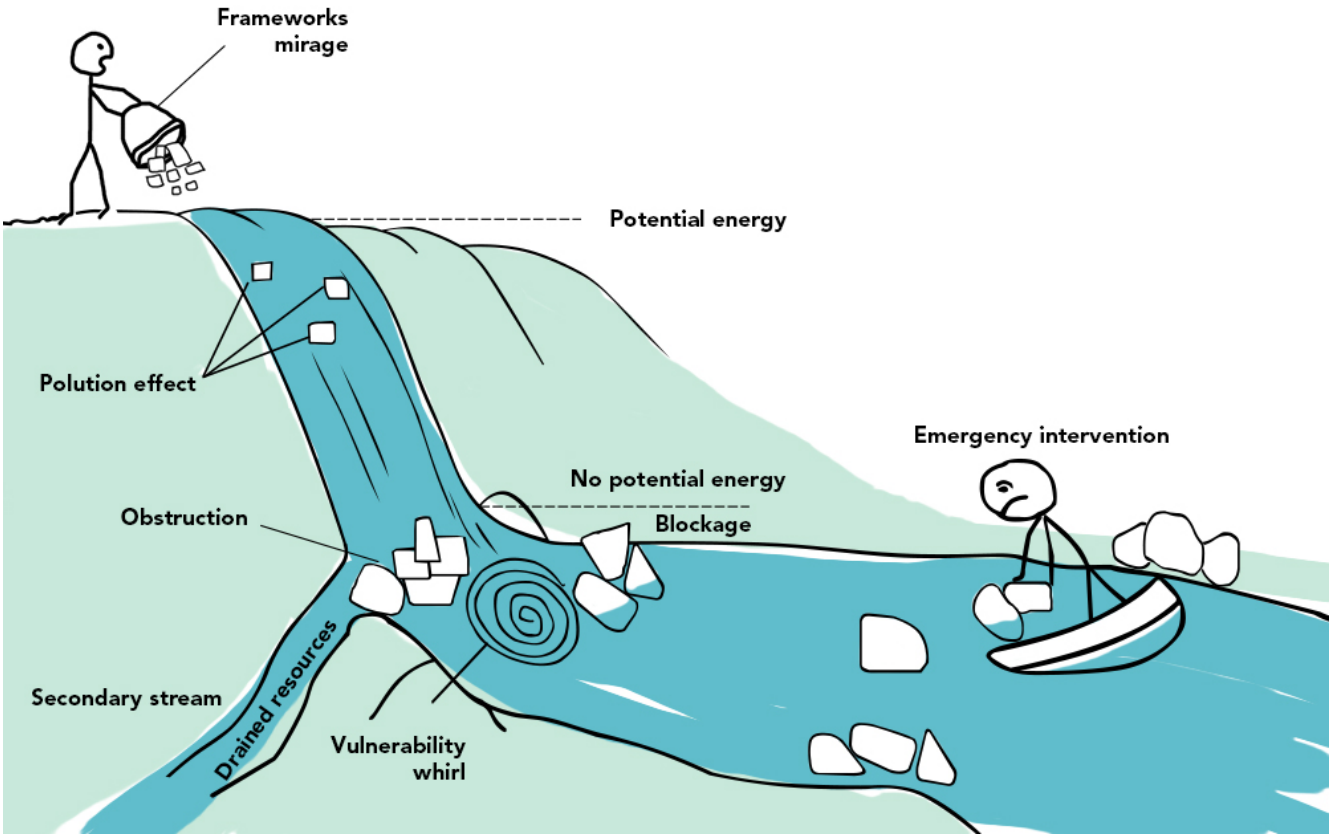
Just like in our daily life, in web development you can also get lost in details that may give you short term satisfaction, but on the long term they prove to be a pain.

The frameworks mirage

Web development is filled with trends. Nowadays, most web applications use front-end frameworks like React, Angular, Vue or Ember. The communities for these frameworks are huge, and there might be a chance you could find a small library out there that already satisfies your needs. But that library might contain a vulnerability, or it might not be suitable for one desired platform.

In time, instead of advancing with your project, you'll end up patching and solving problems that weren't planned.

The self-poisoning waterfall



Here's a bit of my experience: A few years ago, my team and I were working on a software solution that was intended to be a cross-platform, crypto-wallet app. We wanted to build an app that worked on web, but also on devices with iOS and Android, and we were also thinking about a desktop app.

After a bit of research, we decided that using React-native will fit our needs. We started to make the first *Hello World* screen and everything worked smoothly. In only one month, we had a prototype working on 3 different platforms. Also, the React-native was working just fine for mobile device platforms.

But soon we hit the first issue, when developing the web screens using React-native.

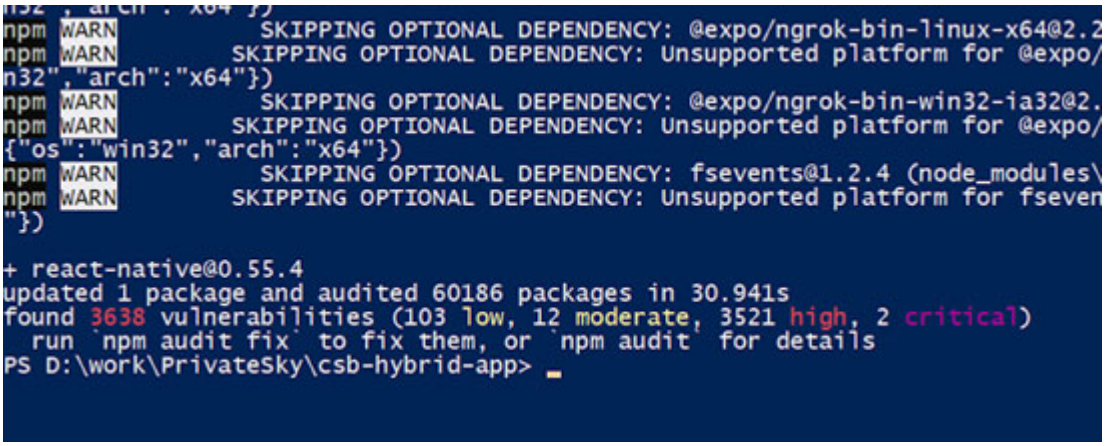
To solve this, we found a React-native-web library that allowed us to use React-native components for web. Still, we were happy we were able to write single code that would work on all platforms, even though we had a lot of compatibility problems between react-native, react-native-web and the building tool called expo.

As we advanced with implementing the app functionalities, we started having big issues with vulnerabilities. On one hand, the React-native was trying to keep up with the reactJs, but on the other hand, the react-native-web was almost abandoned.

Some problems came from unexpected areas, like excessive use of NPM modules.

It is known that in software development you don't need to reinvent the wheel when it was already invented, tested and the carbon footprint was already minimized. You can use these open-source libraries (exported as modules through NPM) and publish your work as an offering to NPM. That's why open-source is so cool. And yet, if you are building more than a faculty project, things are not so <3ly. (Funny fact, did you know that there's a NPM module wrapping your text with hearts? Check-out the npm-in-love module.)

Mind you, we were building an app to keep user's secrets better than conventional cloud storage services. We had the idea, we had the motivation, but yet, with no proper infrastructure, we ended with build messages like the following:



So that's what dependency hell looked like. And that is the point we turned around for alternatives.

The frameworkless movement

After this experience, as a team leader, I was ashamed and felt guilty that I fell into the trap of using well marketed, yet, beta version frameworks and libraries. After some research, I found out that I wasn't the only one in the universe who had this pain in the ass. There were many others and they suggestively titled themselves the frameworklessmovement.org.

They even created a manifest of their movement, where they defined four principles that should be rigorously debated before starting a web project:

- The why: The value of a software is not the code itself but in the reasons behind the existence of that code.
- The context: Every decision should be made considering the context. A good choice in a given context could be a bad choice in another one.
- The choice: The mindful choice of a framework is a technical one and should be made by technical people, taking business needs into account.
- The transparency: The decision-making criteria that led to the choice of a framework should be known to all members in the team.

Looking back, it wasn't the first time I used frameworks. In the past I successfully used AngularJs for developing some awesome apps. But the current experience disappointed me and made me realize that I violated the first 2 principles:

- → I put the value of code first, even though I wanted to develop a cross platform app where data security was crucial.
- → I didn't think too much about the context of making a cross platform app.

Tackling web components

When I realized the bad choices I made, I tried to find new alternatives. This time I knew that I had to develop an application that first of all is a secure digital wallet and second, is a cross platform app.

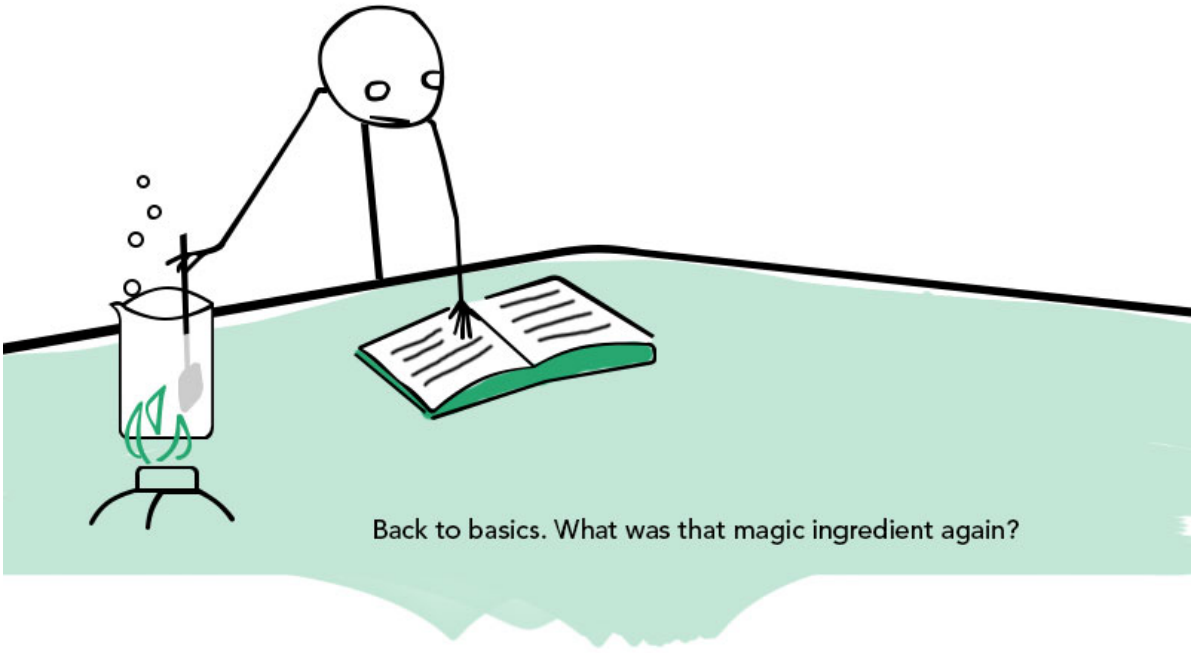
Don't get me wrong. We liked writing react components. We liked JSX templates and we really found Redux useful. And while all these seemed to be convenient to our needs, we didn't want to hear about dependencies anymore.

While we were fixing dependencies, we started to look around for better solutions and observed an increasing interest for native web components.

You must know writing web components can be exhausting, because you have to write the style, the HTML and the JavaScript code in one single file. But surviving dependency hell had somewhat shed a softer light on these shortcomings, and made us embrace the future of web development: Native Web Components.

Alchemy exists

We dug further for more resources to overcome this. After a while I found StencilJs, a self-proclaimed powerful tool for building web components, and at the same time maintaining the concepts of the popular frameworks such as reactive data-binding, virtual DOM, async rendering and JSX. And, as I said, JSX templating is so useful and I really didn't want to give up on it.



That put sparks in our eyes because we had a background with other frameworks and it was easy to adapt and code using Stencil and everything was accomplished with almost no direct dependencies.

What followed from there was a continuous achievement using web components without any dependencies. The project is not completed yet, but we're close to a first release and I'm really satisfied with the results so far.

Final note

In case you're wondering, I don't regret the bad decisions I took in the first place. I think it's important to learn from professional mistakes as well as admitting that we're humans and we're vulnerable to bad choices and well-marketed illusions.

As programmers, sometimes we can't see the forest from the trees. We may be novices in the field, or we may be led to the wrong path, or we just may be too stubborn to accept that we made some bad decisions. The key is to avoid repeating the same mistakes and to find other paths closer to the solution.

Rafael Mastaleru is a technical leader at RomSoft. He is passionate to help people protect their online privacy and keep their online personal data safe by writing specific software applications.

A UX lesson from a decade ago

By Sebastian Dumitrescu

Do you know that feeling when you put all your efforts in a project? When after months and months of hard work you finally arrive to what looks like a good result? And you feel like this project has claimed a part of your soul and a good chunk of your physical self too? And you have the champagne bottle on ice and can hardly wait for the moment when you will be finally able to pop out that cork with the rest of your team?

Well. It was one of those projects. The Sysmex Non Public Pages (or NPP by its short name) was a system designed to synchronise complex documentation from a web library to employees' laptops. It provided a wide variety of information for Sysmex employees, distributors and partners. The content varied from scientific to product and technical information.

And here came the problem: the technical information provided in NPP was needed by Sysmex technicians to maintain and repair Sysmex products at the customer site. Quite often it was not possible to go online to access the necessary information, as the labs usually had poor connection to Internet, if at all. In order to be able to browse the NPP information offline, a Windows application had to be developed, that would be capable to download all NPP contents.

As it turns out, it was easier said than done. A very particular mix of challenges and misfortunes made the project roadmap look like tangled wires. When one problem was addressed, another one surfaced.

A first big challenge was with the way information was organized on the Sysmex NPP site. Then, the communication between the development team and the customer was somewhat indirect, which made requirements gathering rather difficult. The planning was also quite rigid and sometimes required us to bend over backwards in order to meet the project milestones. And on top of all, the team needed to learn some new technologies on the go. The expectations were high and the stress level even higher.

Long story short, it wasn't the smoothest project. But inevitably, the release came and the customer seemed rather pleased with the outcome and that champagne was finally pulled out of ice.

Time passed and some feedback arrived from the customer side that the technicians' laptops were heating up significantly during the synchronization process.

We felt that cork kicking right in the face. All that time spent fighting an imaginary dragon, only to find out it was barely a lizard. In the end, we returned to the design phase considering this significant piece of information.

Hopefully, together with that bottle's content, some useful lessons went down, as well:

The first one, and most valuable for our development process, was to always question the requirements, and never refrain from asking the "stupid" questions.

The second was to not plan too much ahead, but rather iteratively, and to live room for change.

And the third one was for our hurt ego to slowly absorb in the years to come: on the long run, your past mistakes have the tendency to bail you out of difficult situations. But only if you learn from them.

Sebastian Dumitrescu has 18 years of experience as project manager, team leader and software engineer and is Customer Line Manager at RomSoft.

What I've learned working on AI/ML projects

By Claudiu Tescu

Before coming to work at RomSoft, I had been studying artificial intelligence (AI) and machine learning (ML) for a long time. I used to participate in various AI competitions and even won a few prizes. This was back in 2003-2005.

In 2007 I organized a talk with my colleagues about AI and concluded that we could get involved in this kind of projects if the opportunity presents itself. Well, it did, 10 years later, and our managers, knowing my interest in the field, asked me if I was interested to join in.

Obviously, I was happy to accept the challenge. We've started with some small scale projects, with the objective to take small steps in exploring the artificial intelligence and machine learning areas, to acquire skills and capabilities, and to build a project portfolio.

What is, really, artificial intelligence?

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition, and machine vision. In a larger sense, AI is supposed to perform tasks that not only are usually done by humans, but also require human intelligence and discernment.

While the general feeling is that AI and machine learning are very present on today's public agenda, when you actually work in the field, you start to realize how distorted perception from reality really is.

As members of the general public, our imagination is fed with what we see in the movies. Humanoid robots looking like us and talking like us, AI software taking over the world, or leaving us without jobs.

But in reality, AI represents an automated interpretation of statistical data in order to optimise some parameters. At this moment, we can confidently say that machine learning is basically data science.

In order to launch an artificial intelligence/machine learning project, you need data analysis, you need people to perform that analysis, and you need to invest large amounts of money with a 60% failure chance.

Working in AI/ML can mean a lot of things

Working in the AI/ML field can require a different way of thinking. As I explained before, you work with a huge amount of data. On the CNN (convolutional neural network) preparation part, I may be less fit to work than, let's say, a more artistic person, as CNN is an AI algorithm that is most commonly applied to analyze visual imagery.

Imagine the way insects see the world. The same with AI, you need to split the image in small squares and then reunite them. A more visually inclined person may recognize more rapidly the differences between images, if, for example, looking for texture or color differences.

Then, there is the data science area, where you need an analytical person. A person who likes to analyze and work with big amounts of data, who can organize and prepare data.

The programming skills required are a small part – here we work with the classical algorithms, an input and an output.

And there are also the mathematical skills that you need to use a lot in the research part, where you need to correlate the data.

90% of the time spent developing an AI based project is consumed in negotiations to obtain better, more relevant data.

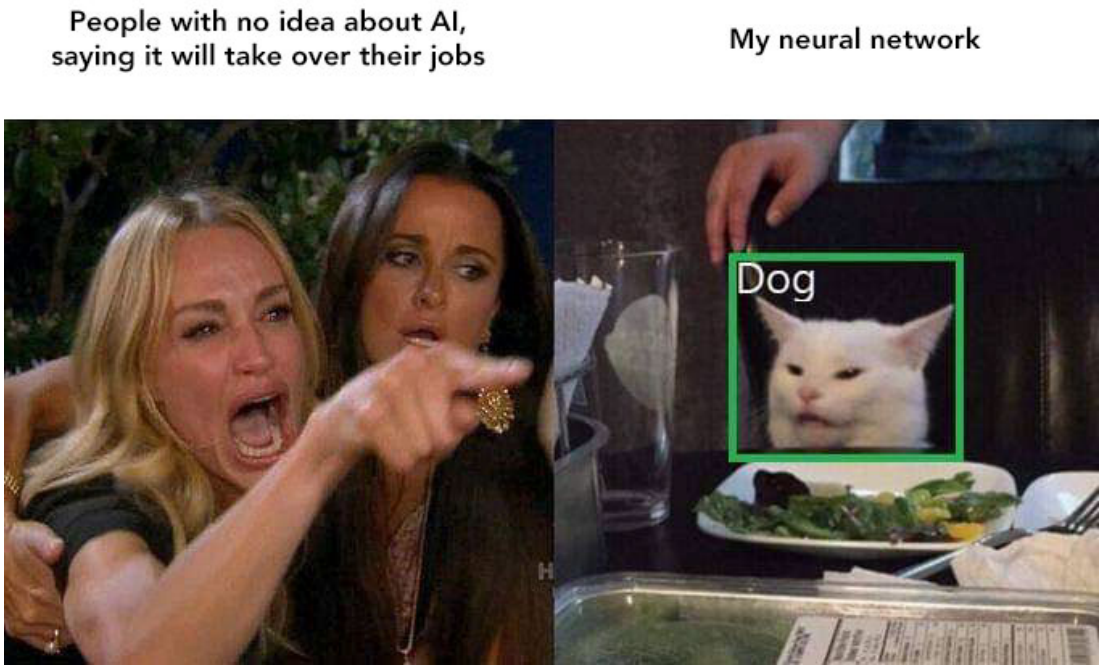
I can give you an example from a project I worked on.

I was trying to figure out, from a range of different images, where, in the photos, there was a human figure. But each picture was taken with the same background that included a green fence.

The program is building some kind of network and tries to see the most common thing for all photos. And the algorithm rendered as "true" all images containing the fence. Of course, it was not the answer we were looking for.

This shows how important the quality of data is. In this situation you turn to the client and let them know they need to redo the entire photo archive. This means more effort and more costs.

Speaking from my experience so far, 90% of the time spent in developing an AI based project is consumed in negotiations to obtain better, more relevant data. The development phase is the rest of 10%. That puts a lot of effort on the client.



Who can work in an AI based project

The technical background that a programmer needs to work in AI related projects is not very different from regular projects. A strong taste for mathematics, complemented by a natural interest for statistics are a great plus. If, on top, you are gifted with some sort of visual thinking, then you and AI are a perfect match.

Of course, depending on the project, things can get very specialized. You cannot be a Jack of all trades all the time. In the automotive industry you may need more statisticians.

To recognize defects or objects in photos you may need graphic designers and visual thinkers. And this is where the negotiations start, and probably a business analyst may have a contribution to translate things from one side to the other.

The skills you may need to work in an AI project are evolving (and sometimes in surprising directions). Some data scientists even consider that the ability to make good PowerPoint slides may be more important than the ability to use the most sophisticated deep learning models.

What are the real challenges in AI today

Locally, we are still taking small steps. There is only one company in Iasi that specializes in AI and machine learning, and we collaborated with them last year on one project and hope to do it again in the future.

But otherwise, everybody is trying to allocate an exploratory budget, while working on other things. So we can say that we're all facing the same types of challenges. At country level the situation is not that much different. There are still very few companies that are profitable only with AI projects.

Given that these projects have a strong research component, we also started to explore the EU funding programs for developing AI and ML projects.

Last year we wrote two projects but they didn't come through. This only motivates us to make more efforts and try again. And it is encouraging that more funds are flowing into AI.

Through the Digital Europe and Horizon Europe programs, the European Commission plans to invest €1 billion per year in AI. It will mobilize additional investments from the private sector and the Member States in order to reach an annual investment volume of €20 billion over the course of the digital decade.

This is good news since at EU level, there is a shortage of specialists in AI and ML related fields, and a 5 to 10 billion EUR investment gap.

Ethics, a challenge we can't ignore

Science was never requested to answer moral questions in all history. This job was for philosophers only. But with the raise of AI and algorithms, the situation is about to change. For example, in the case of a self-driving car, the software controlling the car should have embedded the answers to some moral issues.

The classical example is who should be saved from an impact, the pedestrian or the driver, if given a situation where it is impossible to save both? It is clearly a very complex issue that engineers are now bound to answer.

This is an extreme example, but the truth is that all forefront technology domains, like AI, blockchain or bioengineering are, at the current moment, highly unregulated industries and could raise many ethical problems. Maybe philosophers, psychologists, or ethics professors could also make an important contribution to these areas, alongside engineers and technical people.

What is (y)our place in the AI (r)evolution

Beyond the concern that is usually advanced by the media, that AI and robots are going to take our jobs and render us humans useless to society, I would speculate that many other jobs will emerge from this technological revolution, like in all past revolutions.

The AI movement will touch the majority of fields and will need the contribution of all kinds of specialists. With these worries addressed, we can now focus on more practical issues.

It takes 3 or 4 engineers to work for one year, round the clock, to bring some results in a major AI project. And they can't be junior engineers. You need to dislocate them from a current project that is producing money for you now. And the market place is limited when it comes to experienced specialists.

So if we, the EU, want to fill in that investment gap, we should start thinking long term. And if private companies are willing to invest and explore the possibilities of the domain, states and the European Union should do their part too, and maybe, if we can meet halfway, we can work towards some relevant results.

Whether we like it or not, even if at this point we are, in terms of the development of artificial intelligence, at the mouse-brain thinking level, evolution will probably be exponential, and I expect that

in 10 to 15 years we will reach technological singularity. And we can't act surprised when this happens. It's up to us to be part of this movement and to bring our contribution.

Claudiu Tescu is a Team Leader and a Software Architect in the Extended IPU project. He is passionate to solve complex technical problems, and will always lend a hand when other project teams bump into difficult challenges.

People stories

pitiCODE – our kids programming course

By Carmen Stavarache

In early 2019, some of our teammates had the most wonderful idea, to create a programming course for our employees' kids. Soon after, the idea grew into a small, volunteer-based internal project, named pitiCODE.

I've been drawn to the idea as soon as I heard of it especially that I have two candidates of my own who are curious about everything that involves looking into a screen.

Along with nine other colleagues who decided to get involved, we decided to create this course, where we would slowly, and in a playful manner, introduce children to notions like operators, algorithms, or basic principles of computer programming.

For some of them, some of these notions were known. For others, they were completely new.

But with the help of visuals and stories, and a graphical programming language for kids named Scratch, we managed to help them catch on pretty quickly. I liked very much a term coined by our manager, Nicu, who said that programming with kids should be named "fungramming" – these lessons were as fun for them as they were for us, too.

But with every lesson, we were amazed of how fast they were learning. I remember that in the first edition we had four year olds who didn't even know how to hold the mouse. When the course ended they were able to create an entire game and they were so proud of what they had accomplished, that they couldn't wait to get home so they would continue playing in Scratch, or show off to their siblings.

I was once more convinced that children can do anything they put their mind to, if guided with love and given information that is properly adapted to their level of understanding.

Since I joined this project, I get asked a lot by people around me, what is the final purpose? Do we want them to grow up to be software developers, or follow other IT related professions?

I don't think that at pitiCODE we prepare them for a future job. We don't know exactly what jobs will still be around in 10, 20, or 30 years.

Scratch and algorithms and programming notions are just tools. Maybe we work with the tools that we know best. But behind these, we want to help them develop their logical thinking, their critical thinking, and their creativity.

Although we put them on hold when the pandemic started, I can't wait for the moment it will be possible to resume our pitiCODE meetings. I'm proud to be a trainer in this program, as much as I'm proud to be a parent. These kind of ideas can turn the workplace into a playground, and as a former child myself, I'm always happy to pitch in.

I'm proud that my children have the opportunity to see this side of me too, and to associate my job with a place where you go with a smile on your face. I think that's part of our legacy, too.

Carmen Stavarache is Project Manager for the teams developing the Office Timeline suite. A former math teacher, she now uses her teaching skills in her favorite role at RomSoft, as a pitiCODE trainer.

Why be a RomSoft intern?

By Corina Enache

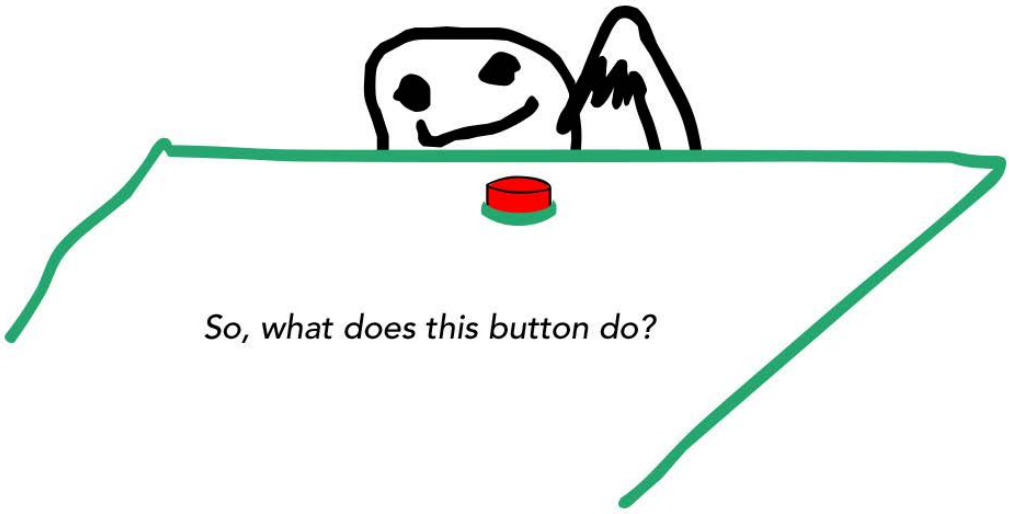
The summer internship is a tradition that dates a long way back in RomSoft. Partly due to our long term partnership with technical universities in Iasi, partly due to our commitment to find the best available talent at all times, we've managed to run more than a dozen successful internship programs.

Through the years, I gained a lot of insight on the key things that could make or break the experience of your summer internship – which is nothing else than your first real, paying, software developer job.

Here are some key questions I think you should consider before accepting an internship offer:

Will you work in a real project?

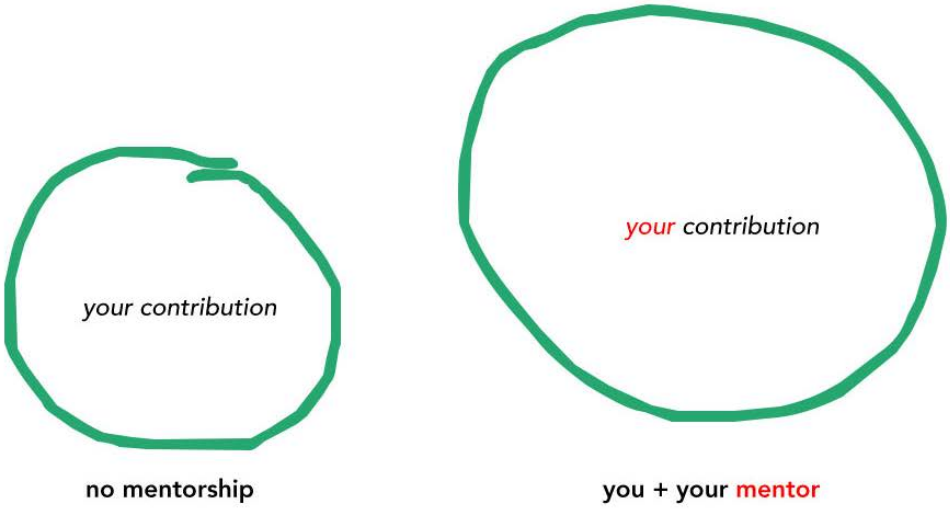
RomSoft internships offer paid professional work experience in a safe and structured environment with help from experts. As a RomSoft intern you will acquire hands-on work experience in one of our current projects. This increases our responsibility to help you do great work, and also helps you gain better understanding of how you can put to practice the things you are learning in school.



Will you have a mentor to guide you?

I think it's important to have a senior colleague guide you through this journey.

Why? When you work on your own, your contribution to the project will be limited to the theoretical skills you acquired in school, or small experience you managed to gather working on individual projects. That is not good enough for you, or us. We need you to hit your full potential as soon as possible, and this is where the mentor steps in.



Will you have access to learning resources?

An important aspect of RomSoft internships is that it offers unlimited access to all kind of learning resources. From books and online courses to in-house trainings, from meetups to one-on-one support. At the same time, a RomSoft internship is a great opportunity to work on your soft skills such as communication, team work and time management.

Is initiative encouraged?

We are a flat hierarchy organization where you can talk to anyone in the company regardless of their title or position. Personal ideas and initiative are fully supported and welcome. If you have an idea for a new project or training subject or think something can be improved in the project you are working on, you will definitely be heard.

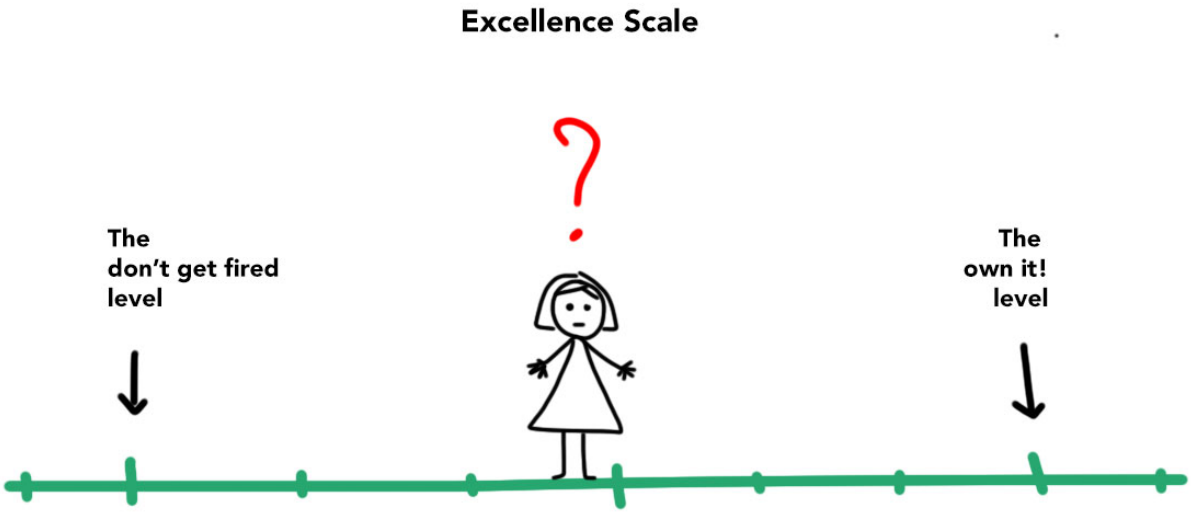
Will this be your career rocket or just a tripping rock?

We are serious (and pretty good, too) about helping you speed up your career. If everything works fine during your internship, we will surely make you an offer to stay. If this will be the case, we encourage you to take this opportunity. You will enter an environment that is characterized by continuous learning, knowledge sharing, and developer autonomy. You will have full support (including financial) to obtain

professional certifications, improve your technical level and work your way up to becoming a master of your craft.

Last but not least, I have a question for you:

How do you picture your future professional self?



The truth is, you'll get hired anyway, rather sooner than later. We are lucky to live in a pretty great moment, when the IT area is growing and there are multiple opportunities for everyone. The market is great, the salaries are in the middle-top category and companies compete one another in offering the most creative benefits to employees.

But you are responsible of designing your own strategy for success. Are you going to simply get hired and maintain the good-old "not get fired" level? Or are you in for something more extraordinary than that?

Corina Enache has a track record of 17 years in discovering the best talent for RomSoft. She has a degree in ancient Greek and Latin, a voice that's soft and warm, and will passionately get involved in solving any grammar controversies.

When you get pitched by your dream company

By Elena Spinachi

If you have skills in the IT area, landing a job may seem like the easiest thing. But what if you're looking for THE JOB?

I was one of those people who were content at their job. I loved the project I was working on, and was part of a great team. But about one year ago I felt the need for a change.

My background

But let me start from the beginning. I am a textiles engineer re-trained as a product owner in the IT sector. My first job experience lasted thirteen continued years and it was the one that paved my way to product ownership. I liked the role so much that I even took my Agile Practitioner, Product Owner and SCRUM Master Certifications to make it official.

At my first job, I had navigated through all the departments and multiple roles, and I have to admit that towards the end of my time there, I was feeling very comfortable. Maybe a bit too comfortable...

And then, the Covid-19 pandemic hit. At least at first, there was this period of uncertainty and anxiety. The company I was working for was providing services for the textiles sector – that was terribly hit – and was less in the position to invest in their research and development departments. At the same time, I was flirting with the idea to manage other projects, to learn new things. I felt I was at a point where I had to continue developing my skills.

Looking for a change

I was looking for a company with genuine product development and a sense of stability. RomSoft was the number one company on my job hunting radar, as it seemed from the outside that it was one of the local companies that fit the profile I was looking for. But at that time, there were no opportunities for me at RomSoft, so I started looking elsewhere.

One multinational company approached me with a proposal for the role of product owner in the outsourcing area, where I had the opportunity to work for a big international brand.

For me it was a good occasion to consolidate my theoretical knowledge in the product ownership area and at the same time it was the perfect opportunity to witness first-hand what it's like to work in an outsourcing company.

In the meantime, as we are fortunate enough to work in a sector where opportunities come in abundance, job offers never ceased to pop-up in my e-mail. I was surprised to see that my skills and

experience were sought by some of the biggest IT companies (both local and multinational). But I decided I was very well where I was, and if I were to make a change again, it would be in the product development area, not outsourcing.

My moment of serendipity

About one year after this change, I received an invitation to discuss a job opportunity from RomSoft. It felt like the universe or fate or whatever you choose to believe in, had decided to throw at me the challenge I was looking for. A little bit later than expected, but better late than never, right?

So I pressed the YES button the very next instant, as my research had already been done.

Why was I set on RomSoft

I have to admit that I had done a bit of stalking through the LinkedIn profiles of people who were working at RomSoft. One thing that I that interested me was how much time they had since they were working there. And a lot of people had between 5 and 15 years with the company, which was quite impressing. For me, this spelled stability. After all, we are humans, and we can't deliver our maximum potential anywhere, before checking those basic, security needs first.

More than that, I've had the experience of a workplace with a high personnel fluctuation and it was mind-wrecking. I wouldn't like to go through that again.

What I also liked at RomSoft was that they had a product development approach. As I was saying, I had done my research long before receiving this job offer. So I knew bits and pieces about the medical purposed software products developed here, and even came across some beautiful visuals created with Office Timeline – the flagship product of the customer that I'm currently working with.

Lastly, I liked how they were proactive when the Covid-19 pandemic hit, and how they tweaked one of their products (EMIM) to help people find their medicine at the nearest pharmacy.

How it feels being on the inside

I don't want to embellish things. I would just say that my expectations were outpaced. Here's what I found on the other side of the fence:

(1) A smooth onboarding, with lots of help from the other members of the team. I felt like I belonged here from day one. I took over my BA/PO tasks at my own pace, without any pressure, which helped me understand my role and become better at what I do.

(2) In the product interaction area, things were exactly as they were supposed to be. I can honestly say it's the first time that I feel that indeed, I am doing product ownership. The workflow and processes feel so familiar and so close to what I've been learning and preparing for all these years.

(3) The relationship with the client is awesome. I admit that I was coming from an area where the client relationship was more rigid and formal. And I kind of expected the same thing here. But that was far from reality. In our project at least, this relation is very cordial, almost family-like, and it helped me relax and concentrate solely on the best product outcome.

A message to the non-IT graduate out there

I think that if you want to become a business analyst or product owner, a predisposition towards research and analytic thinking is very much needed. You have to be curious and proactive.

But also, I think a freshly graduate person would find the domain very challenging. You need to have a certain maturity (not only in the work field, but also biological). You need to understand people, products, and technologies. You need to have very good interpersonal skills and a bit of self-confidence. And in this case I think that it's best if you see it as a process, and work on your theoretical knowledge as you go through more roles and departments before putting your product ownership hat on.

Closing thoughts

There are all kinds of opportunities in the IT industry. You don't have to be a software developer to fulfill a key role. But you do need a lot of work and perseverance. Knowing what you want – and working towards it – will open the right opportunities.

The reason I took on the challenge to speak about this is to share some insights on how my career roadmap took form and the mind-set that guided me through my journey. If this can inspire other non-technical people to pursue a career in IT, I would see as mission accomplished.

Elena Spinachi is a Product Owner/Business Analyst at RomSoft. She is certified as an Agile Practitioner as well as a Scrum Master and Product Owner.

A case of knowledge sharing

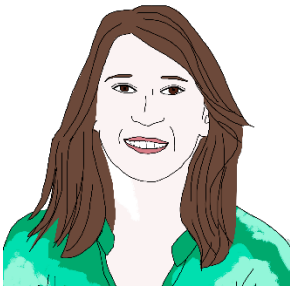
Interview by Iulia Weingold

Of the many initiatives that our colleagues get involved with at RomSoft, I chose to bring forward the Object Oriented Programming study group. It was formed to help RomSoft testers write their own automated tests and not depend so much on the developers in their daily work. They used to meet every week in one of our colorful meeting rooms and dive into the secrets of object oriented programming. That was before the work from home days.

To find out the "whats", the "hows" and the "whys", I talked to Paula, the driving force behind the OOP study group and to Stefan, the trainer for this group. They shared with us the challenges and benefits of group learning, each from their own, personal perspective.

Paula Alixandrov: "Every meeting is like a two-hour brainstorming"

Paula is a dedicated software tester who's been working at RomSoft since 2013. When she is not testing software applications, she works on improving her communication and leadership abilities, and she won a prize in a Toastmasters competition back in 2019.



Who had the initiative of the OOP Study Group?

As software testers, it never hurts to know a bit of programming. Moreover, it is required if you want to do automation testing.

The fun thing is that the company encourages a couple of hours of study time per week, which means that our study group would cover this, and give us the opportunity to really learn something that has true added value for us.

Why did you feel the need to organize this study group?

We used to organize some meetings with the testing community in Iasi but at some point things dragged and we stopped organizing new editions. Also, in our organization, we started to approach software testing differently. We were assigned per project and stopped working as an independent team. There are pros and cons to this, of course, but one of the cons is that we kind of got disconnected from the other testers in the company.

So we felt the need to get together sometimes, in formal or less formal environments, to see how we can keep each other up to date and how we can help each other. The first idea was to meet and learn Selenium. But things didn't really take off.

Also, the biggest challenge we have in our work is that we kind of depend on the programmers to help us "tame" the code. In some projects we use testing frameworks written by colleague developers – but they are dedicated to the specifics of that project. And even with such a framework, we would still need developers to write helpers, so that we can easier access functionalities, get access to ID's, etc. And the problem is that their time is also limited.

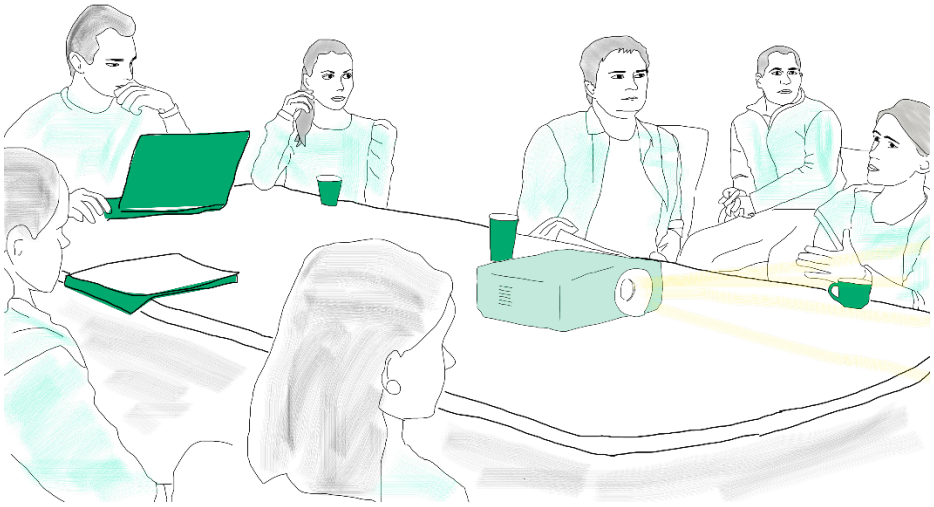
I think every software tester secretly wants (or should want) to know how to write automated tests. So that we don't depend too much on the developers.

The next time we met, we dropped the idea to learn Selenium (you can learn that online, anyway) and decided to find somebody in the company who could teach us basic programming, so we can do automated testing independently from the developers.

How did you pick the trainer?

We needed to bring along a C# developer, because most of the applications are written in C#. And this is where Stefan enters the scene, with the comment that he expected it to be a 2-3 meetings engagement, but things turned out a bit different.

The first challenge for us was to synchronize our paces because we were really novices about programming and sometimes Stefan needed to go over things more than once. And a secret I learned is to get over the prejudices, and ask a lot of questions, even for the things that didn't seem important.



How many people participate to a study group usually?

Usually, about eight people show up to every meeting.

I noticed that people prefer to move their other priorities around in order to participate, and I guess that proves the big added value they feel they are receiving from the group.

What do you actually study?

We study C# basic programming and how to write automated test cases.

After three or four meetings where we discussed theoretical things, we decided we needed to choose a project in the organization where testing is already fully automated. The project is called Office Timeline.

Do you also need to study individually? Do you have homework?

At the moment, during our meetings, we are writing automated test cases for Office Timeline, where we all have access through GIT (a continuous integration tool) and where we can check-in our work, and also see the tests written by our colleagues.

At some point we will have homework, but we still need to go through all the basics.

What programming skills does a software tester need?

A software tester should have basic programming knowledge: how to write a method, a class, a function, how to call them, access identifiers…

What is the coolest thing you've learned?

For example, in one meeting, although I had a different opinion, I decided to hear Stefan's solution for a specific test all way through. And I was surprised by his way of thinking and the solution he thought of. I think it is very interesting for a tester to find out a developer's point of view. And of course, it goes both ways.

What's next?

We want to write a testing framework, it's actually in works.

What are your benefits as a team from these meetings?

It feels good to know that we have support to learn and develop new skills. That we can look beyond the daily tasks, and that initiative is welcome. Also, this is good training, every meeting is like a two-hour brainstorming where we can see how each of us sees the process of building a test, and ultimately, how different points of view can lead to valid results. Or at least useful learning experiences.

Stefan Dascălu: "This group is helping me learn how to teach"

Next, I talked to Stefan, the trainer of the OOP study group, as I wanted to see how things look like from his perspective. Stefan a dedicated software engineer, always inclined to land a hand to colleagues or teams who need his expertise.

What are the technical and non-technical challenges you had to deal with in this program?

The technical challenges are minor, like the fact that some of the functionalities Selenium offers are usually not presented by most online courses you can find. You end up looking stuff up and find a whole new set of functionalities that you didn't think the software was capable of.

The non-technical challenges, I would say, refer to getting the right balance between what I want to teach them and what they have to do during a normal work day.

In a software project, the tester should bring a perspective that is different from that of the programmer's.

My purpose in the end is that they learn as much as possible from a developer's point of view, but not affect the way they see things from a tester's point of view.

What should software testers learn in this program?

They should gain general OOP knowledge. The goal is that, at the end of this program, they are able to write their own tests, to isolate code areas that could be used for multiple tests, in order to avoid duplicates, in other words to make their work easier.

Of course, problems will still arise. If it's a known problem, they can access it through the GIT sources, if it's a new problem, they can always ask for the help of a developer.

For the web part, in the end, they should know how to address specificity in CSS and choose the right identifiers so that they don't have to rewrite the test every time the developer makes a change.

How would you describe this entire experience?

Coming from a longer period of having to deal exclusively with developers, I have the tendency to speed up, and speak fast, and not wait for feedback from my interlocutors. This group is helping me learn how to teach.

Closing note

Most of the times, especially in the IT industry, we are educated to be good self-learners, as technologies are rapidly changing and expectations are set really high. Actually, you will see the self-learning ability mentioned in so many job descriptions out there, that it's almost cliché.

But in many instances it's easier to learn from others. And some of us learn a great deal by teaching others. The obvious thing is, we can team up to share our knowledge. And, apart from getting to the technical insights a lot easier, as a bonus, we learn communication, teamwork and empathy.

7 paradigm shifts when becoming a blockchain developer

By Bogdan Mastahac

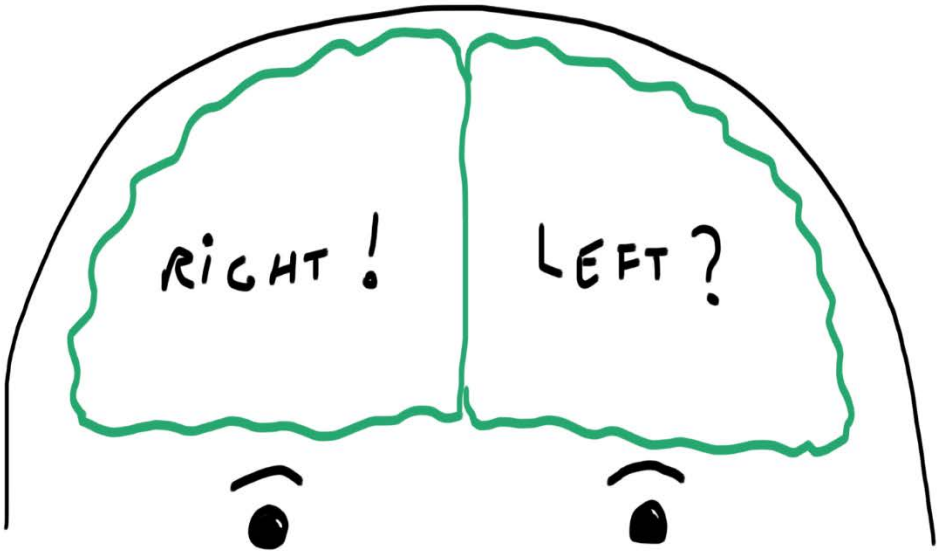
When I embarked in my first Blockchain project as a developer, I had zero knowledge about the technology, and was soon to find out that I was starting from the least favorable position: a software engineer who's been working for more than 13 years exclusively with high level programming languages and Microsoft technologies. But since both computers and developers need a good old-fashioned restart sometimes, I considered the change a welcomed one.

Where do you start?

I didn't have any in-depth knowledge about Blockchain or cryptography when I started out, so I just begun by covering the basics. What is Blockchain? What is a distributed system? What is immutability? And so on.

Even if you have the knowledge of the concepts, it's best to ensure that how they are applied corresponds to your understanding of them. Just because you are familiar with them and how they are applied in some domains that you are familiar with, doesn't always translate the same in other areas or technologies.

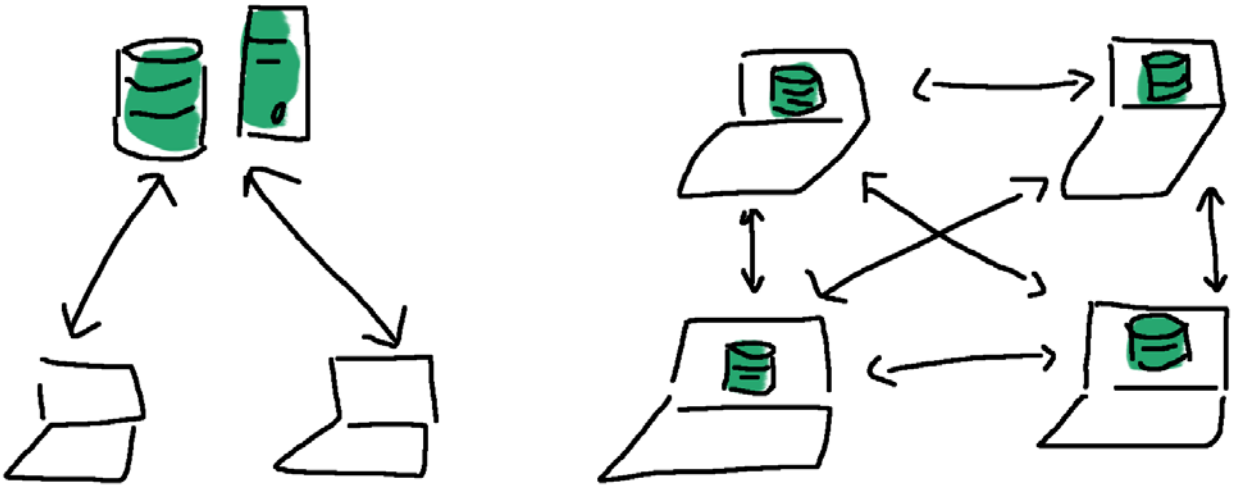
From this first reading I extracted a first pile of unknown concepts that led me to the second layer of learning. It was quite the journey down the rabbit whole, and not because of the big learning load. But because I soon realized that I had to change the way I saw and did things for so many years. **This, I think, is the hardest part: to know what you don't know.**



Here are some of the key perspective turns that I had to take in order to be able to complete my first task in a Blockchain based project.

#1 Blockchain architecture is different

When you contribute to develop an application on blockchain and you consult an architecture document, you should know that this is not the standard client – server database architecture that many of us are used to.



As opposed to a centralized client/server architecture used in traditional databases, blockchain nodes are organized in a peer-to-peer network where each computer runs blockchain software and maintains an updated version of the whole blockchain data structure.

This is the first biggest difference that leads to other particularities of the blockchain systems.

#2 There's no such thing as central validation

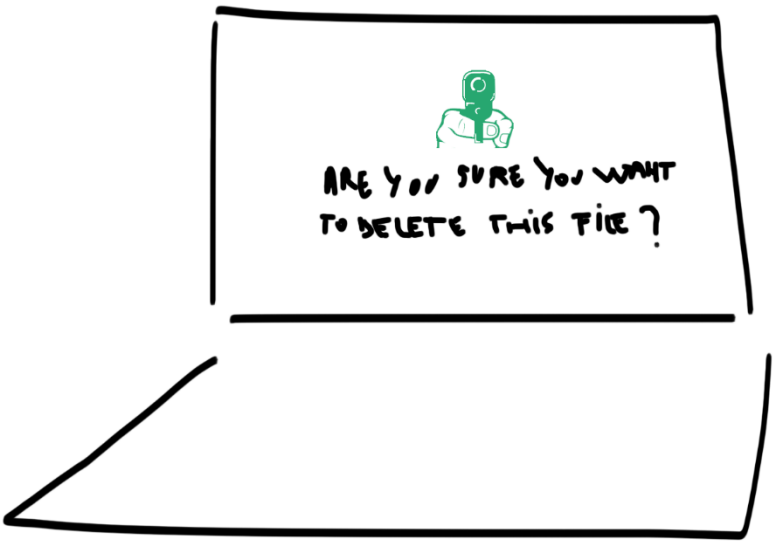
As blockchain is supposed to be an alternative solution to regular central authority-based systems, there can be no central validation either.

One of the key features of the Blockchain is that it is a decentralized system, where the same data structure is distributed throughout every node of the blockchain network.

The system is based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party.

Throughout different blockchain networks, the way to obtain proof is different, but always depends, in one way or another, on the consensus of most of the nodes in the network.

#3 You can never delete any data



In a client-server database for example, you can delete and change data at any time, without any restrictions, considering that you have the necessary permissions.

In a blockchain-based system, this thing is not possible.

Well, in theory, it is possible to delete data from a record of transactions that are part of a blockchain, if two conditions can be simultaneously met:

1) More than half of the network nodes agree with the deletion.

2) All the hashes of the transactions registered after the one that is deleted must be recomputed.

To achieve this, an immense CPU power would be needed which renders the attempt impossible.

Therefore, you cannot say that you can delete data from a blockchain. You can only add to the existing chain, for example you can update the state of a piece of information you initially stored. Depending on the type of blockchain, of architecture, of what the application is doing, you may need to support or make implementations at the smart contracts level.

#4 You must think about the resources you use

Speaking of smart contracts, you may be interested to know that they are also software components, but they are written in languages that are specific to the blockchain distribution you are using.

To create or modify a smart contract you need to learn this specific language, to acknowledge its limitations and what you can do with it. If you are just starting out you need to do a lot of research, as these languages are very low level.

What does low level mean? It means that sometimes you may need to consider the amount of memory that is needed for the operations you are making. How expensive they are in terms of resources, processor, and time. These are usually non-issues when programming in a high-level language.

In C# for example, when you want to perform operations with strings or other data types, you don't need to care about that, as there are many ways to perform them. But in blockchain, all these aspects become important, and in some contexts, exceedingly difficult to execute.

#5 The compatibility issue

Another issue that arises with blockchain-based application development is the compatibility with what you want to integrate.

When you implement a solution in C# for example, you only need to know this one programming language.

But suppose you are implementing a smart contract that performs some operations on the blockchain side, such as some validations and then stores some data in a specific way, and you also need to be able retrieve the data at wish. For this you have to learn the languages that allow you to implement these functionalities.

In my case, for example, in the timeframe I've been working on PharmaLedger, I had to implement APIs or software components in Node.Js, GO, and Solidity. So, you can end up navigating through many programming languages that don't resemble one another.

This requires a greater flexibility on your side as a programmer or system architect.

#6 There's no such thing as support

The blockchain distributions like Bitcoin, Ethereum, Quorum, or Hyperledger Fabric are all open source.

If you've been working for example with Microsoft distributions for most part of your software development activity, when you make the transition to blockchain distributions you'll have to adapt to the fact that there's no central entity to offer support or fix some issues. The community will offer help, or, if something doesn't work, can offer insight on why and what would be the correct approach.

However, the bigger the community that is involved with the distribution, the bigger the chance that documentation is up to date, that coherent examples are available, and that you can find the tools to help you optimize or deploy components.

#7 Deployment is a whole different experience

Finally, a big mind shift for me was with the deployment for blockchain. In a cluster, when you raise a container, that container is based on the image of a machine you are building. On that machine you

specify the operating system, the software products you install, the source code, dependencies, and the program you need to execute. It's like raising a virtual machine. These are built in an optimized mode, with the minimum necessary so that your software component works.

It's not like a Windows machine where you load all possible applications, no matter if you use them or not. It's bare-bone, there's no UI, no mouse. This machine emits only logs. And you have only a command prompt to interact with it. You will not be comfortable to make a deployment to Kubernetes for example, if you are very used to rely on operating with a mouse. You have to adapt to using the keyboard exclusively, to learn how to navigate between folders, to read and search log files without the support of various APIs. The workflow is a lot different from what you previously knew and it takes some getting used to.

To the blockchain beginner

If you want to become a developer in a blockchain based project, it may be easier if you have strong Node.JS knowledge and you are familiar operating on Linux.

But in the end, the technology is less important if your mind is set on it. What I can say now, after going through this process myself, is that, if this transition process does something, is to test your tolerance to change and your capability to adapt to new ways of doing things.

After 13 years working as a .Net full-stack developer, Bogdan Mastahac made the jump to blockchain technologies. He is currently a DevOps engineer and researcher in the PharmaLedger project.

People skills in tech: what if you just haven't got it?

By Claudiu Tescu

Everybody talks about the need of people skills in tech. But what happens if you just haven't got it? Here's an introvert's guide to surviving an environment where it becomes more and more difficult to get by without strong people skills.

A little bit of context

I'm the technical geek type. I started programming at 9 years old. Programming is the only hobby that makes sense to me. I feel like a fish in the water when I find myself in front of a computer screen, solving technical challenges. On the other hand, social activities make me feel awkward. I don't do small talk. I don't tell jokes in front of unfamiliar audiences. And I don't dance.

I love my job. I'm happy to be a full time software developer. But since programming is a hobby for me, that's pretty much all I do in my spare time, too. And all that time invested pays off, I got to be very good at what I do and I gained a lot of experience in over 16 years working for the same company, RomSoft.

To be clear, I'm not the type to aim for a management or leadership position. While there's nothing wrong with that, it wouldn't fit my personality. My problem comes when my knowledge and insights on the company's projects need to be shared. Therefore, I have to interact with many colleagues, to train, to mentor, offer feedback, and help them with various technical challenges.

This is funny, too, because I'm always happy to help, and I feel proud when somebody turns to me for support. But being the introvert geek that I am, I have to admit that social interaction is not exactly my comfort zone.

A story of communication gone wrong

But let me tell you a short story that will help you better understand where I'm coming at.

At some point, an opportunity came to organise some knowledge sharing activities in the company, covering various technical topics. There were, as I remember, two directions to these activities. The first one targeted solutions to complex technical problems that we were facing in our existing projects. Another direction was more focused on technical innovations.

I was more inclined towards the first category, as these type of challenges had the most impact in our day to day work, our productivity and our general performance as a team.

As this program was in need of coordination, I don't remember if there were other people interested in that role, but as I already mentioned, I was really happy to be of help and share from my experience. So I pulled the trigger and took the challenge of managing the program. I really wanted to make it work.

In short, here's what I had to do: scoop for topics to discuss, propose them to colleagues that I saw fit to speak about the respective subjects, plan the activities, get everyone on-board with the schedule, collect feedback, write reports for management, basically everything to keep the activities going and my colleagues interested.

I was happy to get things rolling and that all those complex issues were finally laid out in the open. But I realised that all the coordination work and backstage negotiations were grinding against my enthusiasm.

At some point I had a meltdown and I chose to suspend the activities. That experience took a toll on me for a long period.

The problem with failure is that it is easy to talk about it when you are not directly involved. But when it happens to you, it's a whole different story. It's messy. It lowers your self-esteem, makes you doubt yourself, and downgrades your concentration and productivity. Learning a lesson about failure is a painful process.

So, as much as I love my job, and as much as I know my expertise is appreciated, for a while, I struggled to find my way back to teaching and mentoring activities without having to feel like a weirdo with a communication problem.

How was I going to fix this?

At first, I took things in a very personal manner and I tried everything I could think of to get better at those skills I was lacking. I knew that I wasn't a people's person and people skills didn't come naturally to me. At the same time, I wanted really badly to improve in that area. So, here is a short list of actions that I pursued, in search of people skills nirvana:

- Reading self-help articles and books
- Introspection
- Group trainings
- Individual trainings
- Therapy

After all these efforts, things started to clear up in an unexpected direction. I was actually trying to change my personality. But as much as I got to learn about social skills and communication in the workplace, I also realised that this wouldn't turn me into the outgoing, socially confident person that I was hoping to become.

So that's where the second part of the journey started: an honest reconciliation with my own limits.

Recognising my own limits

A first step was to admit that I'm an introvert with limited capabilities to interact with people who are not my close friends and family, and there's nothing wrong with that. Simply admitting this, felt liberating.

A second step was to identify more precisely the factors and situations that generated the biggest amount of stress for me. Here are a few of them:

Managing a discussion group was just too much for me

While I really believe that a software development company does need a think tank, a group where complex issues are discussed and solutions are sourced, for me, personally, the way I tried to do it at that moment wasn't working. Managing the technical communications activity for the entire company was just a too big hat for me.

At a closer look, I found that the main source of stress was the official part. Scheduling the activities, convincing people to engage and make presentations, writing reports about the activities, gathering feedback, while another person would cruise through them easily, for me, they were a burden. All these things were taking away from the joy of just discussing technical challenges. I would rather have more informal meetings with my colleagues, just plunging into the subject and trying to find actual solutions, rather than having to deal with so many preparations.

I have limited energy for certain type of situations

I could spend hours looking for a technical solution, searching forums, looking into every corner of the Internet for a technical answer. That is my structure. I'm a self-taught person. In other areas, especially where social interaction is needed, my resources are more limited. And it is very important to me to admit these limitations.

Just to give an example, sometimes, in one on one interactions, I need to explain the same thing over and over. It is not a bad thing, and for sure it's nobody's "fault" that this occurs. There are certain technical aspects that are more difficult to grasp. And I know that certain aspects require me to be more creative in finding new ways to explain them.

But the problem is that I'm only "equipped" to do it for so much time. At some point, my empathy and patience shut down and my ego takes over. And I learned to pay more attention to that breaking point. I understood that I needed a safety net for this kind of situations and I kind of developed a strategy of my own.

Socializing at work is not my thing

I am aware that a big part of work communication is socializing. But for me, socialising at work is a source of stress, and not at all productive. While meeting with my friends and spending time with loved ones relaxes me and eliminates the stress that builds up during the day, at work I prefer to keep it professional, to talk about tasks, planning, and projects.

What has worked for me

I decided to make a major change and switch to full time remote work. In March 2020, when the entire world was plunged into telework almost overnight, I already had one year of remote work experience.

I really enjoy the comfort of working from home. It helps a lot that I'm the self-taught and self-organised kind of guy. Taking a break at home, for me is not equal to taking a break at work. Being able to take a short walk or eat some fruit really helps reset my brain.

Yet, I still need to communicate with people. And I still need to grow my people skills. But in my case, communication comes a lot easier online than offline.

As previously said, I'm not the kind of person with endless energy for explaining things over and over. At some point, a little selfish "me" feels the need to wave a white flag. And I learned to recognise that particular moment and try to step away from the "danger zone".

Here are a few strategies I apply in situations like this:

- → Take a step back. Recognise the moment I reach the maximum amount of energy that I'm willing to invest in a particular situation. It is not failure. It is simply admitting what is the maximum I can give there, and that spending more time with that particular problem will only cause me to spiral downward emotionally and in terms of productivity as well.
- → Take a last shot at it, via email communication. Writing a tutorial or a guide sometimes also helps me gain clarity over how some particular technical problems should be communicated.
- → If feedback about somebody's performance at work is involved, ask someone I trust to proofread the message. Given I don't exactly trust my personal communication skills, It sometimes helps to ask for help from another colleague, who is not emotionally invested in that situation. Even just a few tweaks to how a message is formulated can make a huge difference.

While this strategy has worked for me, I know that it may not work for everybody. But for sure, knowing your own personal limits and planning ahead is quite helpful to avoid conflicts and sensible situations.

A new philosophy on communication skills

After the work group "fiasco", I took some time to understand what my sources of discomfort were and what generated them. I realised that I felt a lot better when we organised more ad-hoc meetings. Like, for instance, when a colleague finally solved a problem that we were bumping against on and on. Or when somebody asked for help on a specific topic.

Communication in the workplace is essential. And we can get better at it. In my case, keeping discussions as informal as possible helped a lot. No feedback forms to fill in, no activity reports. Just essential notes.

The next thing that helped was understanding my limitations and accepting them.

Just addressing these two aspects already removed a lot of pressure. Finally, the third thing that did the trick was to build some safety nets I could use in emergency situations.

I know that labelling is a shortcut our brain takes in order to deal with uncertainty. But sometimes, these labels do more harm than good. So I stopped stressing about being a "great communicator" or a "people

person", whatever that means. This experience has helped me know who I am, and who I'm not, in terms of people skills and work communication. And this should be the first rule in the book.

Claudiu Tescu is a Team Leader and a Software Architect in the Extended IPU project. He is passionate to solve complex technical problems, and will always lend a hand when other project teams bump into difficult challenges.

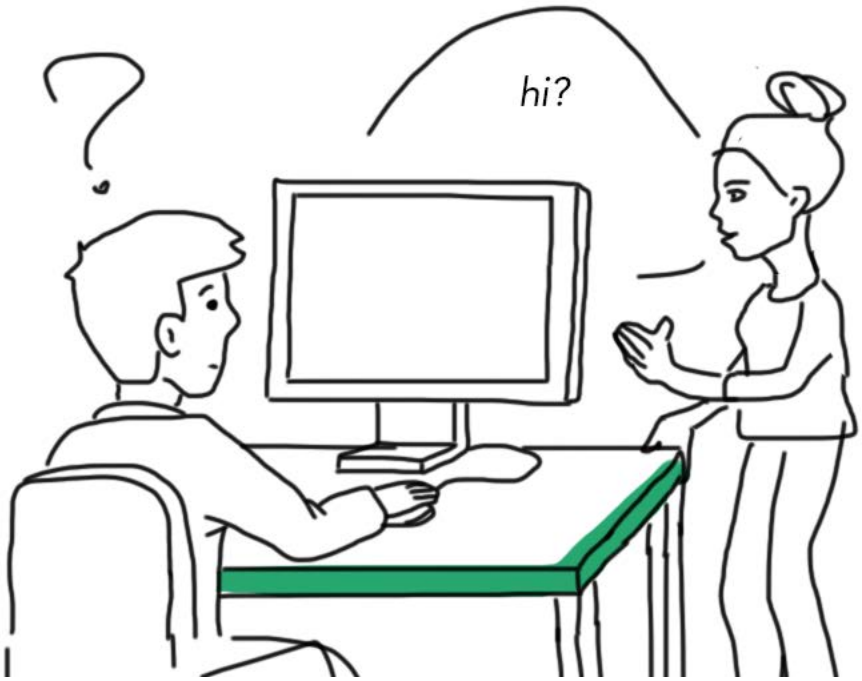
Debunking 7 myths that Tech people believe about storytelling

By Diana Niculaes

In late 2019, we opened a conversation in RomSoft about storytelling, seen as a valuable strategic tool in building better products. So, what's holding people in tech back from using it across the board? Throughout the many workgroups, debates, and chats we've had in these two years about storytelling, I've been carefully curating a fascinating collection of the most common misconceptions that many of us believe when it comes to tech people and their storytelling skills. In this article, I tried to round them up and also fact-checked them – so you don't have to.

Myth #0: Tech people don't give a sh*t about storytelling

But first, I'd like to make a confession. When I and a colleague of mine started talking about storytelling to people at RomSoft, I admit I had my own misconceptions and prejudices. I was convinced that developers prefer machines to people and would never be receptive to storytelling, which is essentially about human connection. I was afraid they would just roll their eyes, let out an exasperated growl, and go back to tapping feverishly on their keyboards.



I was wrong. Many of them were intrigued and open to learn more about storytelling, as long as we wouldn't waste their time and explain clearly why it's useful and how they can apply it. So, really, the first myth about storytelling that got debunked was my own fixed idea that IT people are antisocial and would ignore me or tell me to piss off. They didn't. And somehow along the way, they got comfortable enough to voice their own preconceptions and we worked together to debunk them.

What follows is the result of all these insightful discussions. So, go get your lab suit and put your safety goggles on because we're just about to start busting some serious storytelling myths.

7 false storytelling myths from people in Tech

Myth #1: Storytelling is about making things up. Like fiction, right?

Reality: Well, no, not necessarily. Sure, when we say storytelling, we're firstly thinking about fiction: books, films, or the stories that made us daydream as kids. But the truth is that storytelling is all around us – at school, at work and in all our interactions. *Catch Me If You Can*, *Hidden Figures*, *A Beautiful Mind*, *Wild* are just a few of the many films or books based on true stories.

A good story does not conflict with the truth. And while it's not okay to lie and fabricate stories to deceive people (e.g. false advertising or propaganda), it's definitely helpful to use a story, which can be plausible even if not entirely accurate, to describe a problem or explain a benefit. "*Se non è vero, è ben trovato*" ("even if it is not true, it is well conceived") – where "well-conceived" is key. This is an Italian expression that I learned from a colleague when we were debating this, and it was *ben trovato* indeed.

Myth #2: Stories are for children. I'm an adult, give me facts and data, not stories

Reality: Look, no one is saying it's all about the story and that facts don't matter, because they do. Without the underlying facts, a story is just that, a good story. But what storytelling does so well is making raw data easy to understand and to remember. It makes information sticky.

The purpose of storytelling is to complement the data, not to replace it. Delivering data as a story worked well for us when we were children, but even as we grow into adults, we still need stories to explain, persuade, and inspire. When we wrap them into a story, that's when data, facts and information become even more compelling and memorable.

Myth #3: Storytelling is too egocentric, and I don't like talking about myself

Reality: Oh, it's anything but that. I admit, it may seem that way because during a story, the attention becomes focused on the storyteller like a spotlight on an actor. But storytelling is not about you. It's about making a connection with your audience.

It's a transformative experience that changes both the "story-teller" and the "story-listener", and it drives shared understanding between the two of them. It's all based on how our brains work and I could go on and on about mirror neurons and oxytocin, the bonding hormone – but in the end all you have to do to understand this is remember how you felt when listening to a good story. You felt part of something bigger.

Myth #4: Storytelling is unpredictable, unorganized. I'm not a spontaneous person

Reality: That's fine, no need to change who you are. If there's something I absolutely love about Tech people, it's your logical, organized, process-oriented approach to work (and life). On that note, remember you can tackle storytelling just like any other project: make a plan, break it down into steps, and improve with each iteration.

And although it's a rather creative process, storytelling is still a process after all – it has a clear structure (beginning, middle, and end) and some common elements (e.g., characters, conflict, and resolution). Think of storytelling as an art but also a science, a collection of techniques that you can learn and apply anytime, in a variety of contexts.

Myth #5: I've never had any talent for writing

Reality: Let me guess… growing up, you got caught in the eternal division between science and humanities? I know that many of us were repeatedly told as children that we have to choose: play with cars or with dolls, be good at math or at arts, and work with computers or with people.

And sadly, some of you have been told that a good text has to be stylish, long, and intricate – and that you just don't have the talent for it. If that's what happened to you, I'm sorry for what you had to go through.

Forget all that crap. A good story is not complicated. It follows an age-old formula: the hero has a problem that they need to solve and in doing so they learn a lesson. The simpler, the better. No need to use idioms to beat around the bush (see what I did here?) or stretch it more than necessary.

And as for talent, nobody is born with it, it's just that some people have more practice than others. As you practice, you will get better, too.

Myth #6: I don't have any good stories to tell

Reality: Of course you do! Even if it seems like everything has already been written, I assure you that's not the case. Whenever I hear this myth, I always like to tell my colleagues: "There's not another YOU on this planet". Each of us is a unique combination of experiences. We don't have any clones, not yet at least. And even if we did, there's no way they could replicate your knowledge and skills as well. Or the adorable awkwardness.

Did you learn something the hard way? Things didn't go as planned? There's your story – and it's a damn good one! Nobody is interested in stories about projects that went along perfectly. We want to hear about learning from mistakes, growing after failures, and perseverance despite hardship. We're not perfect and that's okay. But we strive to be better.

Myth #7: I'm not a storyteller

Reality: Yes, you are. We are all storytellers, it is part of being human. We are hardwired to use stories to understand the world around us, share crucial information and create a sense of belonging. It's how we built our society and it's what makes us different from other life forms.

You're already a storyteller, whether you acknowledge it or not. So, the next time you reach out to someone, and you burst out "oh man, let me tell you what happened…" - just remember you're doing something that you're programmed to do. You're telling a story.

Recap

These are just a few of the fascinating conversations we've had around storytelling at RomSoft, but I always feel like we're just scratching the surface and there are still many more things to uncover.

The good news is that in time, more and more colleagues have taken the plunge and started to share some of their stories – on our company blog, in the present e-book, or in our meetings. They have taken the first steps and we can't help but admire their courage.

Diana Niculaes is an experienced digital communication specialist who leads the content strategy for Office Timeline. She enjoys sipping a cup of tea, solving sudokus, and helping Tech people with the tools and the support they need to bring their stories to life.

The story of CodeCamp

Interview by Iulia Weingold

I'm happy to bring forward an inspiring story about knowledge sharing and community building, via Florin Cardasim - Software Architect, community builder at Codecamp, co-founder at Strongbytes, and cherished RomSoft Alumni.

In his relaxed conversational style, he shared valuable insights on how Codecamp grew from 25 to thousands of participants, on what he learned during the pandemic, and how the IT landscape has changed in the last 20 years.

Florin, too many times we tend to use the word "community", but what does it mean? And where do you start building?

I would like to start from the story of how we built the community that is closest to my heart, Codecamp.

I've always had sort of a restlessness wanting to try new things, to meet new people, to learn and to share what I know. 15 years ago, I was working at RomSoft, and the good thing was that I found there this context where I could experiment a lot. For example, I remember the first time .NET and C# technologies came out and I was the first to say that I wanted to do something in that area. I started learning and experimenting with them, and after a while we were able to start a first project based on these technologies.

In the same period I was experimenting with a technology called WCF (Windows Communication Foundation), on one of the projects we were developing for Sysmex. And being a new technology and all, you couldn't find many people who had experience with it. I didn't have a lot of experience either, but I compensated with being a motivated learner. And I was invited to make a speech at this thing called the MSDN RoadShow, one of the few tech conferences that existed at that time. It wasn't my greatest presentation but I enjoyed making it a lot. Then I started to be invited at various Microsoft community

events and that, for me, was the moment when it all clicked together. I started to understand what community means and how it can transform an entire city or region.

I was very much into it. And was lucky to find these two friends, Dan and Gabriel, who were working in other IT companies in Iasi as well. And with the support I found in RomSoft, and along with my colleagues at the time who were also interested to take part in this movement, we did a first Codecamp edition. I remember we were 25 people at Ramada Hotel, and we literally camped there for three days and three nights, doing presentations, workshops and getting high on ideas.

After that, the word spread and we started having more and more participants. So we upgraded to a conference format. And I see that as my personal contribution to the local IT community, as a thank you for all the help and support I received. This community helped me grow, I made a lot of friends and I learned a lot. I had access to information that I wouldn't have dreamed otherwise.

Knowledge sharing and a bunch of other common values with RomSoft, is it not?

I want to believe that in RomSoft, also, I was one of the promotors of the sharing knowledge philosophy. Being few people in the company at that time, I tried to challenge anyone who would listen to be better, to want more. Many times I was exigent, maybe heard a few protests here and there, but I was applying the same scrutiny on myself. And in the end, we came through, as we were also good friends and all the beers and basketball sessions we shared helped as well.

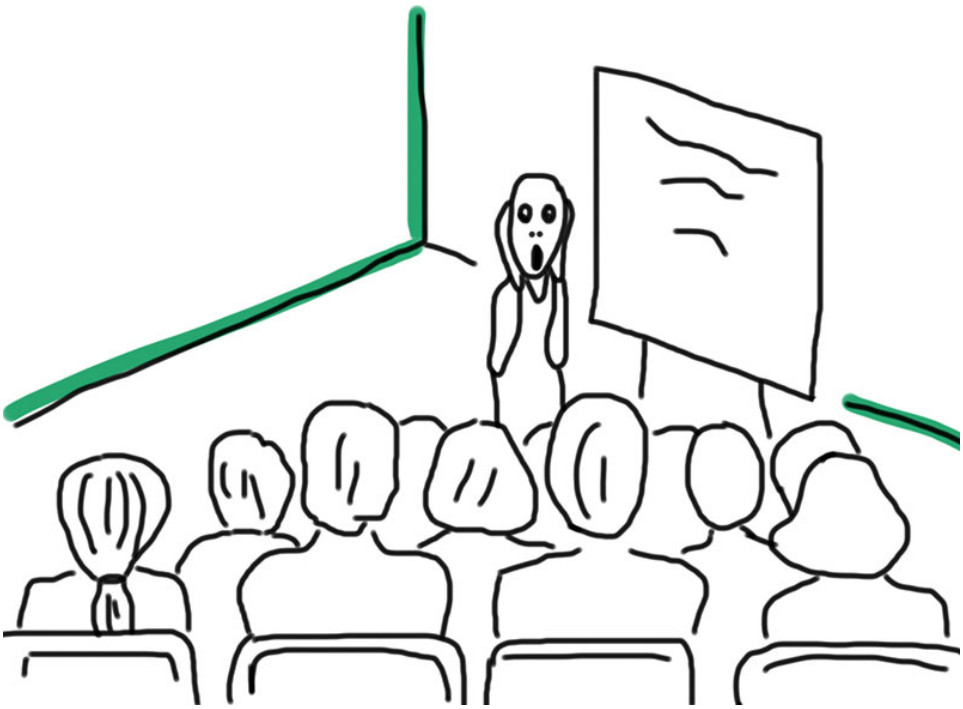
We were doing a lot of presentations in the company, we were doing the so called learning groups. If I was running Codecamp outside of RomSoft, I felt bound to do all sorts of learning activities inside the company, too.

Looking back, I hope that nobody felt I was pushing. Or at least I think it is sometimes useful to have somebody encouraging you to do something that takes you out of your comfort zone. And beyond the pain and the struggle, you may discover that you like the experience. Or maybe not, but it is important to find out.

What is the biggest blocker for somebody to come in front of an audience and make a presentation? It's not about knowledge I guess…

At least that's what I found out for myself, that you can never know everything. And nobody does. And that many of us, willingly or not, live with that impostor syndrome. Especially in our world in tech where things evolve so rapidly, there's no physical time to catch up with everything. And the impostor syndrome can block you, and you have to become aware of it.

But you will always find an audience, or you can choose an audience with less experience, that can benefit from what you know. You can bring value to someone. So it's a process to recalibrate your expectations. This will also reduce the anxiety. Then you have to be aware that you are scared. If you want to do this, you have to wonder "why do you have to?", and "why do you want to?"



Only after you answer these questions and the answer is different from “no, you definitely do not want to do it”, then, you must realize that this anxiety, this fear of speaking in front of others, is like any other phobia. It can only be “treated” with exposure. And what you also need to know is that the most important thing in these interactions is preparation.

I remember when we were preparing the first Codecamp editions. One or two months ahead we started preparations and rehearsals, we took turns in helping each other with our presentations. Often times, we stayed up until 2 a.m. in the RomSoft meeting rooms.

So being there a long time, in the trenches, having been through all the sweats and pains, I realized that I can use this experience to teach public speaking, in a manner that I think is as friendly and as human as possible. And I think I can unblock some reticence in this area.

I remember you encouraging me to do a presentation on photography. I was really surprised as I didn’t believe the subject fit the interests of an IT company.

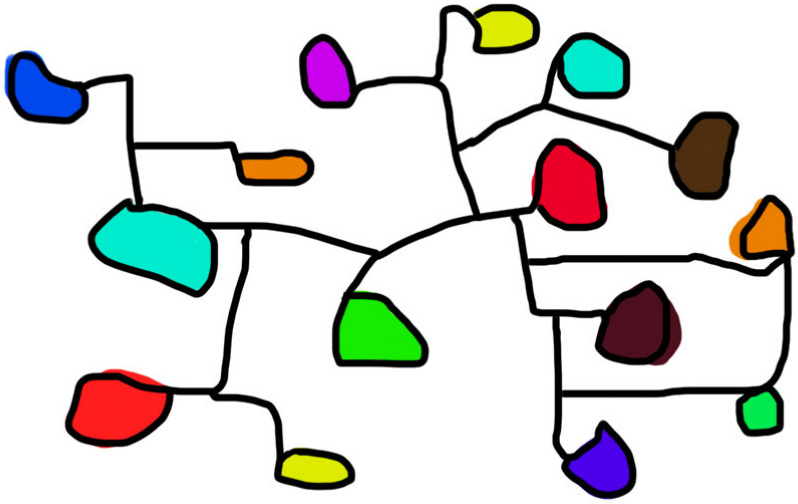
I always thought that, whatever you do for a living, you have to fit that activity into your life as a whole. You cannot separate your professional life 100% from who you are in your day to day life. I don’t wake up in the morning thinking OK, this is when I go to the office and submit a number of bytes and code lines and then 5 p.m. comes along and I get completely disconnected. I need a larger context where I have space to evolve, to learn new things, to be exposed to new ideas. I may learn something from you while I’m teaching you something. You have to keep an open mind.

And people picked up on this approach, and many of them got involved. And the best thing was that after they did their training in this micro-community that was very familiar and friendly in the company, they took the step outside, to present at Codecamp or other events and achieve their next growth level.

Let's say I want to build this kind of micro-community, where people feel more at ease to express themselves and share what they think is important for our common growth. Where do I start?

Do you know what I think is most important? A thing that I also applied with my colleagues in RomSoft and elsewhere?

You need to keep running at all times a few small engines, or energy sources, that can engage others. The more the better. They can come up spontaneously, but if not, you have to find them and prepare them.



That means to come out of your shell and talk to people. To engage in conversations, to convince them to get involved. To build from within, with people who are interested in doing this. It is important to have these multiple engines, because sometimes people get tired or encounter a problem and have to concentrate their attention elsewhere. And the other ones will take over and fill in and take the movement forward.

This is a minimalist structure that can unblock energies. And yes, people need to be supported, trained, and mentored.

As a motivation, the biggest advantage in this way of doing things is that it is an amazing learning source. You know how it is: you go to a presentation and maybe go home with 10% of the information. Then, you go to a workshop and your impact grows to 50-60%. But when you come in front of your peers and you want to make a presentation, you must go in 90% for sure. You have to go deep into the subject, you have to understand it and be able to verbalize it. When you put on this hat and you realize that you can't explain a certain subject, this means you didn't understand it yourself well enough, and you need to dig deeper, to double check, to do more research.

How did Codecamp grow from 25 to thousands participants?

For me, community is an open organism. It shouldn't belong to one company, it has to belong to everybody, or it won't grow. And then, slowly, we tried to see how we could partner more companies and reach more people. In RomSoft we always found an open door. We had the liberty to experiment, we were gathering 5-10 people to rehearse our presentations.

One thing you should know, whoever is a speaker is first and foremost a person like you and me. Nothing is achieved without effort and hard work. So we put in the work and the hours. There were no gurus or pros among us. Sometimes we started to prepare a month or two in advance. And we tried to repeat the same schedule to more companies, more diverse. It was quite the roadshow. We went everywhere people let us in. Sure, not everybody did, and it's normal. But RomSoft did, and many others, too.

In 14-15 years we held 100 big conferences. They grew as the industry got bigger. In Iasi, we had up to 2000 participants per edition, and then we extended to Timisoara, Cluj, Bucharest, Bacau, Suceava, Baia Mare, and Chisinau (Moldova).

And now? How did you adapt to the "new normal"?

Now, with the pandemic crisis, everything is online, and already, in the last months we held up to 20 conferences, with international participation. We had to narrow down the content, because it is difficult to manage more than 10 different topics in 2 days of online conferences. For live conferences you can go deeper. For example we had 10 to 12 conference rooms where we could organize separate tracks for different interest areas like .NET, Java, AI, down to the soft skills and project management zone, the entire IT related spectrum of learning.

Another thing that we lose in online is the ability to bring speakers of all magnitudes. We have to concentrate on high-level speakers over beginner speakers from the local community, which is a great loss, from my perspective. We hope to be able to win back some field in this area when we'll go into a hybrid format, whenever that will be possible. Because this is what made us unique, in the end, the ability to help people from our community to grow this particular set of skills.

What have the last 18 months taught you?

I learned a lot in this period. In March 2020, when the first wave hit, we had the plan outlined for the entire year. After the lockdown period we had to change perspective. In about 2 weeks we regrouped, and I can honestly say we barely dodged bankruptcy. As much as we are not here for profit, we still need to be organized as a company, with balance sheets and all. And we were at risk. Everything was blocked. Our system is that participation is free, but we sustain ourselves through our sponsors. For about 6 months, we had to make it with the reserve fund we had from the previous years. And since autumn 2020, little by little, things started to pick up. Now, after 18 months, I can honestly say that we are doing fine.

There were conferences where we had 1000 people participating. Others have two or three hundreds. We have to admit that in online there is an abundance of content and there's a certain psychical fatigue. The engagement is certainly different when you know that in 2 weeks there's this big event in your community and everybody will be there.

What are your comeback plans? Is it worth making them?

We have comeback plans for sure, but we have to wait and see how the situation evolves, because we've made plans before and had to cancel them. In the optimist scenario, next year, May 19th we want to organize a Codecamp Festival in Bucharest, in order to allow access for more people, from all over the country and the world.

Depending on the success of the festival, we will then see how we continue. But still, I think the future format will be hybrid. Throughout the year we will have online conferences and one or two live, itinerant, summits or festivals.

How hard was it to stick with such a project throughout the almost 15 years?

For me, the Codecamp project was very natural. I cannot think if it was hard or not. For me and the friends who I started this thing with, this was our way of living and learning. It complemented very well our professional lives. Through Codecamp I have access to some things that, for sure, if I stayed in my place, isolated, I wouldn't have heard of. It leveraged my knowledge and my ability to ask for help where I needed it. At any time I could pick up the phone and ask someone "how do I do this, can you help me"? And instead of hitting my head against the wall for several days, somebody came in and said "this is the problem, and this is how it's solved".

How do you see the IT community in Iasi now, as opposed to 20 years ago?

We are in a developing stage, I'd say. It has grown a lot, for sure. The context is different. 20 years ago I was staying in line for a job interview with 50 other people. Now we have this fantastic luxury that 50 recruiters from different companies want to talk to us. In some way we are in a maturing process. We need to relearn, step by step, what the correct ratio between money and value is. But this is also related to this fantastic pressure on the talent market, where there's little we can do, we just have to face it.

What is clear is that the professional level has grown a lot, with the growing industry, and with the competition. But coming from this community area, I'm more into cooperation and that is also the idea with which we joined the Imago-Mol cluster, with Strongbytes. I honestly believe that we can become stronger together. And this allowed me personally, once again, to reconnect with RomSoft, this time, from a partnership level. I've always recognized RomSoft as a breeding ground for entrepreneurship.

And this is the idea that I try to seed wherever I go, in other companies where I have the opportunity to talk to people. That in a community there should be no territorialism. Sure, there's a balance that you can keep, but community is that neutral space where you should feel safe, where you are not at war with anybody. There, you only contribute. Give and you shall receive. From this point of view, Iasi is very bubbly, companies get involved, and we've gone a long way from the times when it was hard work to explain to IT managers in Iasi that they should be present, that they should get involved and contribute to the community.

Thank you so much, Florin! I hope the tips and tricks that you've so kindly shared with us will inspire some more of us to make the step forward and do something for the communities we work and live in. And a big cheer from the entire RomSoft crew, we can't wait for more future projects together!

Thank you! It was a very pleasant throwback, and I'll use this opportunity to wish my former RomSoft colleagues a very happy 20th anniversary this year!

The story of Pricy

By Vlad Balan

A few weeks ago I was inspired by my friends at RomSoft to share the story of Pricy, a personal project that I and my best friend, Sorin, have put a lot of effort and enthusiasm into. So here it is. May you have only rewarding shopping experiences from now on, with a little help from your faithful shopping assistant Pricy!

My history with RomSoft

Right out of university I was recruited by RomSoft. I worked there for a number of years and this was the time that shaped me as a developer. At RomSoft I met my best friend, Sorin – who also became my partner in the project that I want to tell you about.

Later on, life took us both to a different place – Ireland to be more specific. But on and off I continued to collaborate with RomSoft, and together with Sorin, we continued to work on this darling project of ours.

The idea kicks in

So there I was working at RomSoft, in my first years as a software developer, navigating through tasks and projects and learning as much as I could on the side. I remember some pretty intense study groups that I and several colleagues were organizing back then. We used to study everything we thought interesting. From WCF, a technology that I don't use anymore, to .NET, C# or various soft skills that I still use every day in my profession.

Long story short – everything built up to that turning point in every programmer's life, when one decides one needs to buy their first dishwashing machine.

So I started browsing online shops in search for a perfect product at a perfect price. That was somewhere back in 2012.

Now, I'm a developer, so my life is comprised of two types of problems: ones that can be solved with a script, and ones that can't. This one fell in the first category, so that is what I did.

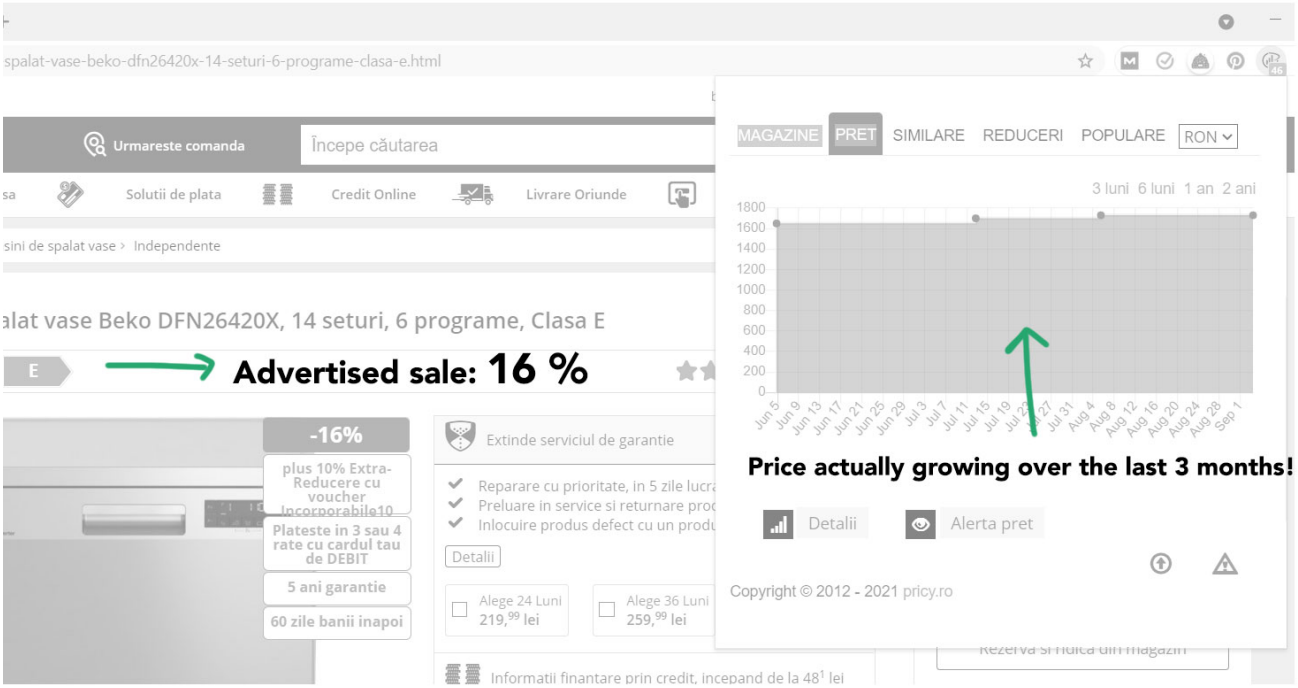
I wrote a script in Google Sheets that retrieved prices on various dishwasher models I was interested in, and I watched them shifting for a month. The surprise was that these prices were constantly changing, in a dance that looked more like a ping-pong ball trajectory and far from any correlation to their advertised offers and discounts.

I thought this data was worth further analysis and exposure. So I talked to Sorin and we enhanced the script to retrieve prices across more product categories and more retailers. Then, we built a Chrome extension – which we called Pricy – so that we could display this data in a more visually appealing form.

During that time, we started a small research among our friends and colleagues, to see if they could see any value in this type of tracking. The feedback was positive and it encouraged us to continue. We kept adding retailers and developing the extension. Today we analyze data from over 500 Romanian retailers, we have over 60,000 users and even became self-sustained due to our referral revenues.

How Pricy works

Let's say you want to buy a product and you just clicked on a commercial advertising a 16% price off on a specific website. If you have the Pricy extension installed, you can see how the price evolved in that specific shop in the last three to 24 months.



In the example above you can see Pricy actually showing a slightly growing price in the last 3 months, which contradicts the advertised sale of 16%.

Now that you've seen for yourself that sales are not always what they appear to be, Pricy can help you further, to find the best price for the desired product, or, if you can afford to wait, to set an alert for the desired price.

The perfect tool to navigate a Romanian black Friday weekend

First of all, you may ask, why is the Romanian Black Friday so special?

While this US tradition was cheerfully imported by the Romanian retailers no matter their size or their trade, there are some aspects that make it an "authentic" experience.

In Romania, black Friday is usually celebrated a week before December 1st so that delivery companies have enough time to send packages before the Romanian national holiday.

But many retailers, both online and offline, extend their "Black Friday" events to several days, sometimes a whole week.

At a closer look, you can find many discount opportunities all year round, with even better prices than those practiced on Black Friday, and this makes the retail market look like a continued discount festival or a never-ending Black Friday experience.

On the other end, some black Friday "offers" come right after a major price increase, making it pointless to wait on your purchase for this one-day-in-the-year-opportunity, that, on top of everything, comes with customers flooding websites and leading them to crash, products rapidly flowing out of stock and at least a couple days delay in delivery.

I hope Pricy can be a useful assistant for you and your friends in search of the perfect product at the perfect price. If you ask me, I'd rather use it all year round, and not only throughout a particular weekend, be it Black Friday or not.

Did I find the perfect dishwashing machine?

The dishwasher I bought back then has been used for about… four months. We took it to Dublin with us, when we moved. Sorin has it now, but he doesn't use it too much, either.

Vlad Balan is co-founder at Pricy.ro, Vice President of R&D at Mi9 Retail, and proud member of the RomSoft alumni team.

A greetings card from Canada

By Catalin Sandu

Every important story in my working life has had two beginnings. In RomSoft, I started, as many of my colleagues, coming over from the same company where we had previously worked together. So I basically knew everybody. We were already friends and it wasn't difficult at all to jump right into the projects.

I was involved in developing some very interesting applications for Sysmex Europe and Add-On Products. As some people in RomSoft probably remember, my style was to always season our long hours of hard work with a bit of fun. Because there was plenty of work.

I always enjoyed our trips to the customer sites in Germany, where we got the chance to get feedback directly from our customers and even from the lab workers who interacted with the Sysmex analysers and therefore, with our software. Even after all these years I still keep in touch with some of the customers we worked for.

I even have a funny little story from one of these visits. I remember the looks our clients from Sysmex Hamburg made when I declined to drive a Mercedes they made available for me to get around during my stay. In all honesty, I didn't refuse because I was picky, but at that time I didn't have a driving license. They would probably be surprised to find out that even after all these years I still don't have one. On second thought, had it been a Ferrari or a Porsche, I'm sure I would have jumped right at it. Even without a driver's license. Vrrooom... vrrrooooooommmm...

I'm sure that I would have continued working at RomSoft for many years, because I'm the steady type of guy. But it has been my life's dream to move to Canada, and I got to do so 14 years ago.

My story here also has two beginnings. At first, I worked for a small company, maybe similar to RomSoft 20 years ago. The second part started when we were acquired by the company I currently work for, TELUS. In a short time I had gone from having a handful colleagues to 50,000.

But what I realized during all these years is that being a developer in Canada is almost the same as in Romania or elsewhere. It's a pretty competitive field where most people prefer to move fast from one company to the next. I guess I'm the exception that confirms the rule.

I do miss my colleagues in RomSoft, though, and I'm honored I was asked to write a story for this anniversary e-book. I was happy to find out that the drawings I used to make for my teammates' birthdays or for our advertising brochures are still remembered by some of them.

Even though in the meantime I switched from cartoon drawing to professional photography, I still want to send you all an anniversary card, together with my best wishes and warm thanks for the opportunities you've opened for me 20 years ago. So here it is:

Note: We took the opportunity to turn Catalin's drawing into our very first RomSoft NFT

Catalin Sandu is Development Team Lead at TELUS Health and longtime RomSoft alumni. He's always been this curious mix between a tech geek and a visual artist, going with ease from .NET and Visual C# to drawing and photography.

We’d like to thank our customers, our teammates and former colleagues who chose to share their stories and contribute to this e-book.

We’d love to continue these conversations on https://www.rms.ro/, so don’t be a stranger and pay us a visit.