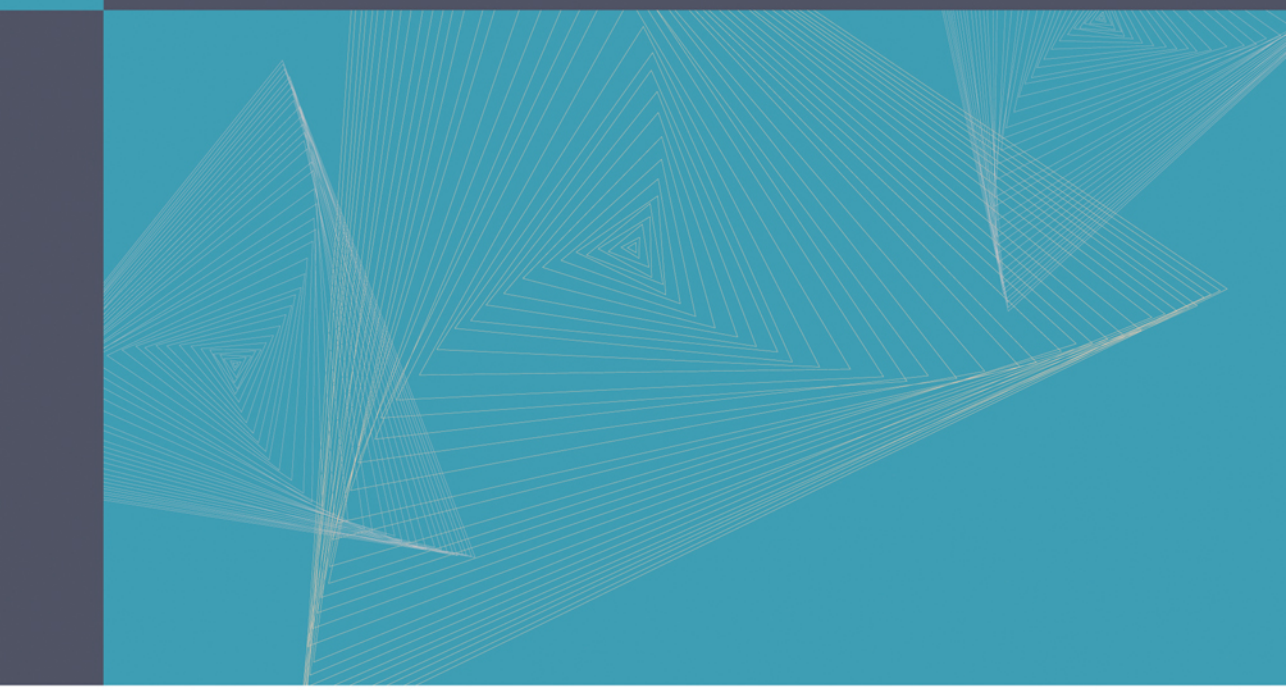
# DEVOPS

## A BUSINESS PERSPECTIVE



Oleg Skrynnik

Van Haren PUBLISHING

## Other publications by Van Haren Publishing

Van Haren Publishing (VHP) specializes in titles on Best Practices, methods and standards within four domains:

- IT and IT Management
- Architecture (Enterprise and IT)
- Business Management and
- Project Management

Van Haren Publishing is also publishing on behalf of leading organizations and companies: ASLBiSL Foundation, BRMI, CA, Centre Henri Tudor, Gaming Works, IACCM, IAOP, IFDC, Innovation Value Institute, IPMA-NL, ITSqc, NAF, KNVI, PMI-NL, PON, The Open Group, The SOX Institute.

Topics are (per domain):

### IT and IT Management

ABC of ICT
ASL®
CATS CM®
CMMI®
COBIT®
e-CF
ISO/IEC 20000
ISO/IEC 27001/27002
ISPL
IT4IT®
IT-CMF™
IT Service CMM
ITIL®
MOF
MSF
SABSA
SAF
SIAM™
TRIM
VeriSM™

### Enterprise Architecture

ArchiMate®
GEA®
Novius Architectuur
Methode
TOGAF®

### Business Management

*BABOK® Guide*
BiSL® and BiSL® Next
BRMBOK™
BTF
EFQM
eSCM
IACCM
ISA-95
ISO 9000/9001
OPBOK
SixSigma
SOX
SqEME®

### Project Management

A4-Projectmanagement
DSDM/Atern
ICB / NCB
ISO 21500
MINCE®
M_o_R®
MSP®
P3O®
*PMBOK® Guide*
Praxis®
PRINCE2®

For the latest information on VHP publications, visit our website: www.vanharen.net.

# DevOps

## A Business Perspective

Oleg Skrynnik

Van Haren PUBLISHING

# Colophon


| | |
|---|---|
| Title: | DevOps – A Business Perspective |
| Author: | Oleg Skrynnik |
| Text editor: | Roman Jouravlev |
| English translation: | Oleksandra Spiegler |
| Illustrations: | Oleg Skrynnik |
| Publisher: | Van Haren Publishing, 's-Hertogenbosch, www.vanharen.net |
| | |
| ISBN Hard copy: | 978 94 018 0372 4 |
| ISBN eBook: | 978 94 018 0373 1 |
| ISBN ePub: | 978 94 018 0374 8 |
| Edition: | First edition, first impression, December 2018 |
| | |
| Lay-out and DTP: | Coco Bookmedia, Amersfoort – NL |
| Copyright: | © Van Haren Publishing |

This book is a translation from original Russian version: DevOps для ИТ-менеджеров: Концентрированное структурированное изложение передовых идей,
ISBN 978 5 00006 016 2

# Foreword

This book is written by an IT manager for IT specialists, IT managers and IT executives. It does not show DevOps as a phenomenon associated with new automation tools, programming techniques or technologies; it explains the management aspects of DevOps for those who are professionally engaged in information and technology *management*.

It differs from other books by the structural nature of the narrative (perhaps, excessively structured) and by the attempt to cover fully a phenomenon of DevOps at a basic, fundamental level. This does not mean that the narrative is superficial, sufficient just for creating awareness of the new subject area. 'Fundamental level' means building the foundation, the basics: I'm talking about the origins of DevOps; the inevitability of its emergence; the key prerequisites and their reflection in practices; about the practices themselves and the principles they are based upon.

Despite the abundance of literature on this subject, this is the very book I really missed when I myself studied DevOps. I aim to provide a clear, structured and concise review of this complex yet very interesting subject. I dare to hope that there are no superfluous words in this book, and on the contrary, all the necessary words are here.

I have to express my sincere thanks to my family and friends. I cannot say they helped me to write this book: fortunately, they have very little to do with such matters as DevOps. However, they definitely suffered from it: quite often, from July until December 2017, I went incommunicado and failed to react to their signals; sometimes I even demanded silence in evenings.

I also have to thank my colleagues at Cleverics. We established this business together with the brightest people I ever met, and it happened to be one of the most important decisions of my life. Common goals and principles; freedom in decision making; responsibility for the outcomes; and partners ready to support me when it is needed — without this I would not find time to structure my thinking on DevOps and to transform it into this book.

Finally, I thank our clients: they keep offering us new and exciting problems to solve; new challenges. They keep demanding new training, workshops and simulations; they want more and better… They literally do not allow us to stand still and constantly make us moving forward.

The author, Moscow, Autumn 2018

# About this book

This book is the core literature of the EXIN DevOps Foundation certification. This exam tests the understanding of basic DevOps concepts and how they relate to each other, as well as the value of DevOps for the business. EXIN DevOps Foundation is the first level of the EXIN DevOps certification program. The EXIN DevOps Professional certification tests the knowledge of DevOps practices and how to integrate teams. The EXIN DevOps Master certification is about promoting organizational change and leading the way towards continuous delivery and improvement.

# Acknowledgements

"I thought I knew a bit about DevOps these days, but I learned a lot from this book. It's in a narrative style, which I like. It comes from the perspective of legacy enterprise 'horses' moving to new ways of working, rather than a development technical focus. It hits all the points I would hit if I had ever gotten around to write such a book. I won't now, as Oleg has made such a good job of it. Well done!"

Rob England

*"DevOps - A Business Perspective"* is a well-written and carefully-curated summary of the key DevOps topics that IT managers should be aware of. It combines impartiality with astute and useful personal observations – the author clearly knows his stuff. I expect to be consulting it from time to time and I have no hesitation in recommending it."

Mark Smalley

"An easy to read and well thought out and constructed overview of the history of Devops in terms of practices and the accompanying technology. It offers some great summaries, poses dilemmas that organizations will face and gives some good examples on making choices for moving forward, at the same time signaling potential barriers and pitfalls to avoid. If you want a DevOps 101 this is it."

Paul Wilkinson

# Contents

# 1 What is DevOps?

Methods of IT management do not stand still. Approaches to the development and operation of information systems nowadays are different from those several decades ago. Moreover, tomorrow will be the time of the next generation of refined methods and techniques, which will be based on new knowledge, experience and technology. Most of the time, management methods evolve gradually, by means of systematizing and honing of the models created earlier, based on certain basic principles and postulates. However, from time to time, discontinuities occur, allowing individual leader organizations to make a significant step forward with regards to effective and efficient use of information technology.

A good example is the transition of IT management from focus on IT systems to managing IT services. Having started around the year 2000, this change in the view of management enabled pioneers to gain significant competitive advantages. Successfully adopted by the leaders, emerging management practices became so-called best practices; and some of the best practices evolved further to generally accepted good practices, and even contributed to industry standards. Of course, some organizations did not use the best practices or standards in their work: not all spheres of economy were significantly IT-dependent in those days.

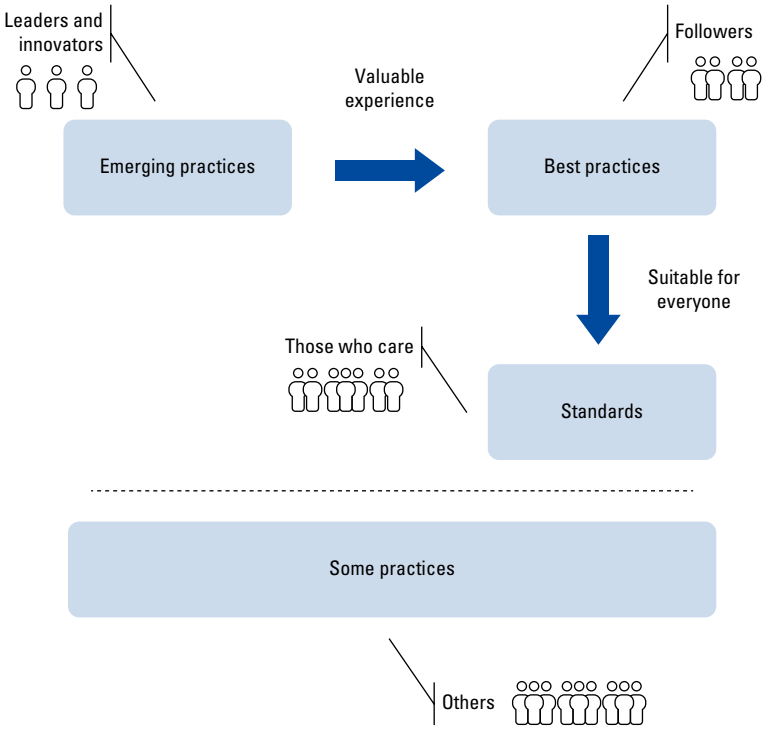



Fig. 1.1 Emergence and use of new practices

Let us look at IT service management, for example. In the 1980s, the idea to provide value from information technology in the form of services and to organize IT activities in the form of processes arose. Certain European companies became pioneers, developing new practices in organizing work and approaches to solving management problems. Some of the practices, such as introduction of a Service Desk; distinction between incidents and problems; managed and controlled processing of IT infrastructure changes, etc., were formulated in 2000-2001 in key publications such as ITIL® (it used to stand for *IT Infrastructure library* in those days)[1]. This allowed them to move into the category of best practices, and not only leading organizations, but also the 'followers' started using them. Eventually, in the year 2002 BS 15000-1:2002, the first standard for IT service management was published, which established a certain norm to be followed by those who seek to build a coherent IT service management system. That said, practices, publications and standards do not stop developing:

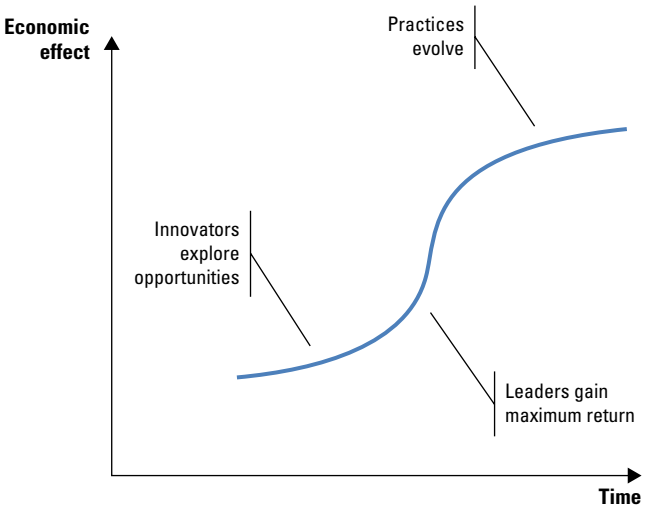



Fig. 1.2 Development of practices

Similar dynamics can be observed now in Agile software development. However, the revolution that is brewing here affects a larger area than software development alone and the scale of the consequences may be on the same level as that of ITSM.

New, emerging practices have been labelled 'DevOps' (Development + Operations), which is as far from the intended meaning as ITIL® is far from the concept of 'library', and today's COBIT from control objectives.

1 https://www.axelos.com/best-practice-solutions/itil

While publishing COBIT 5 in 2012, the copyright holder pointed out that, even though originally COBIT was an abbreviation of 'Control Objectives for Information and Related Technology', now it is a just proper name[2].

ITIL® custodian since 2013, AXELOS Limited has made similar comments about ITIL®.

DevOps experts, who were the originators of this movement, acknowledge the limited nature of the name, calling to use more accurate in their opinion 'BizDevOps', 'DevSecOps' and the like. However, the probability of changing the name is now insignificant.

So, the DevOps phenomenon is worth studying. To understand fully the essence of DevOps, it is necessary to consider the background of both the idea and the movement associated with it.

## 1.1 Origins

One could argue that DevOps appeared due to two factors: wide adoption of agile software development methods and of management of IT infrastructure as a program code. Let's look at each of them.

### 1.1.1 Agile methods for software development

At the end of the 20th century, the dominant methodology of software development was the so-called 'waterfall model': sequential execution of predetermined stages, each of which takes significant time and ends with the achievement of previously agreed results; transition to the next stage in many cases occurs only after the previous stage is fully and formally completed. An additional distinguishing feature of this model is the functional specialization of the people involved at each stage: analysts, architects, developers, testers, and so on.

When developing large information systems of pre-defined functionality and with no or limited requirements for fast delivery of the product, this model enables creation of high-quality products, combined with effective and detailed cost control.

However, at the end of the 1990s, with the rapid growth of Internet technologies and web programming, downsides of the waterfall model started to affect interaction and understanding between information systems customers (internal or external business) and providers (internal or external software developers). Indeed, emerging market opportunities available for business customers required rapid launches (within a few months) of new products to the market. However, a typical development cycle from the

2 http://www.isaca.org/COBIT/Pages/FAQs.aspx

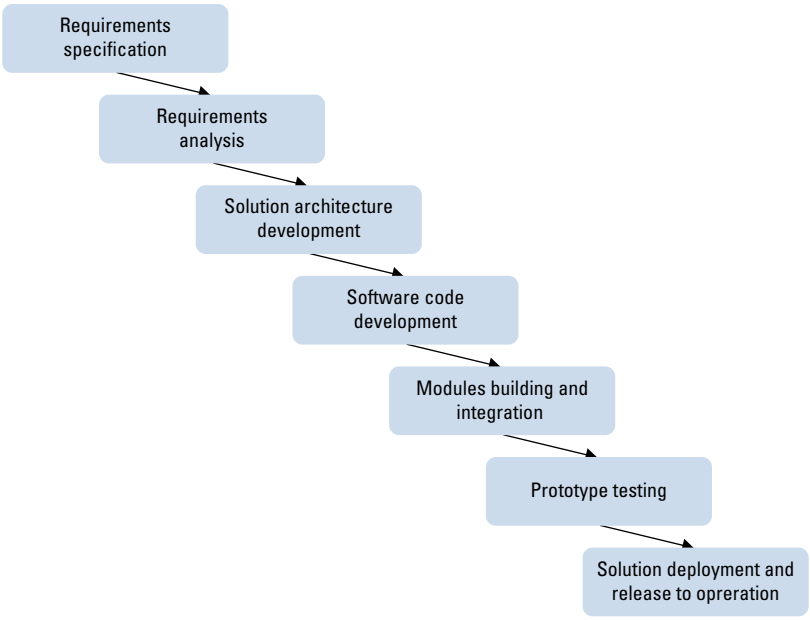



Fig. 1.3 An example of a waterfall software development model

beginning of the project to the first working prototype could take from six to 18 months; up to 2-3 years in larger enterprises. In addition, with the emergence of previously unknown but potentially promising market opportunities, customer requirements could change in the course of the development, which was extremely difficult to take into account without extending the deadlines, or reducing the quality of the product.

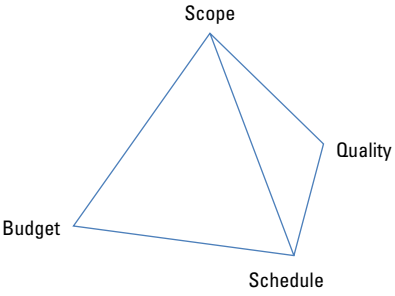



Fig. 1.4 Classical pyramid of the project management constraints

Thus, tension was building up between customers and providers; between the core business and software developers. Innovative approaches to programming were the answer to this challenge. Ken Schwaber published several books about Scrum[3]. Kent Beck published a book on extreme programming, or XP[4]. However, the effect of the application

3 For example: Schwaber, K., *Agile Software Development with Scrum*, 2001, ISBN: 978-0130676344
4 Beck, K., *Extreme Programming Explained: Embrace Change*, 1999, ISBN 978-0201616415; 2nd edition, 2004, 978-0134052021

of these new ideas was moderate, mainly because it was focused on just one of the stages of the software development cycle — the actual programming, while the problem was wider. The end-to-end software development cycle needed to be simplified and speeded up.

In 2001, Schwaber and Beck, along with fifteen other experts, met up to discuss the existing problems and to work out a solution. The outcome of the meeting was the so-called Agile Manifesto. It was designed to bridge the gap between business and software developers. One of the manifesto's authors, Robert C. Martin, explains[5]:

*'Trust between developers and business can emerge and develop when the right disciplines and the right minimum process are used. Business will start to trust the developers, instead of thinking that they are lazy, corrupt, nasty creatures, and the developers will start to pay attention to business and realize that they are reasonable and rational beings, rather than someone from another planet.'*

The subsequent developing and adoption of agile methods by the community of programmers and project managers greatly accelerated and restructured software development.

### Agile Manifesto[6]

We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to 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 |

That is, while there is value in the items on the right side, we value the items on the left more.

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

5 https://www.youtube.com/watch?v=hG4LH6P8Syk, also https://www.aaron-gray.com/a-criticism-of-scrum/
6 http://agilemanifesto.org/iso/en/manifesto.html

4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Agile processes promote sustainable development.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done — is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The key elements of agile development are: closer interaction between the customer and the developer, reduction of the batch size, products delivered at short intervals (cycles) and limited size of the teams.

Using an agile approach, the software development team releases a new viable product every two to four weeks. End users are closely involved in the development, thus ensuring fast feedback, which, in turn, inspires faster changes.

However, in many companies, abandoning the waterfall model in favour of agile development, effect was smaller than expected. Failure to benefit from agile observed in many companies, often has little to do with advantages of the waterfall model or the disadvantages of agile. The problem roots in the fact that development of the code is only one of the links in a long value chain.

Indeed, prior to the development there is still a significant group of steps aimed at identifying business needs, their elaboration, analysis, prioritization, and so on.

Furthermore, after development, applications need to be quickly deployed in the production environment, so that the customers received all the benefits they had been promised, and could provide feedback to the developers. However, IT infrastructure of almost every organization established before 2010 is based on rigid, expensive hardware procured a long time ago; budgets for it were obtained with great difficulty and the budgeting process for new procurement is lengthy.

Moreover, this infrastructure is in a rather fragile state in a large number of organizations. One of the factors contributing to such fragility is that the IT solutions used are extremely complex. There are many thousands of interconnected items in the infrastructure. Another

contributor is the lack of IT systems documentation, as well as the rapid obsolescence of the documentation. The latter is continually enhanced by the loss of knowledge due to the staff turnover.

In many organizations it is unsafe to touch IT infrastructure. Change is the biggest evil for IT operations department, and a constant large flow of changes may lead to truly catastrophic consequences.

Thus, advanced methods of software development are held up by obstacles on the IT operations side, which decreases the possible positive effect of applying agile approaches.

To deal with IT infrastructure fragility, some organizations use formalized and automated change management process designed to structure the flow of changes and minimize the risks associated with their implementation,

### 1.1.2 Managing infrastructure as code

The emergence of management of IT infrastructure as code was preceded by development of two technologies: virtualization and cloud computing.

The history of virtualization of software and hardware environments began quite a long time ago, in 1964, with the beginning of the development of the IBM CP-40 operating system[7]. During the years of consistent development of this area, considerable progress has been made. First commercially available systems for mainframes appeared in the 1970s, and those for subsequently more common machines based on the Intel x86 architecture appeared in the 1980s[8]. The chart below shows the number of key events related to virtualization between 1964 and 2008 (the graph does not stop at this year by accident, as you will see further):

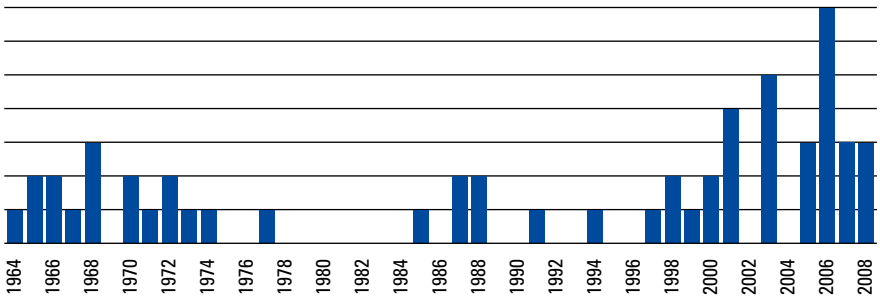



Fig. 1.5 Key virtualization events distributed in time

7 https://en.wikipedia.org/wiki/IBM_CP-40

8 It is interesting to note that, according to Jez Humble, in those years during a certain period of time IBM avoided recommending virtualization products to its customers, as this affected the sales of the hardware.

Virtualization made it possible not only to use expensive and powerful hardware more efficiently, but also to introduce an additional level of abstraction between the executable code that provides something useful to the customer and the underlying system software. A significant step was taken in the direction of separating the competences and responsibilities of, so to speak, 'application engineers' and 'system engineers', in the broad sense of these concepts.

Cloud computing technology developed even faster. Until the middle of the 1990s, telecommunication companies offered their customers the Wide Area Network (WAN) service by connecting the relevant endpoints, for each customer with direct cabling. However, with the emergence of private virtual network technology (VPN, Virtual Private Network), it became possible to send data packets of different clients via the same data transmission channels, providing the necessary level of security, privacy and quality of service. At that time providers started to use the cloud symbol to show the border between the client's private network and the shared network, and the respective separation of responsibility.

With the new capability of transferring data over long distances, customers started using these technologies not only for the information exchange between their remote systems, but also for distributing the computational load between different nodes of their networks. The emergence of a technology to simplify and cheapen this interaction was prompted. Small providers took the first steps, but truly significant changes happened in 2006, when Amazon presented ECC (Elastic Compute Cloud). Soon, in 2008, Microsoft launched its service, Azure, and Google introduced the Google App Engine, subsequently evolved into the Google Cloud Platform. Of course, these are not the only examples of renting out the computing capacity, but they are the largest ones.

Virtualization and cloud technologies have significantly changed the computing landscape. Resources offered by commercial providers have become affordable and reliable; they also assured the necessary level of security. The customers' attitude to the clouds and their use has changed from *"someone else is controlling my hardware somewhere"* to *"I have an infrastructure that I manage remotely"*.

The US National Institute of Standards and Technology identified five essential characteristics of cloud computing[9]:

1. On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

9 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

2. Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms.
3. Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
4. Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
5. Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service.

What does managing the infrastructure remotely mean? Let's recall one of the key paradigms of UNIX-systems: all the necessary actions with the system should be accessible from the command line (hence, using a script). Graphical user interfaces are a beautiful, but optional.

Now, let us combine the virtual cloud technologies and the command line interface for all tasks. As a result, IT professionals could create the parts of the IT infrastructure they needed, including servers, storage systems, and network components, and all interfaces between them, all settings and configurations by means of the of text commands... The degree of automation has increased significantly, and so has the speed of the change implementation. Previously, to deploy an IT infrastructure based on in-house hardware, it was required:
- to justify and agree a budget (weeks and months);
- to wait for the next purchase cycle (months);
- to order equipment from the supplier and pay for it (days);
- to wait for delivery (weeks and months);
- to receive, install, configure, prepare for use (days and weeks).

Today it is possible to create a similar IT infrastructure by:
- running a script, waiting for completion of its execution (minutes, rarely hours);
- clearing the invoice from the cloud provider at the end of the month.

That is, the required infrastructure is created using the program code. It not only is created, but also can be managed as a program code: with version control, change tracking, debugging, reusing previous versions and so on. These aspects will be discussed in more detail Chapter 3 *The Principles*.