RISC Software GmbH
www.risc-software.at

INSIGHTS[1]

The technical magazine of RISC Software GmbH
on current research and development topics

Agile Software Development

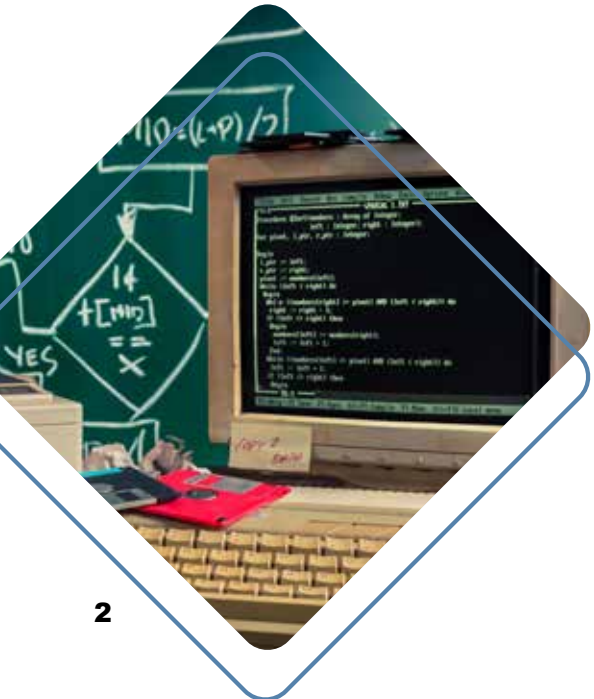
Software Reengineering

Data Science and Prescriptive Analytics

CONTENT

Agile Software Development

Data Science and Prescriptive Analytics



Software Reengineering

Dear Reader,

The magazine you are currently holding is the first issue of our collected technical papers. Our experts offer you both general introductions to topics and profound information on the areas of optimization, simulation, data science and prescriptive analytics, as well as software reengineering and agile software development. Most of what you are reading here is know-how which was built up over many years and had been developed in research projects, paired with the latest scientific methods successfully implemented in customer projects. These successes are not based on the personal commitment of our employees. What distinguishes us is our ability to solve problems together with our customers in a way that, beyond digitization, we gain a decisive competitive advantage and develop a good partnership on the long run, too.

Digital transformation affects all departments and organizational areas and requires a high degree of change, often a cultural change, too. We perceive ourselves as enablers and therefore we bring comprehensive multidisciplinary competencies to the table. We provide the experts with whose experience and domain knowledge this process is driven forward. This is a huge advantage for our customers, since the technological challenges are becoming even more various due to the increasing dynamics and complexity.

RISC Software GmbH is not only an experienced partner for research projects and long-term collaborations, it is also a knowledge mediator. Within the framework of our AI Academy, we offer workshops on topics such as agile project management, artificial intelligence and much more, also according to specific customer requirements. We support customers in their development of specific know-how related to digitalization and new technologies.

No matter where you are - whether you want to digitize complex processes, bring old inventory software up to date, use your manufacturing data for forecasting or anywhere else - our team of experts supports you in the implementation of your R&D projects with their different expertise!

Enjoy reading,

Wolfgang Freiseisen
CEO Software GmbH

Agile vs. Classic Software Development

– DI (FH) Andreas Lettner
Head of Domain-specific Applications Unit and Head of Coaches

The right process model for your project

Classic approaches such as waterfall, V-model or spiral model or agile approaches such as Scrum or Kanban? At the beginning of a project, the question arises as to which of the existing process models is best suited for the current project. RISC Software GmbH has many years of experience with various process models and advises its customers at the beginning of a project which process model is recommended.

Changes

Classical process models follow a linear approach, where a defined final state is worked towards from the beginning. Here, the project passes various phases in sequential order. The phases are usually completed by milestones. Due to rigid transitions and sequence of phases, the introduction of new requirements or a change of requirements into a project are avoided, if possible. This is usually not compatible with the with budget and time goals.

Agile approaches rely on an iterative-incremental model instead of this sequential progression. Iterative means that product development happens in cycles, while incremental means that each cycle a potentially usable product increment is produced. As a result, necessary deviations from the plan can be identified at an early stage.

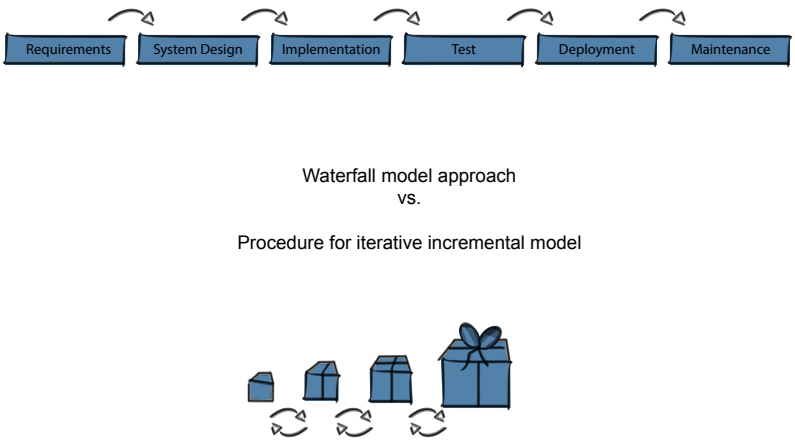


Figure 1: Waterfall model versus iterative incremental model

Feedback

Recognizing the need for specific changes is a challenge for any project team. In classic process models, this happens in best case at the milestones, i.e., between two phases - in worst case at the end of the project. In agile approaches, each iteration includes timely feedback from customers, which prevents the final product from not even meeting the market requirements.

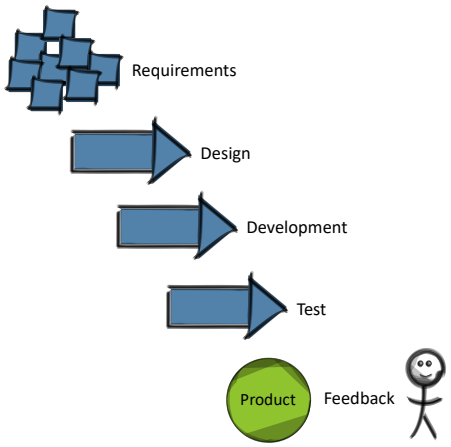


Figure 2: From request to feedback

Complexity

Projects fail because of the misjudgement of their complexity. The definition of complexity according to the Cynefin framework results from the fact that the relationship between cause and effect cannot be predicted in advance. Complex systems require ermergent practices to achieve a product goal. This contrasts with complicated systems where - with the appropriate knowledge base - the relationship between cause and effect is predictable. Thus, complicated systems or products can certainly be planned in advance. Agile approaches are based in their principles on emergent product development and promote the reduction of complexity through their iterative-incremental practices.

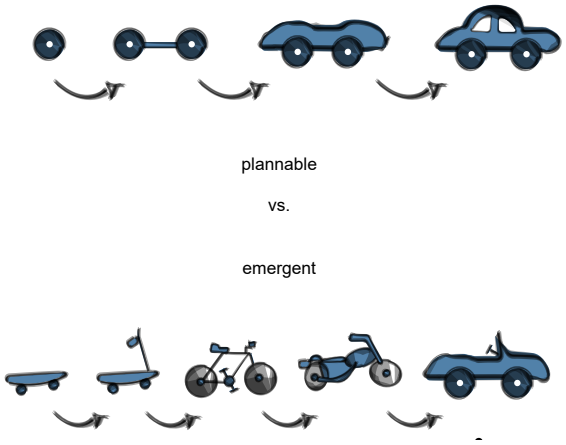


Figure 3: Planned versus emergent product development

Conclusion

Both approaches - the classic and the agile - have their respective advantages. Ultimately, the assessment of which process model is the most suitable is based on the experience of both partners - the client and the implementation partners. Nevertheless, the following basic rules can be used to make a decision:

Projects that can be realized with small teams in a short period of time and are within a clearly defined context may be suitable for classical process models. Even with a classic process models, RISC Software GmbH works according to agile values and usually supplements the project processes with agile practices. Above all, active and regular communication with customers is a policy that is never neglected.

Agile practices are recommended for all other projects. Steering product development and reducing complexity are elementary principles of agile values, and even long-term projects with multiple teams can be coordinated through scaling techniques. A team of specialists at RISC Software GmbH advises its customers individually on the possible process models as early as the project initiation phase and also continuously monitors the results during project implementation in order to be able to make any necessary adjustments to the process model with the customers in good time. ◆

Trust in Artificial Intelligence

– Christina Hess, MSc
Data Scientist, Logistics Informatics Unit

How we create and use trustworthy AI systems

Artificial intelligence (AI) already supports us in our everyday lives, consciously or unconsciously, in many areas. We act with automated assistants such as voice assistants or the parking aid in our cars, use facial recognition to unlock our smartphones, and let music and movies be suggested to us. AI is already very present in our everyday lives - perhaps even more than we realize. The number of AI applications will continue to grow in the future. As we create and use AI systems in the future, we need to be aware not only of the opportunities, but also the risks and challenges of using them. Guidelines can help us create AI systems that are not only efficient, but also ethical, safe, and trustworthy.

AI on the rise

The advantages and areas of application of AI are multifaceted. It is easier to understand with a few simple examples: Recommendation systems try to learn our tastes from our previous choices and then suggest content we like. Streaming providers like Spotify or Netflix have been relying on this kind of AI for years. Natural Language Processing (NLP) methods help us understand or generate texts. This makes it possible, for example, to automatically translate texts in such high quality that these texts are often indistinguishable from texts written by humans.

In industry, machines can be maintained at an early stage through the application of AI, thereby reducing or completely preventing failures. Such applications are referred to as predictive maintenance. In general, predictive analytics can be used to predict machine failures, the quality of manufactured products, or even orders or order quantities. Prescriptive analytics - a current field of research as an extension of predictive analytics - attempts to make optimal decisions on the basis of predictions, causal relationships and domain knowledge.

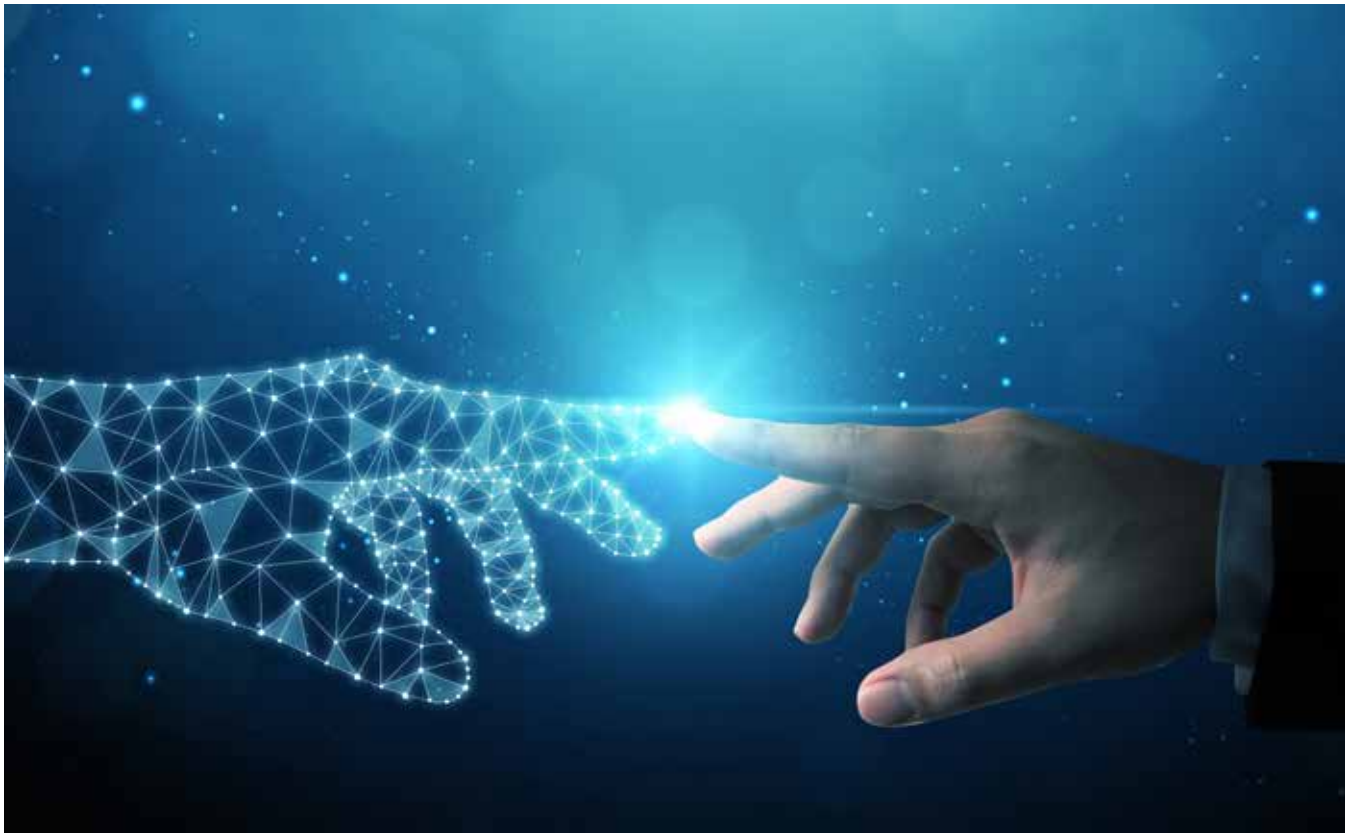
In medicine, AI systems can support medical professionals in both diagnosis and treatment. For example, tumors or diseases can be detected automatically on the basis of image data, diagnoses can be suggested on the basis of patient data and symptoms, and disease progression can be analyzed and predicted.

AI systems in cars enable automatic traffic sign recognition, help to keep in lane and even enable autonomous driving.

Current challenges and problems

However, the use of AI systems also brings some disadvantages and risks with it. Systems that are not fully developed or have not been tested enough can be prone to errors. If the AI used by the streaming provider for user profiling is not good, you as a user will get suggested content that does not interest you or that you do not want to watch. If qualities or outages are predicted incorrectly, it can lead to outages, lost revenue and increased costs. When AI is applied in more sensitive areas, there can be even more tragic negative consequences. Consider, for example, an autonomous driving vehicle whose AI is poor at distinguishing people from objects. Then, in the worst case scenario, the vehicle will not avoid the person and hit the trash can, but the other way around. In addition to the possible susceptibility of AI systems to errors, other aspects also need to be considered. Currently, ethical aspects, data protection and the transparency of AI systems are the main topics of discussion.

In addition to the possible susceptibility of AI systems to errors, other aspects also need to be considered. Currently, ethical aspects, data protection and the transparency of AI systems are the main topics of discussion. Errors and inaccuracies in the data used, the implementation or the interpretation of the results can lead to people or groups being discriminated against, to them being restricted in their ability to make decisions or to the system causing harm in some other way. A well-known example is the following: Bias (distortions) in the data causes models trained on the data to also reflect that bias. A really good model eventually learns to decide as humans would. If the data on which the model is based are racist, i.e., if these data represent racist decisions made by humans, the model will also make racist decisions. A really good mod-



el like the one we would want as a society - one that is secure and trustworthy - would not be able to meet the challenges presented in the data or provide us with a way to prevent it.

In addition to data quality, the interpretability of AI models is often a major challenge. In black box models, the decision of the AI system cannot be understood or can only be understood very imprecisely. However, this would be extremely important in terms of transparency and trustworthiness of the system. Imagine an AI deciding whether to get a loan or not. In case of rejection, you now want to know why you were assessed the way you were. You even have the right to know why an AI made a decision about you. Here, a major current problem becomes apparent: If you want to be able to understand the decisions of an AI, you cannot use models whose results cannot be explained completely - even if these models are better in their assessment than others.

Regardless of whether an AI makes good decisions about you and whether they can be interpreted: Maybe as a human you just don't want to be judged by an AI. A topic often discussed within this context is the so-called social profiling. To use the example of lending again as an illustration: If an AI automatically divides applicants into cohorts, you may be denied a loan based on your age or origin, but given one to comparable candidates. Such ethical challenges are being addressed by various institutions, research organizations and NGOs. Amnesty International, for example, provides examples of how AI systems can pose a threat to our fundamental rights.

Guidelines for creating trustworthy AI systems

The previously mentioned questions are being addressed by different organizations/institutions and from different directions. In order to be able to develop AI systems in such a way that they are robust and transparent and do not pose a threat to us, but rather support us and help us wherever and whenever we as humans reach our limits, various guidelines have already been developed. The goal here is that we as a society can trust AI and its decisions. A well-known example is the Ethical Guidelines for Trustworthy AI published by the EU in April 2019. According to these, AI systems should be developed and deployed in compliance with various ethical principles. In addition to fairness and harm prevention (AI systems should not cause harm or have a negative impact in any other way), the ethical principles of respect for human autonomy and explainability are described in the context of AI systems. Explainability in this context includes not only the comprehensibility of the generated results (keyword: black box models), but also transparency with respect to all processes, the capabilities and the purpose of the AI system. To preserve human autonomy, human supervision and control over the AI system and all processes have to be ensured. Accordingly, AI systems should not subordinate, enforce, condition or do anything similar against humans but AI should support and complement human capabilities.

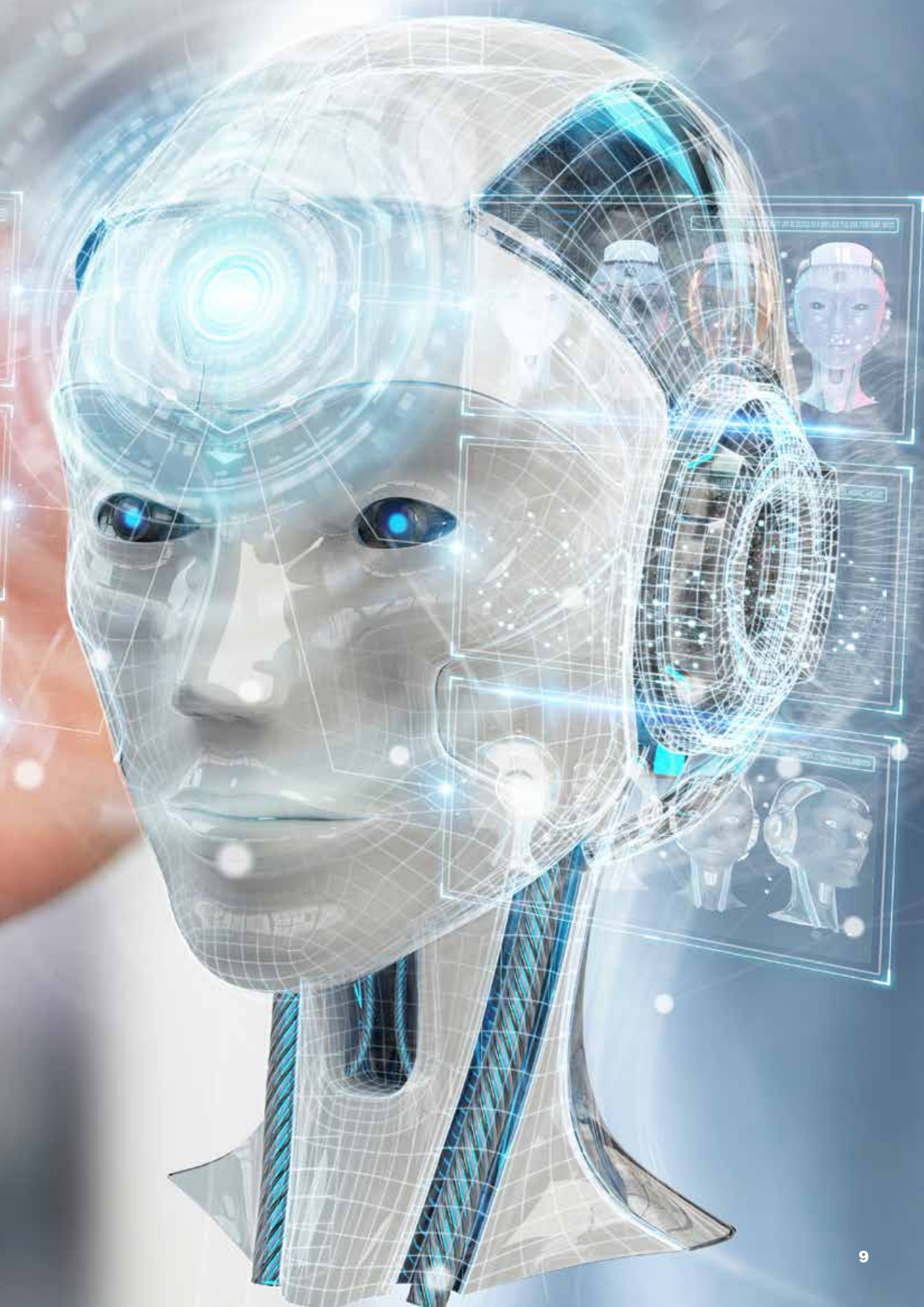
The Guidelines set out various requirements for AI systems to preserve ethical principles and establish trust, and suggest various

technical and non-technical methods for implementation. Below you can see a list of all the requirements for trustworthy AI.

- Priority of human action and human supervision: AI systems should support human autonomy and decision-making and be able to be supervised by humans. The less supervision a human can have over an AI system, the more extensively it must be tested before and the more strict the guidance and control must be.
- Technical robustness and security: For a system to be considered robust, it must work properly and the results must be accurate and reproducible. Security includes risk prevention, the creation of fallback plans and the ensurance of protection against attacks and misappropriation.
- Privacy and data quality management: User data must be protected and not used to discriminate against other users. Data quality management is necessary to ensure the quality of data, documentation and to control over access.
- Transparency: Only when a system and its results are traceable, i.e., verifiable and explainable and the capabilities and limitations of the system are clearly communicated and defined, then the system can be called fully transparent.
- Diversity, non-discrimination and fairness: All stakeholders must be involved and treated equally during the lifetime of the AI system. It is important that bias is avoided and that the system is designed to be as barrier-free and user-oriented as possible.
- Social and environmental well-being: An AI should be used for the benefit of all people and during its lifetime, the society, other sentient beings and the environment should be considered as stakeholders. This includes sustainability, environmental friendliness, social awareness, and the preservation of democracy.
- Accountability: Provisions must be made to ensure responsibility and accountability for AI systems and their results. Vulnerable persons must be given special consideration and adequate legal protection must be provided in case that adverse effects occur.

I Conclusion

In the future, it will be even more important for the development of good and safe AI systems that there are no negative effects and that all decisions are also comprehensible for us humans. Otherwise, people will place little trust in AI. And if there is a lack of trust, there will also be a lack of acceptance of such intelligent systems in our society. It is particularly important that both developers and users know and understand the opportunities of AI systems, but also their risks and limitations. Through this and through adherence to guidelines, mindfulness in development and the use of human-centered development and design principles, we can create AI systems that are capable for the future. ◆



Software-Reengineering: When will the legacy system become a problem?

– DI (FH) Alexander Leutgeb
Head of Industrial Software Applications Unit

If ain't broke, don't fix it

Business-critical software is not immune to a certain aging process. However, an equivalent replacement is often not that easily available. But when is the time to replace the legacy system with the help of software reengineering? A common saying is: "Never change a running system." This sentence, which is presumably a modified form of the statement "Never change a winning team" by the British soccer player and coach Sir Alf Ramsey, is, however, hardly known in the English-speaking world. There, the statement "If it ain't broke, don't fix it." is used instead. The statements are similar, but the second formulation does not seem quite as strict and dogmatic. In other words, one should only try to fix something if it is actually defective or damaged.

From a functional point of view, this would mean that a system is no longer capable of performing the task for which it was designed. For software, we can go a step further here. A software system is "in need of repair " when it reaches a state that makes maintenance difficult or impossible. We have collected six signs for you that indicate that action is needed for your software system:

Missing or outdated documentation

If the documentation of a system is outdated or does not match the actual system in many parts, this is a clear sign of a legacy system on which many changes have already been made. Even worse is the lack of comprehensive documentation. Such a circumstance complicates the maintenance and further development of a system and makes it almost impossible to train new employees quickly.

Original developers have left the company

Usually, a lot of knowledge about a complex system is stored in the heads of a few employees. This leads to a situation where the continuity of the software is directly dependent on the original developers, especially if the documentation is not adequate. The continuous departure of these employees inevitably leads to a situation that makes further development of the system almost or fully impossible.

The knowledge base for the system is missing

A significant alarm signal is when there is hardly any understanding of the basic functioning of the software among the employees. In the worst case, no one in the company knows the original contexts and concepts. This makes it extremely difficult to find sustainable solutions if there is a problem. The solution often is quick "hacking" to solve the problem, but these accelerate the malicious circle even more.

Even the implementation of small changes is very complex

A "law" formulated by Lehman and Belady on the evolution of software [1] states that systems become increasingly complex until active work is done to reduce the complexity. If even the implementation of small changes or enhancements requires a great deal of effort, this is a clear sign that the system has already become too complex. If even small changes require a large amount of time, more difficult changes become almost impossible to implement.

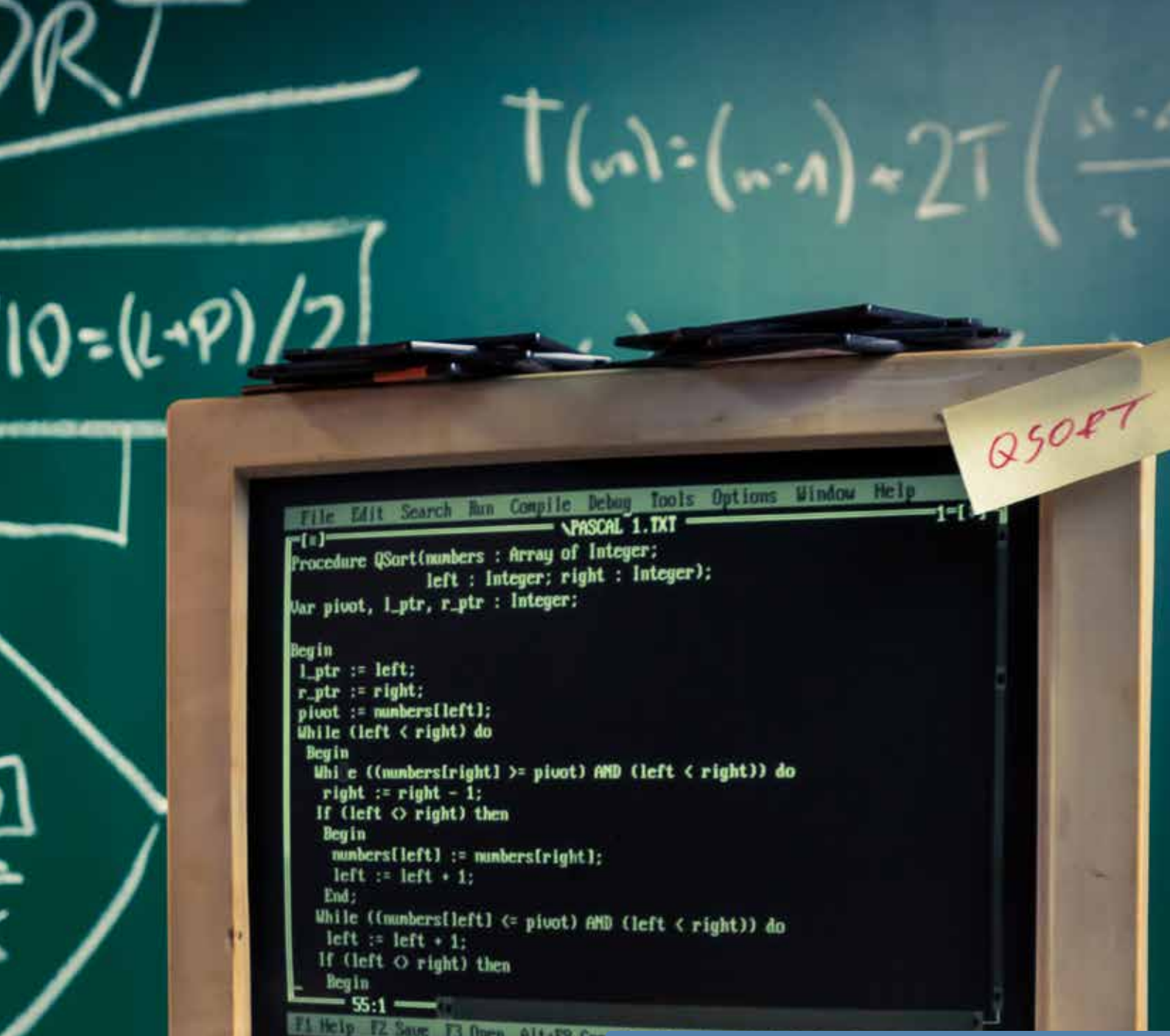
Constant need to fix bugs

Large software projects will not ever be free of bugs, so, making regular "bug fixes" are necessary. However, if repeatedly fixing a bug leads to the appearance of a new bug, this is a sign that the architecture of the system may no longer meet current requirements. This represents the risk that small changes in the program will have unforeseen negative effects.

Code smells

Code smells are parts of the source code, which are not faulty, but badly structured and implemented. For example, the duplication of source code, i.e., the use of the same lines of code in different parts of the program, has a very bad influence on the subsequent maintainability of the system.

[1] M. M. Lehman, "Programs, life cycles, and laws of software evolution," in Proceedings of the IEEE, vol. 68, no. 9, pp. 1060-1076, Sept. 1980 see https://ieeexplore.ieee.org/document/1456074



The solution: Software Reengineering

If the "symptoms" are recognized in time, the decay of the software can be counteracted by using suitable methods in order to keep the quality of the system at a high level and to avoid possible failures. By Reengineering an existing system is restructured and above all also prepared for future development and extension.

While maintenance involves adapting software that is in a productive environment to changed environmental conditions and correcting detected errors, reengineering involves redesigning and implementing systems or parts of them from scratch.

RISC Software GmbH will gladly support you in the initial analysis as well as in the implementation of a reengineering of your software system. ◆

Working with Fortran in 2020: Areas of application

– DI Dr. Christoph Hofer
Software Engineer, Industrial Software Applications Unit

Fortran - the proven programming language for technical applications

The name Fortran stands for "FORmula TRANslator" and is one of the oldest programming languages. For many software developers it is the archetype for an old, ponderous, limited and difficult to understand programming language with which it is best not to have anything to do. For the old versions of Fortran, this bias may indeed be true. However, Fortran has changed a lot in its long history, so in its "modern" variants (like Fortran 2003) the language has a much worse reputation than it deserves. The typical use case for Fortran as a programming language is computationally intensive numerical simulations, such as weather forecasts, flow simulations, stability calculations and many more.

From old to new

Fortran is considered as the first ever realized higher programming language and was developed between 1954 and 1957 by IBM (FORTRAN I). The extent of the language was still very limited, for example, there were only integers and reals (floating point numbers) as data types and no functions yet. In the following years new improved and more extensive Fortran versions were developed (FORTRAN II, FORTRAN III, FOR-TRAN 66). The next big update Fortran got was in the year 1977 (FORTRAN 77). Due to new features in the language, this version became very popular and thus quickly became "the" Fortran. Even today, when talking about Fortran code, mainly FORTRAN 77 code is meant, which also explains those numerous bias against the language. Since then, there have been several more updates to the language, in order to keep it up to date with modern programming concepts and standards.

Major milestones in the development were the updates to Fortran 90 and Fortran 2003, which, in addition to the name change (FORTRAN → Fortran), added common concepts such as free source file formats, modules, operator overloading, derived data types, pointers and object-oriented programming to the programming language, among others. In addition, Fortran 95 and Fortran 2008 were each minor updates to the language. The latest version of the Fortran standard is Fortran 2018, although no compiler provider supports all of the features yet.

Compiler: the agony of choice

In addition to the extent of the programming language itself, a simple and pleasant workflow during development is also important for the programmers, i.e., in which development environment can one test and debug the written code well with which compiler. The limited number of available development environments strengthens the impression that Fortran does not belong to the most common languages nowadays. To program modern Fortran (Fortran 2003/2008) one has several compilers to choose from, open source and as well as commercial ones.

- Intel Fortran (ifort, commercial)
- NAG Fortran (commercial)
- GNU Fortran (gfortran,Open Source)
- Flang (formerly "f18," in development, Open Source)

Regarding to development environments, there are IDEs developed specifically for Fortran, such as the NAG Fortran Builder, or integrations for existing IDEs, such as the Intel Integration for Visual Studio, or extensions for editors, such as Visual Studio Code. In addition to the included debuggers (gdb, idb), external commercial debuggers, such as Total View for Fortran, are also available. (https://totalview.io/)

Why is Fortran still used today?
Legacy code

Many Fortran systems which are still in use today were developed in the 1980s. At that time, Fortran was the language for scientific computing and numerical simulations. Such programs contain a lot of knowledge and experience, often from several generations of engineers and scientists. Transferring them to other "modern" programming languages is often far too time-consuming or expensive, and the added value for companies would hardly be given. Reengineering the program code to a modern Fortran standard and actively developing it is often the better alternative.

Fortran is designed for technical application

As already mentioned at the beginning of the article, numerical simulations are the typical use-case for Fortran as a programming language. These areas are characterized by the fact that, on the one hand, efficient program execution is important but, on the other hand, the calculations are also described by mathematical formulas. Fortran gives programmers the possibility to represent these formulas in a readable and compact way in the program code. Furthermore, the language is designed for efficient calculations, so that high-performance code is the rule rather than the exception.

Fortran is a simple language

Compared to C++ or scripting languages like Python, Fortran has a rather limited feature set, which is why for many of today's business applications the development would be very tedious, but for technical/scientific applications it is very suitable. The smaller feature set allows even less experienced programmers to develop good software. Especially because of the many possibilities of C++20 but also like with Python, less experienced programmers are often overwhelmed with the possibilities. Wrongly used complex languages often create more confusion than they contribute to the solution of the problem. In this correlation, Fortran was designed for scientists and engineers, not for computer scientists and software developers.

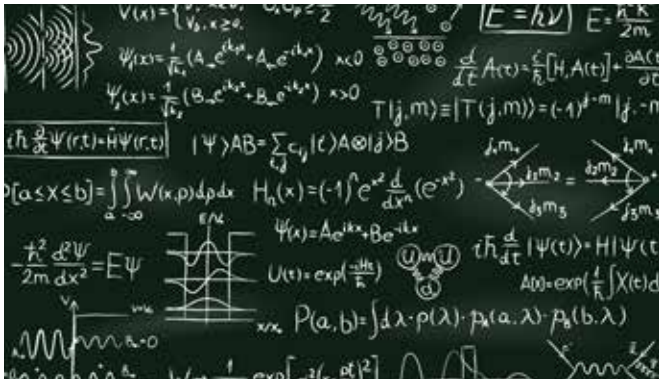
Integration Fortran/C/Python

2003, a new possibility for easier interaction between Fortran and C code was created with a module called iso_c_binding. This module includes:

- types of primitive data types (integers, reals) corresponding to the corresponding C data types (int, double...)
- functions for working with C pointers, such as converting C pointers to Fortran pointers or reading the storage address of a Fortran variable
- constants for non-displayable C characters, such as the newline character (\n) or the horizontal tab (\t)

Not all data types from Fortran can be transferred into the C code by means of the iso_c_binding and vice-versa. For example, in Fortran there is no equivalent of a C union or in C there is no direct equivalent of the Fortran ALLOCATABLE attribute for arrays, which allows a dynamic allocation of memory that is automatically freed again if the array goes out-of-scope. Furthermore, C/C++ has no support for array slices, i.e., a structured subset of an array, so copies must be created when passing them.

In some projects of RISC Software GmbH the default was to write Python interfaces for Fortran code. Fhe SWIG program (http://swig.org/), which generates code for Python C interfaces, has proven itself for this purpose. To generate a Python/Fortran interface, the Fortran/C interface is written first and then the Python/C interface is generated using SWIG. SWIG is very flexible in its application and even allows numpy-arrays as arguments of the Python functions, which leads to efficient Python, on the one hand, and readable Python, on the other hand. By integrating it into the CMake build-system, this approach works platform independent and provides a simple and efficient workflow in development. ◆



Estimating in Agile Projects with Story Points

– DI (FH) Andreas Lettner
Head of Domain-specific Applications Unit and Head of Coaches

Effort estimation of agile software projects

In order to estimate the extent of a project for offer as well as for implementation in advance, person hours or person days are traditionally used. This involves some problems and dangers. Complex tasks often cannot be estimated before the start of the project, too generous buffers or short time windows often make offers inaccurate.

One of the biggest challenges in the development of individual software lies in a task that sounds rather simple: the estimation of effort. The term implies an "estimate," which basically indicates the empirical nature of this task. Nevertheless, especially in the initiation phase, the most "accurate estimates" possible are in demand, and in some cases these are even used as the basis for contracts.

Everything is relative

Estimation with Story Points is different: it is not based on absolute values but on relative ones. Story Points can be applied to different values that define the complexity of a task. However, the most common is a general assessment of the extent or effort. A direct conversion into a concrete time statement is not possible for now.

Story Points are used by members of the development team to estimate tasks in relation to each other. As an example, a team defines a known task which, with the associated estimate (e.g., 3 Story Points), provides a starting value for further estimates. Each future task is now estimated in relation to this metric by the entire development team. People usually find it very easy to estimate an effort in terms of "greater than" or "less than", which is why there is a first advantage of Story Points at this point. Furthermore, Story Points are used in an adapted Fibonacci-sequence which includes the values 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, and ∞. This scale allows an estimation in increasing ranges, which here makes especially not yet exactly specified or too comprehensive tasks recognizable. It is not necessary to use this scale in its entirety. Such an estimation is carried out by the development team, e.g., with cards in "Planning Poker".

Include different velocities

In order to be able to use Story Points for agile software development, another value is required: the team velocity. The team velocity indicates how many story points the team can accomplish on average per iteration. This value is variable and usually changes over the duration of a project, which is why continuous observations and adjustments are necessary. Above all, the stability of the team composition is an essential condition. The reason is that the estimation is not only relative with respect to the tasks, but also independent of the performance potential of the individual team members. A task with 3 Story Points can mean an actual time expenditure of 5 hours for one team member, and only three hours for another team member. Since Story Points are assigned as a team and the tasks are also processed as a team with known team speed, there is good predictability about the possible implementation extent of an iteration.

If three values are used for the team velocity, so the average, the fastest and the slowest of the last six iterations, a corridor can be specified for the predictability of the implementation duration, which offers further advantages with regard to planning. This makes it easy to identify tasks in a backlog which are not possible in terms of time according to the current planning or which can be guaranteed.

| Win-win-situation:

Estimating with story points offers customers the best possible transparency for planning, controlling and adjusting a project roadmap. Thus, reliable state-ments about feasibility, completion, etc. can be made despite the agile approach. The development team is also guaranteed the ability to plan while at the same time remaining agile. RISC Software GmbH has been successfully using story points in agile projects. ◆

Can Data Science lead industrial companies out of crisis?

– Mag.ª Stefanie Kritzinger, PhD
Head of Logistics Informatics Unit

How it is possible to minimize costs, respond flexibly to fluctuations in demand and avoid production downtime due to disruption?

Particularly in economically difficult times digitization and the automation that goes with it play a crucial role. Manufacturing companies are facing a previously never-existing challenge: processes should already be digitized and partially automated controllable in order to monitor and manage production remotely. Sales markets and workforce planning are subject to external, uncontrollable influences and production in small batches is more attractive than ever before. Depending on the industry and the level of digitalization, some companies can handle this easily, while others cannot.
The enormous importance of digital and virtual networking is currently being demonstrated to us in the fight against the pandemic. Thanks to the digitization efforts of recent years, process and production data have increasingly been seen as an essential part of value creation. Those who have already done their homework have for some time been collecting extensive and automated data on their own company processes. On the one hand, in order to use it to analyze and process real-time information for reacting to short-term changes in production. On the other hand, in order to be able to derive future events from the collected data pools and forecast them as accurately as possible.

Processes: Increase quality and minimize costs

In order to increase quality and minimize costs, an important success factor is to extract valuable information from the collected production data. Nevertheless, this is not a trivial task. Intensive data engineering makes process and production data due to quality-relevant process steps usable. Necessary parameters are identified in order to perform automated data analyses using data and visual analytics or modern artificial intelligence methods. This makes it possible to detect anomalies, evaluate them correctly, and predict their effects on the final product quality. The improvement of quality management is complemented by the traceability of production parameters and quality characteristics of the entire process. Thus, a better understanding of the production process is possible by recognizing cause-effect relationships based on anomalies and patterns. This also allows maintenance intervals and cycles to be optimized and, subsequently, production processes to be improved.

This makes it possible, for example, to minimize scrap by ensuring that the machines produce at the correct operating temperature, minimize unplanned downtimes, or optimize maintenance intervals.

Detect bottlenecks early: Prescriptive Analytics

Especially in times of crisis, short-term fluctuations in demand are daily business. In most cases, production systems are also highly complex due to their individual structure and organization. Scarce resources, special requests from customers, the resulting product variety, and deadline pressure overload existing capacities and lead to cost-intensive bottlenecks, for example, due to additional staff or delayed deliveries. Good preparation for the early detection of bottlenecks is based on intelligent forecast-based planning.

Production figures can be predicted on the basis of historical production figures and other influencing parameters as well as with the help of modern methods from the field of statistics and artificial intelligence. Based on these forecasts, which are very likely to be accurate, adequate measures can be derived and the expected development can be influenced in a positive kind of way - this is subsumed under the term prescriptive analytics. Predictive analytics functions create greater transparency in upcoming production. Targeted calculations and visualizations make it clear where bottlenecks may arise and where delays will occur based on planning. This provides real insights that enable intervention before the problem even reaches customers.

Avoid downtime: Fault management

Lack of material, lack of personnel and changing requirements as a result of a crisis are mostly known factors for production stoppages. Disruptions often result in lost sales, large financial losses, and negatively impact the bottom line. Little attention is paid to managing a disruption from its discovery to its full recovery. Especially in the case of unforeseen events, efficient disruption management is able to absorb disruptions in a responsive manner. Rescheduling of the production process is necessary in order to maintain delivery reliability as far as possible with limited resources due to supply bottlenecks or short-time work, while at the same time complying with the recommended measures.

In the current, very turbulent and dynamic environment, it is more necessary than ever to increase the degree of digitization in order to be able to adhere to the potential targets of increasing quality, minimizing costs, identifying bottlenecks and efficient fault management. In that case, the two important key levers are process transparency and responsiveness.

Know-how

The data engineers and data scientists at RISC Software GmbH possess extensive expertise and many years of experience in a wide range of areas of data management and data analytics. By using modern methods from the areas of data analytics and visual analytics as well as machine learning for smart data analysis and forecasting, the challenge of Big Data can be perceived as an important opportunity for process and revenue optimization. ◆

Software Modernization

– Michael Hava MSc, DI (FH) Josef Jank MSc, and DI (FH) Alexander Leutgeb
Software Architects & Project Manager, Industrial Software Applications Unit

Incremental re-engineering for sustainable software development

For long-lasting software systems, maintenance costs far exceed initial development costs. Escape the cost trap through timely proactive modernization measures. An evolutionary approach guarantees predictable costs, continuous releases and immediate customer benefits with manageable risk.

Get out of (technical) debt

Research shows that the success of companies is increasingly determined by software. One of the most successful retailers (Amazon) is therefore not one of the most successful software companies by chance, but has thus created the basis for success. Particularly in the industrial environment, increasing demand has arisen in the area of software development in the course of digitalization over the last few decades. Many companies have therefore established their own software development teams. Initially, however, these teams often consisted purely of domain experts without in-depth software development expertise. As software development increasingly emerged as a core topic, the need for software development as an independent discipline suddenly was recognized and the teams were expanded to include software engineers. The long development history, heterogeneous teams and a corresponding developer turnover lead to heterogeneity in terms of development methodology, technologies used and code quality. If this heterogeneity is not countered by ongoing consolidation, the cost of maintaining the software increases enormously over the years. In addition, changes are increasingly difficult and the integration of new functionality is only possible with great effort and risk. In the worst case, technical debt can be a result in "technical bankruptcy" (see Figure 4).

Targeted modernization can greatly reduce the cost and risk of maintenance and enable a faster response to future requirements. An incremental approach means that the adjustments can be continuously transferred to productive operation and it will be ensured that the system always meets the requirements. At the same time, the effort required for modernization in relation to a new development can be estimated relatively easily and accurately. Another aspect of technical applications is that the requirements in terms of model sizes have grown considerably over the years, so that the term for calculations and the memory requirements are no longer practicable with the current implementation of the software. In the course of a modernization, such bottlenecks can be identified and eliminated by an adequate software implementation exploiting the parallelization potential of modern hardware architectures.

The ideal approach depends on the problem

The basis for a technically and economically successful modernization of legacy software is a well-considered overall strategy. It is important to approach the subject with as little bias as possible and, in addition to modernization and refurbishment, to consider and evaluate radical approaches such as completely new development or the use of standard software. When considering whether to reengineer existing software or to project a new development instead, the former alternative is often preferable because reverse engineering right down to the requirements can be enormously time-consuming (see Figure 5). Many times, almost no documentation is available and the know-how is only manifested in the source code. Due to the substantially lower costs, risks and lead times, in many cases one will therefore opt for a specific modernization, where the necessary re-engineering measures (re-code, re-design, re-specify, re-think) are specifically identified for the different parts of the system. When selecting a respective measure, it must be considered whether the resulting benefit justifies the effort.

During modernization, parts with a high and low need for change should be identified and strategically treated differently. For stable parts with a low need for change, someone should always critically question whether the effort and risk justify the potentially low benefit. Anti-corruption layers offer the possibility of a step-by-step migration from the legacy to the new system, with a stable production system available at all times and further modernization can be decided as required. An anti-corruption layer isolates parts from an overall system and ensures compatibility between differently evolutionary developed parts (legacy application with new code parts, new application with legacy code parts).

An important strategy in the course of modernization is continuous reduction of dependencies and improvement through modularization. The goal is loosely coupled components with clearly defined interfaces and, if possible, standardized data formats and protocols. In the case of highly efficient and specialized implementations in Fortran/C/C++, this enables easy integration into platforms such as Python, .NET and Java in different application

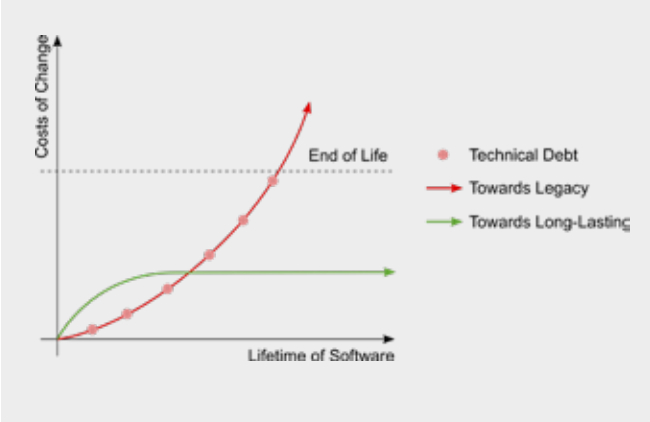


Figure 4: Premature end of life of software due to technical debt

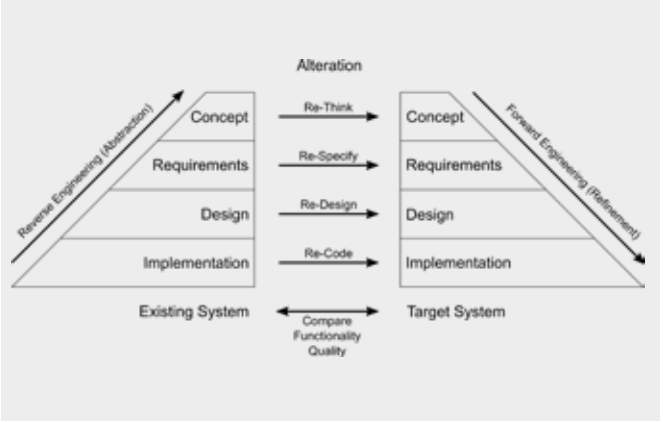


Figure 5: Categorization of re-engineering measures

scenarios. A significant potential for improvement in the course of incremental modernization is offered by the continuous expansion of test automation. On the one hand, this documents the behavior of the current system, and on the other hand, the tests represent a "safety net" to automatically detect unintended side effects and errors in the course of changes.

In the course of the concrete implementation of changes and extensions, there are numerous opportunities to profit from the development of modern programming languages and libraries and with that making the software more maintenance-friendly, more robust and better performing. Particularly with regard to the performance there were many improvements (parallelization, etc.) in the last years from which one profits immediately. In addition, parts of the in-house development can often be replaced because this corresponding functionality is directly supported in the meantime or corresponding open source alternatives are available. Technically demanding tasks with complex calculations can be solved faster (strong scaling) or larger problems can be solved (weak scaling).

In addition to the aforementioned modernization at product levels, the entire software development process should be analyzed and, if necessary, improved or modernized, too. In recent years, studies have clearly shown that in particular the rapid changeability (incl. rollout) of software is a good indicator - not only for the performance of software development, but due to the increasing importance of software usually even for the overall economic success of a company. RISC Software GmbH has been a development partner in large industrial software systems for many years and therefore also supports and pushes the modernization of the software development process at its partner companies.

Benefit from the potential of modern C++

The development of many large software systems started in the 90s. At that time, there were fewer programming languages compared to today, OpenSource was still hardly an issue and the number of available libraries was manageable. C++ was the programming language of choice for demanding modern (industrial) applications. At that time, the first C++ standard was still a work in progress and existing implementations were fragmented. Standard functionality such as containers were developed individually according to different design philosophies, because modern alternatives such as STL were not known or widespread by then.

While numerous new, modern programming languages established themselves in the 2000s, the development of C++ seemed to stagnate. After the release of C++ 98 the work on the follow-up version did begin - but completion was delayed until 2011. To prevent another long period of stagnation, the C++ committee changed its release process and delivers a new standard every three years. The response to new C++ versions is extremely positive - more and more companies are participating in the development of this powerful language. RISC Software GmbH is represented on the committee by Michael Hava via the ASI (https://www.austrian-standards.at/).

The C++ committee has succeeded in improving readability, robustness and performance in wide areas without breaking any existing code. Applications therefore benefit immediately from the improvements without a lengthy/expensive rewrite and can be incrementally "ported" to modern C++. While the existing code still works, it offers the possibility to incrementally improve source code locally. Below are some examples to show what is possible in the context of local improvements in terms of compactness, expressiveness, robustness and efficiency.

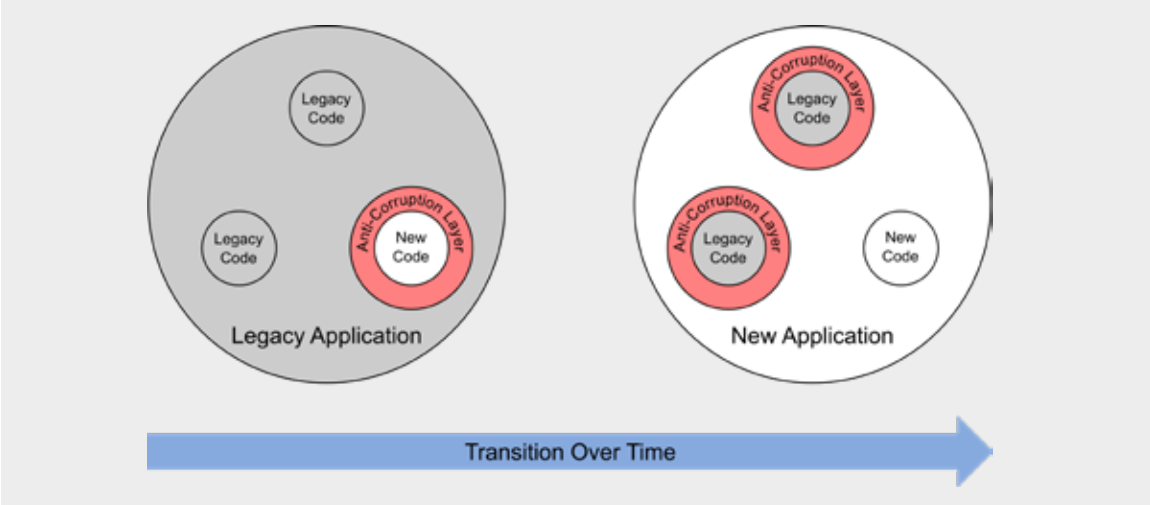


Figure 6: Gradual migration from the legacy to the new application with the help of the Anti-Corruption Layer

We have already achieved a lot for our customers

RISC Software GmbH has many years of experience in the development of native software systems in technical applications. These are often very complex and have high requirements in terms of robustness, reliability and determinism of the results or time. Its customers include Airbus, WFL, DS-Automotion and numerous others, with whom it often has long-standing development partnerships. The range of services extends from the development of new systems and the re-engineering of existing systems to consulting and training.

Development of the software library VML (Virtual Modeling Library)

The VML (https://virtual-modeling.at) is a software library from RISC Software GmbH that implements new algorithms for the exact geometric modeling of solids. It supports operations similar to Constructive Solid Geometry (CSG) and the envelope volume calculation. The VML offers good scalability in terms of the number of operations performed during modeling. The library is best suited for industrial applications that have combined requirements for geometric accuracy, speed and scalability. To ensure high efficiency, the VML is implemented in C++ and uses parallel algorithms that exploit the potential of modern hardware architectures such as multicore central processing units (CPUs) and graphic processing units (GPUs). The VML is used, for example, in the product CrashGuard Studio from the company WLF Millturn Technologies (https://www.wfl.at). CrashGuard Studio is a 3D simulation software for multifunctional CNC turning, drilling and milling centers, which enables machines with their complex kinematics and extensive machining and expansion options to be simulated very realistically.

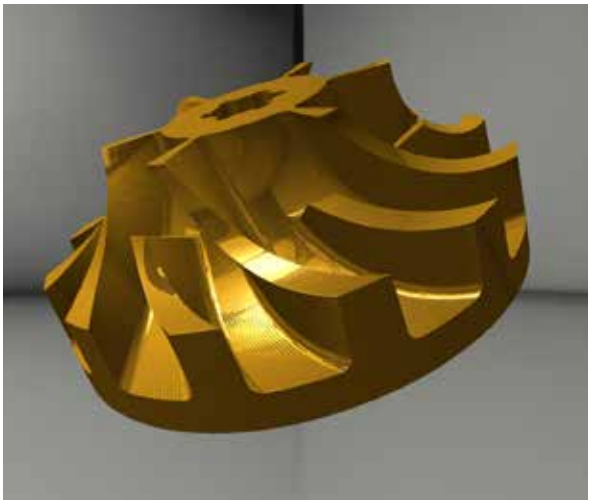


Figure 7: Software library VML (Virtual Modeling Library)

Re-engineering of the Lagrange structural optimization system

Airbus Defence and Space uses the multidisciplinary structural optimization system Lagrange in the field of aircraft structure engineering, starting its development in the early 1980s in the Fortran77 programming language. RISC Software GmbH started the re-engineering of this software system in the year 2005. The first step was to replace a critical calculation component in order to remove its limitation with respect to the maximum problem sizes. The new development was done in C++ and was integrated into the old system. In a next step the entire system was ported from the old Fortran 77 language standard to the newer Fortran 2003/2008. Since the overall system had limitations in many places with regard to the maximum problem sizes and the expandability was limited due to the design, an incremental re-engineering of the overall system in Fortran 2003/2008 was done. The new system should also be able to be used in production operation at any time and deliver the same results as the old system.

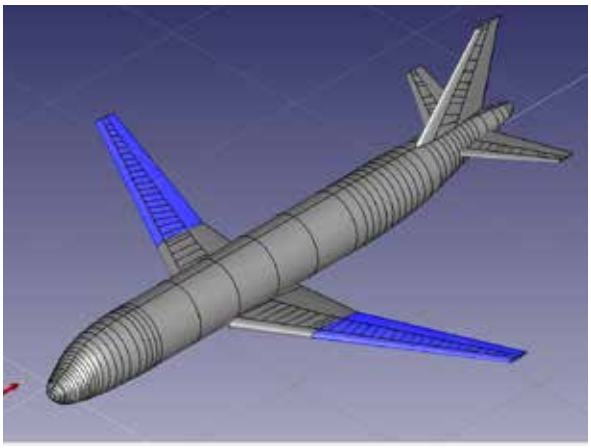


Figure 8: Structural optimization system Lagrange in the Airbus Defence and Space division

Consulting for optimization measures of a lubricant film calculation routine

The FH Wels developed a routine for the calculation of the elastohydrodynamic pressure, which occurs in the oil film between piston and cylinder in an internal combustion engine, within the framework of a simulation system. RISC Software GmbH conducted a consulting for optimization measures of this calculation routine implemented in C++. The aim was to make the best possible use of the potential offered by modern hardware architectures. Through an analysis of the system, the critical points were identified and suitable optimization measures were developed. Finally, the approach and the results were communicated in a workshop. ◆

Figure 9: Combustion engine simulation for the calculation of elastohydrodynamic pressure.

Welcome Change

– DI (FH) Andreas Lettner
Head of Domain-specific Applications Unit and Head of Coaches

The new Scrum Guide

Welcome Change - an essential principle in agile development, which is supported in its importance in the Scrum framework by the three pillars of transparency, inspection and adaptation. And just as change is something we encounter every day during product development, it is also necessary to allow and live change in the development of agile practices and frameworks. On November 18, 2020, the time had come: the new Scrum Guide was published. Here you find the fundamental changes at first sight.
First of all, what hasn't changed? Scrum is still Scrum - a lightweight framework that helps teams solve complex problems and is designed to deliver value to customers. Customer-centric, nevertheless a focused view on the team and its members.

More freedom leads to more individuality

The Scrum framework has never been particularly prescriptive in its practices in order to get a team to work and grow optimally. The new Scrum Guide goes a step further and becomes less "prescriptive" in many parts and reduces to the essentials. For example, in the Daily Scrum, the previously defined questions have been eliminated.

A stronger team

In the future, the Scrum Team will only be one team. This may sound strange at first glance, but with the previous designation of the "Development Team" in the Scrum Guide, there was a danger that the developers would form themselves as a "subteam" and that the positive synergies in the entire Scrum Team would be missing. In the new Scrum Guide the term "Development Team" is replaced by "Developer." There is now only one team - the Scrum Team!

No more hats - only responsibilities

Previously, Scrum Master, Product Owner and the Development Team were described in terms of roles. The new Scrum Guide completely dispenses with the concept of roles and now uniformly introduces the communication of accountabilities. This may also lead to Scrum teams growing closer together in the future and being able to move better as a team.

The Scrum Master becomes a "real" leader

At first glance, a small change was made to the Scrum Master: the "Servant Leader" became the "True Leader," who continues to support the team and the organization. This change must first convince us, since the concept of the "Servant Leader" previously represented a clear image and demarcation from classic management, which could be weakened by this.

Self-organization vs. self-management

The "self-organizing" Scrum team becomes the "self-managing" Scrum team. This emphasizes the high value of the autonomy of the entire team. While in the Scrum Guide from 2017 the Development Team was still self-organizing, it is now the Scrum Team that decides together.

The way is the goal

The product goal is now the basis for these joint decisions. The product goal is intended to create a common picture of the product and to present the possible paths more clearly.

Three obligations

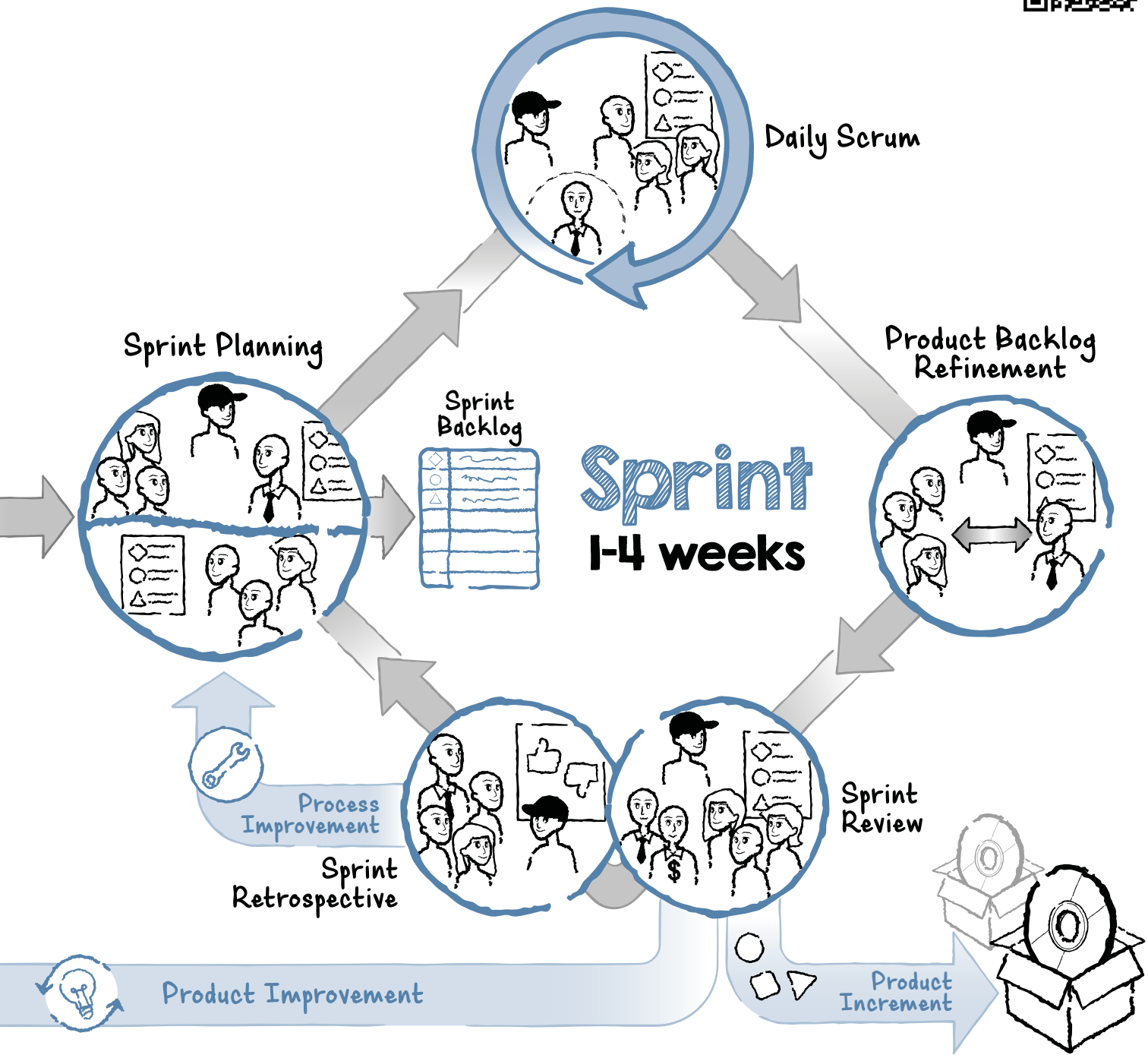
The 2017 Scrum Guide already mentioned the sprint goal and the definition of Done, yet these were not particularly strongly anchored. Simultaneously with the introduction of the product goals, these three were now assigned to the artifacts as commitments:

- Product backlog receives the product target
- Sprint Backlog receives the sprint target
- Increment receives the definition of Done

Previously, the sprint goal was communicated as part of sprint planning. What is new here is the joint definition of the sprint goal by the Scrum team. Through the institutionalization of a preceding, additional question in Sprint Planning, the following topics have arisen:
- Why is this sprint valuable?
- What can be implemented in the sprint?
- How is the chosen work implemented?

Scrum

You can download the poster by scanning this QR code



Conclusion

The changes in the Scrum Guide 2020 can simply be described with one word: exciting. We are curious to incorporate the new topics into our everyday project work and to experiment with them. We are confident that many of the changes mentioned here will influence the development of our way of working in a positive way. We would like to invite our clients to take this step of development together with us.

About Scrum

Scrum is an agile approach to project management developed by Ken Schwaber and Jeff Sutherland and frequently used in software development. The approach is defined in the so-called Scrum Guide and is developed independently of companies and manufacturers. An updated version of the Scrum Guide was published on November 18, 2020. ◆

"OK Google: What is Natural Language Processing?"

– Sandra Wartner, MSc
Data Scientist in the Logistics Informatics Unit

How machines read, decode and understand human language

Not every language is created equal - while humans have created their own communication channels over thousands of years, millions of zeros and ones serve as machine code, or machine language, for computers to understand and execute commands. Natural language processing, or NLP, enables machines to read, decode and understand human language. Speech assistants, spelling correctors, email spam filters - NLP as a technology is omnipresent and already hides behind many processes and software applications deeply embedded in our everyday lives. The often hidden potential in many mountains of data is far from exhausted.

The flood of data generated by us humans is growing day by day. For the year 2020 alone, growth statistics showed that 1.7MB of data is generated per person every second. We send photos, store documents in the cloud, stream music or videos, communicate via video conferencing tools, and use many more conveniences that the Internet offers us. In the last two years alone, approximately 90 % of the world's data was generated - and the numbers continue to rise. The COVID pandemic, among other things, is also contributing to a sharp rise in the growth rate due to the increased need for online communication and home offices.

A considerably large amount of the existing data consists of text data. We primarily generate these ourselves by writing emails, product reviews, tweets, or text messages, for example. At the same time, we can use the potential of the continuously growing mountains of data to create the applications that increasingly support us in our everyday lives in the first place. We use translation functions from one language to another (e.g., DeepL), programs alert us to typos when composing texts and messages, digital voice assistants such as Alexa, Cortana, Siri and co. support us in a wide range of activities, and search engines offer search completion - all these services and functions are built on one essential technology: Natural Language Processing (NLP).

Artificial Intelligence as an interface between human and machine

Machine processing of natural language is not a new field of research, however, due to the availability of higher computing power, enormous amounts of data (Big Data) as well as modern algorithms, recent years have brought a multitude of revolutionary achievements in the NLP environment: computers are able to read, understand and speak. As an interdisciplinary field of linguistics, computer science, and artificial intelligence (AI), NLP enables communication between humans and machines in different forms (written and spoken) and in a variety of languages.

If we want to ask the Google Assistant on our smartphone to have a synthesized voice explain NLP to us, a simple "OK Google" and the trailing question will suffice. Optimally, we will receive an answer that satisfies us and provides exactly the information we were looking for. While this task sounds relatively simple for execution by a human, for a machine it means breaking down language into its elementary components, understanding the question and context, and having to solve sequentially different problems.

Natural Language Understanding (NLU) focuses on the extraction of information from text and thus on the acquisition of text understanding with respect to a certain aspect. Syntax (grammatical structure) and semantics (meaning of words) play an important role. Examples of this are:

- grammatical analysis (e.g., Part-of-Speech (POS) Tagging),
- recognize people, places or other keywords in texts (e. g., Named Entity Recognition (NER)),
- sentiment and opinion analysis and
- classification of text into predefined categories.

Natural Language Generation (NLG) focuses on the generation of natural language and is used, among other things, for the automated creation, summarization or translation of texts.

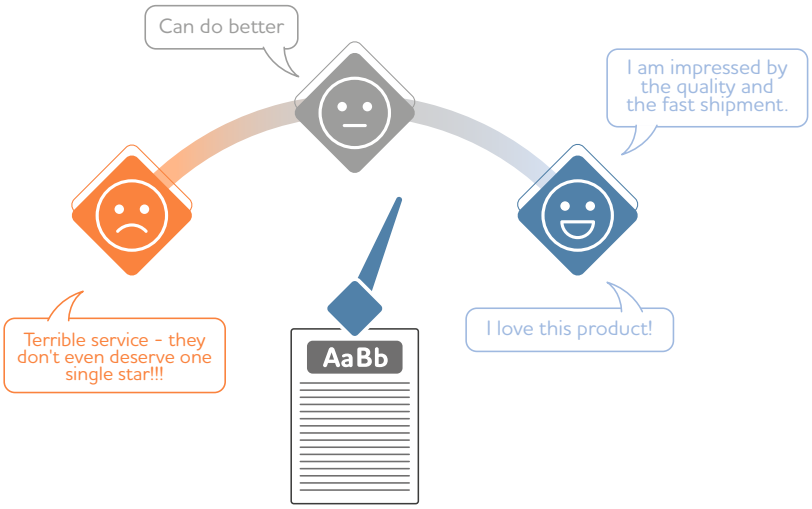


Figure 10: Sentiment analysis

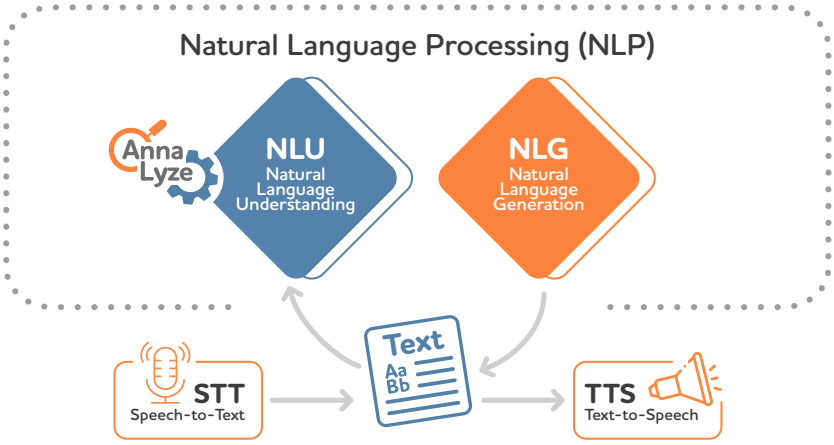


Figure 11: NLP-relevant components

Since NLU and NLG work exclusively with written language, a component for speech recognition (Speech-to-Text, STT for short) and speech synthesis (Text-to-Speech, TTS for short) is often required, which then act as an interface between the NLP system and the real world.

For the "OK Google" example, this means that the query is converted from spoken language to written language using STT. The query, which has been recognized by NLU, can be responded to, for example, by collecting and evaluating relevant search results. The knowledge generated in this process can mostly (depending on the type of result) be played back acoustically using NLG and TTS or the best hits can be displayed on the terminal.

NLP is considered one of the most complicated problems in computer science. Natural language in itself has no identifiable structure (often referred to as unstructured data) and is a complex system of strung together, partially interdependent characters and therefore not easy to understand. German, English, Russian, Japanese, Arabic - each language has its own complex syntax and peculiarities. In addition, there are further complications, as language often does not work in a linear way, but makes use of different stylistic devices, idioms and information between the lines. Recognizing sarcasm is not always possible even for a human. Ambiguities of single words have to be resolved by a context analysis, e.g., to associate the word "bank" with a seat or a financial institution. Mumbling, stuttering, speaking in dialect and background noises make it difficult for the voice assistant to evaluate the information and can lead to an incorrect answer. Algorithms have to face these and several other challenges in order to meet their requirements.

Older systems relied on rule-based or purely statistical approaches, whereas the breakthrough only came with machine learning (especially deep learning) and the availability of large amounts of data. Machine Learning models try to infer general patterns from a set of examples (How do people use language? Which grammar rules are applied?) and apply them to decide for an individual case - similar to a child learning human language. The more examples the system is provided with and the better they reflect reality or the future application scenario, the higher the hit rate for new, unknown tasks the system is supposed to solve. Currently, the most promising models or state-of-the-art results for tasks from the NLP domain are obtained with Deep Learning algorithms, which allow more complex modeling than conventional Machine Learning models. Deep Learning was inspired by the way a human brain works and employs multi-layered neural networks. The highly connected structures enable "deep learning,"which is essential especially for the complex construct of language.

Text analytics und use-Cases in a company

In order to exploit the often untapped potential in corporate data and solve business problems, existing (raw) data must be examined and knowledge derived from it, quantified and visualized. Text analytics can be used to map this process in order to process large volumes of unstructured text data and gain insights. Only if a uniform understanding of all stakeholders can be created for the results and the step of seamlessly integrating solutions into existing workflows and systems can be mastered, can further decisions for action be derived from this and thus the success factor for the company be increased in the long term.

More and more companies from different industries rely on NLP solutions to better manage and use the accumulated, different text forms in a variety of areas. Especially when there are recurring tasks to be done, automating these tasks can be useful.
In the following, exemplary use cases are listed to illustrate the broad applicability of NLP solutions.

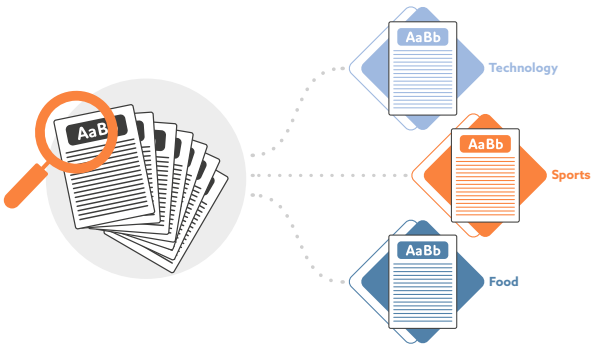


Figure 12: Document classification

Use Case: Automated document-classification

You are working in controlling and would like to receive only those documents (or document types) for which you are responsible.

Use Case: Automated extraction of information from documents such as invoices or delivery bills

You are the manager of the receiving department and want to control the details of the deliveries instead of entering them manually.

Use Case: Customer support

You are an online mail order company and want to reduce customer service response times by automating the processing and answering of customer inquiries.

Use Case: Automated evaluation of customer feedback

You are a marketing representative and would like to get an overview of the mood and reactions regarding to your new advertising campaign on social media.

Use Case: Social media analysis

You are an employee of the federal office for the protection of the constitution and want to discover and monitor extremist, radical and violence-glorifying social media profiles and posts.

Use Case: Support in clinical documentation and organization

You work as a specialist and would like to summarize essential information from several, extensive medical histories of individual patients in order to obtain a holistic view of the disease history.

I Conclusion

Progress in the NLP field is unstoppable and is continuously providing new and better solutions to a wide range of problems. The precision of the models developed and their availability to the masses continue to increase and more and more developments are making the leap from research to production. In any case, it remains exciting to see what further breakthroughs the coming years will bring - one thing is clear, they will come. RISC Software GmbH is happy to support you in the submission and implementation of (research) projects in the field of Natural Language Processing. ◆

C++20 Concepts

– Michael Hava MSc
Senior Software Architect in the Industrial Software Applications Unit
and member of C++ Standard Committee

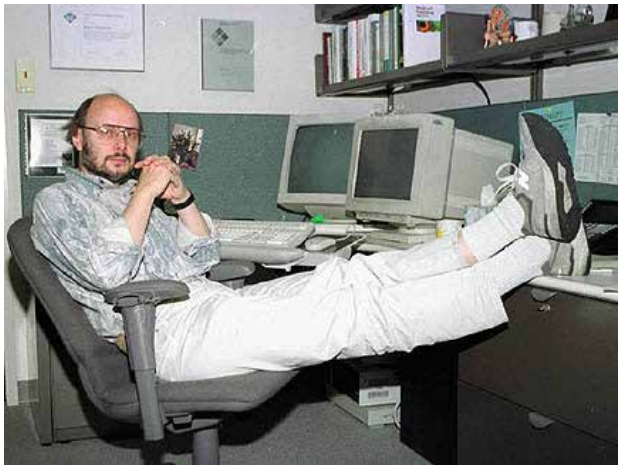
A short historical overview about the development of the programming language and the implementation of “Templates” and “Concepts.”

The famous Bjarne Stroustrup, the inventor of the programming language C++, which is still widely used today, originally started developing C with Classes in 1979 at Bell Labs. His goal was a programming language that would combine the speed of C with the organizational capabilities (=object orientation) of Simula which were and still are necessary for large projects.

Due to the great success, Stroustrup began barely ten years later (1987) to work on an extension of the language, which in the meantime had been renamed C++: Templates - this generic concept was intended to enable the implementation of generic classes and functions, and thus incidentally to largely replace the pre-processor macros taken over from C. The development of these Templates was based essentially on three Design goals:

- full generality: templates should not be targeted at individual use cases, but should be designed for broad use.
- zero overhead: Code generated by templates should be indistinguishable from handwritten code (in terms of efficiency, performance, etc.) after compilation.
- good interfaces (“constraints”): Similarly as C++ had improved the type security in relation to C, generic interfaces are to represent an improvement, too.

Result of the work: It seems to be impossible to achieve all three goals at the same time at the current state of research. Since the first two goals are essential for the acceptance of templates, while constraints “only” improve usability, templates are added to C++ without them. Constraints are to be added as soon as possible.



Algorithms are defined on algebraic structures

In the year 1994, C++ (now on its way to becoming an ISO standard) is extended by the Standard Template Library (STL), a library for generic algorithms and container classes. It is the result of Alexander Stepanov’s nearly 20 years of research on generic programming and is fundamentally different from contemporary object-oriented approaches.

At the beginning of this research his realization stood firm in 1976: algorithms are defined on algebraic structures. Based on the mathematical model of algebraic structures, he developed Concepts [1], the central pillar of generic programming. Analogous to algebraic structures, concepts describe the required operations and underlying mathematical axioms, which an algorithm places on the data to be processed. The goal of generic programming is to define algorithms based only on the minimum necessary concepts and thus make them usable for a variety of concrete types.

After experiments in several programming languages (Scheme, Ada, ...) C++ is the first, which is expressive enough for generic programming Alexander Stepanov thought - even if the missing support for concepts makes complex emulations necessary. With the integration of the STL into the standard library, the idea of Concepts is also taken over in C++. From a programming language perspective, concepts are a formal representation of constraints. The focus for the completion of templates shifts subsequently to Concepts. Nevertheless, the first C++ ISO standard 1998 (C++98) appears without extensions of the template system, Concepts must be emulated further.

No completion of Templates in sight

In the first years after C++98, the standard committee focused on bug fixing and stabilization. Therefore, the development of Concepts did not pick up again until 2003 as part of the work on the next C++ standard (C++0x). Bjarne Stroustrup and Gabriel Dos Reis published a series of papers on a possible Concepts design. A research group at Indiana University also published their results in 2005. The two contrasting approaches gave rise to the joint C++0x Concepts design in 2006, which was incorporated into the working draft of the C++ standard in 2008.

In the months that followed, however, some problems with the design became apparent. Since the completion of the new standard was already postponed several times and it can be assumed that the bug fixing will lead to further postponements, the standard committee removed Concepts from the working draft again in 2009. C++11 thus appeared 13 years after the first standard still without the completion of templates.

First Concepts-based library

While working on C++14, Bjarne Stroustrup, Gabriel Dos Reis and Andrew Sutton designed a new draft - Concepts Lite - based on the findings of C++0x Concepts in 2013. The key difference to the old approach: This time they focus on a "minimal feature" [2], which can be extended later if needed. The result of their work is published in 2015 as a Technical Specification (TS) [3].

In summer 2017, four months after the completion of C++17, concepts based on the TS are integrated into the working draft for C++20. Together with the language extension, a library extension of pre-built basic Concepts is provided. This will be followed in 2018 by Ranges, the first Concepts-based library - it contains, among other things, the STL algorithms verified with Concepts. C++20 will be adopted in February 2020 and unanimously approved in September 2020. ◆

[1] The implicit counterpart of generic programming to the explicit interfaces from object-oriented programming.

[2] For example, language resources on axioms and definition checks are missing, among other things.

[3] Independent ISO document that contains possible extensions for a standard. The goal is to get user feedback and to include a version of the content in an ISO standard.

Making better decisions thanks to Prescriptive Analytics

– DI Dr. Michael Bögl
Mathematical Optimization Specialist, Logistics Informatics Unit

Step by step from descriptive to prescriptive analytics

Digitization and topics such as Industry 4.0 and the Internet of Things have led to many companies collecting their data on a large scale in a structured manner. This collected data is then utilized in different ways. If correlations are present in this data and a prognosis model can be derived, then this model can also be taken into account in planning and control. This guarantees significant added value, such as a reduction in costs and time, a more efficient use of resources, etc. How something like this looks concretely, what is assumed and which possibilities arise, are shown in this article.

From data collection to prescriptive analytics

Digitization in recent years has created the basis for companies to continuously collect and store data on their processes and operations in a structured manner. To create added value, companies must make the best possible use of this data. Forecasting models are created from the collected data in order to be able to estimate future developments, events or conditions. These can be, for example, models for sales forecasts, wear and tear of tools in production, customer requirements, stock levels, traffic-dependent travel times, etc.

By combining the forecast models with optimization models (for calculating optimal decisions), different scenarios can automatically be calculated and compared. This provides those responsible with a solid basis for making optimal decisions. The transparent basis for decision-making guarantees that the decisions made are always comprehensible and can be argued.
Depending on how the data is used, we speak of different areas of application, or phases or stages, with the benefits increasing the higher the stage (see also Figure 13 and Table 1).

If the collected data is used to evaluate what has happened, this is referred to descriptive analytics. Relationships in the data are not yet recognized. One result could be that one recognizes that the product quality fluctuates both in the course of the day and seasonally.

1. Diagnostic analytics identifies the causes of interrelationships. The prerequisite for implementing targeted improvement measures is to know the causes. For the example from point 1, this means: Product quality fluctuates seasonally and over the course of the day because quality drops from a temperature of 30 °C; or after the first orders following a tool change, the machines have to be recalibrated (this may extend the non-productive time, but leads to higher quality).
2. Once all the relevant interrelationships and influencing variables have been determined, models can be created that depict these interrelationships. These models can be used to make statements about future behavior. This is referred to predictive analytics. For example, future sales, rejects in production or the quality of the product can be predicted.
3. In prescriptive analytics, future events will be forecasted at first, various decision options will be simulated and evaluated if necessary and the best alternative course of action is subsequently calculated. For this calculation, companies can draw on existing know-how and planning algorithms. If necessary, the planning algorithms can also be extended or additional methods integrated.

The application areas described are shown in Figure 13 according to their focus and utility. While descriptive and diagnostic analytics are directed toward the past and can create a deeper and useful understanding of current data and relationships between them, predictive and prescriptive analytics are oriented toward the future. The identified relationships from the recorded data are used to create benefits for the future.

Prescriptive analytics in decision support

Figure 14 shows the main components of a prescriptive system. The basis is formed by the collected data from different data sources. The necessary models (forecast model and decision model) are created from the data and the available expert knowledge. Based on these models, different scenarios can be evaluated and the decision makers can make the best decision.
A decision support system is shown, i.e., the system evaluates different scenarios, but ultimately the decision is made by the person in charge. If the decision is made by the system, then we speak of decision automation.

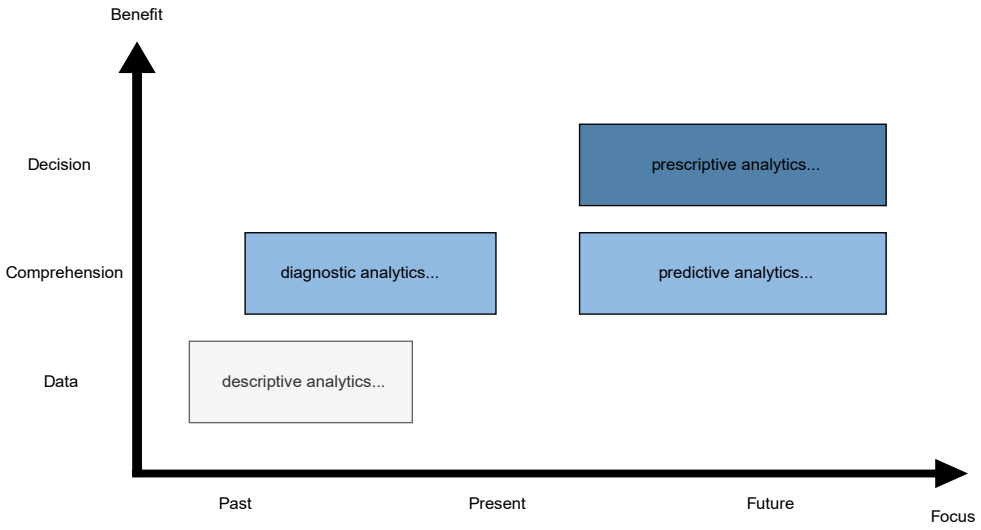


Figure 13: Fields of application

Grade	Description	Application layers	Examples
1	Descriptive Analytics	Evaluation of the collected data and reporting	• Product quality varies over the course of the day as well as seasonally • Truck driving time fluctuates over the course of the day and week • More customer orders to be processed on Monday and Tuesday than on the other days of the week
2	Diagnostic Analytics	Causes for correlations	• The product quality varies seasonally and during the day because the quality decreases above a temperature of 30 °C • Truck driving time varies due to higher traffic in the morning and evening and possibly Monday and Friday • Orders are higher because customers order more on weekends
3	Predictive Analytics	Models for future behavior	• The expected scrap in production • The expected travel time on a certain route on a certain day at a certain time • The expected order quantities for the next weeks per weekday
4	Prescriptive Analytics	Future decision options are simulated and the best alternative action is chosen	• When should maintenance be performed on the production machines to ensure that the product quality meets the given requirements • When must the truck leave in order to reach all destinations on time? • How many products must be kept in stock to avoid out-of-stock situations and how many resources are needed to process the orders as quickly as possible

Table 1: Application layers

Which methods are available?

Methods from different disciplines are available for the implementation of prescriptive models. The concept of prescriptive analytics does not prescribe any particular methods. Predominantly, methods from artificial intelligence are used. The focus is on machine learning and data mining, statistical analysis methods, mathematical programming and simulation. The selection of suitable methods depends very much on the available data, the framework conditions and, above all, the objectives (white-box vs. black-box models, integration of expert knowledge, etc.). The available method portfolio is that extensive that suitable methods can be used for many use cases.

Questions and use cases

In addition to the examples given above, there are many different applications for prescriptive analytics, just like the following ones:

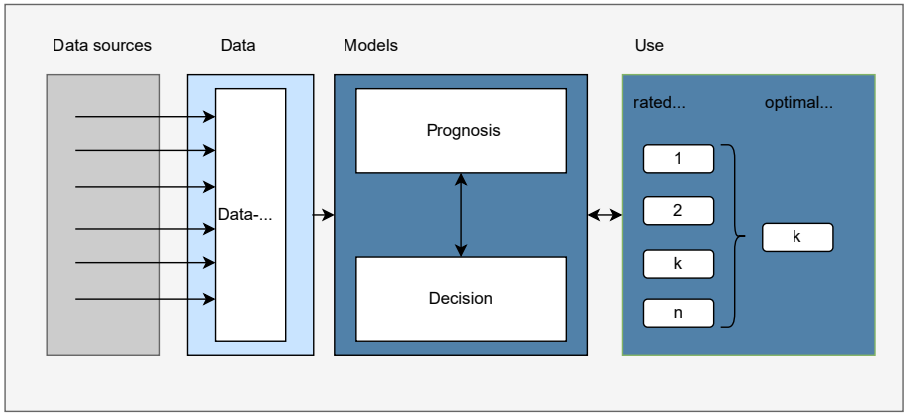


Figure 14: Components of a prescriptive system

Determination of the residual capacity

How many orders can still be produced and when will these orders be completed?

Integrated production and route planning

When should which orders be produced so that customers can be supplied with optimal transport?

Digital twin for the development of production machines

How exactly should the production machine be designed so that operation is as efficient as possible?

Price determination

How high should the price of a product be so that it appeals to a broad group of buyers and the contribution margin remains high?

Composition of the recipe

What should the formulation (proportions, environmental conditions) look like so that the product meets certain criteria?

Healthcare management

What is the (optimal) placement, diameter, duration, and intensity of the beam during radiation therapy to minimize damage to surrounding tissue?

The path to prescriptive analytics

For companies that already have a large database, prescriptive analytics offers the opportunity to generate additional value from the data already available by incorporating the insights gained from the data into planning. Companies that do not have end-to-end data collection or are not using their data for such tasks by now have the opportunity to incrementally increase the value they derive from their data. The stages from descriptive to prescriptive analytics build on each other aand valuable insights can already be gained (together with the business experts) in the diagnostic analytics. Then the next steps can be planned in order to be able to successfully shape the path towards prescriptive analytics. ◆

Data Understanders: Leveraging enterprise data through intelligent Graph Databases

– DI Paul Heinzlreiter
Senior Data Engineer in the Logistics Informatics Unit

Use modern database technology to intuitively capture and understand your data

The great strength of graph databases - databases that use graphs to connect and store networked information in the form of nodes and edges - is the extensive mapping of relationships between data points. This enables an intuitive mapping of many real-world scenarios, which have gained a lot of importance especially during the last years. Examples include modeling relationships between people in social networks, making purchase recommendations in e-commerce, or detecting fraudulent transactions in finance. In addition to these application areas, graph databases are also useful in the fields of industrial manufacturing, traffic data analysis or IT infrastructure monitoring for identifying causal relationships.

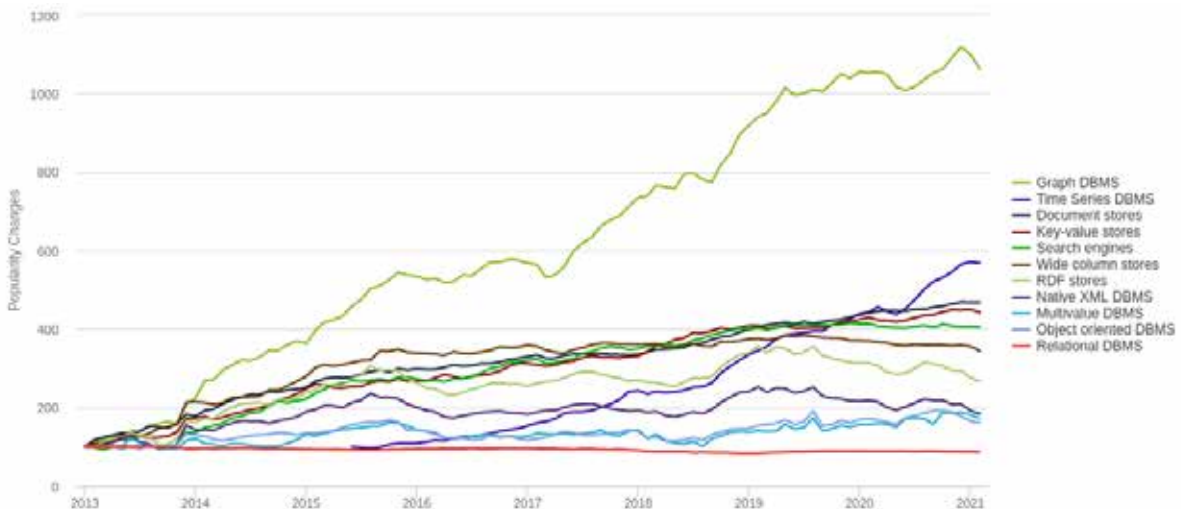


Figure 15: According to a survey on the popularity of database categories (https://db-engines.com/en/ranking_categories), graph databases represent the fastest growing category of database technologies over the last few years. Graph databases represent their data as sets of nodes and edges, where nodes represent data objects with attributes, while edges represent the links between the objects.

Graph databases compared to relational databases

Relational databases are excellently suited for representing tabular structures, such as those commonly used in the commercial sector. By using the third normal form, the data is stored in tables clearly separated according to the objects they describe, whereby other objects with which they are related are referenced via foreign keys. The goal is, among other things, to avoid data duplication and to enable linking through flexible queries in the Structured Query Language (SQL).

These applications are also characterized by the fact that once a data model has been designed, it usually remains constant over a longer period of time. However, in problem domains where one is primarily interested in the links between data, the relational data model has weak points. Since links are mapped via foreign key relationships, queries often have to be implemented as multi-level joins, which can be very runtime-intensive, especially for large tables. In addition, many application domains contain semi-structured data whose structure changes over time. Such data is difficult to

represent in the rigid data model of a relational database. In addition, the more flexible data model of a graph database often allows it to more directly represent the reality being modeled, making both the design of the data model and the queries applied to it more intuitive. Furthermore, it is more easily adaptable in the event of changes in the application domain, since the graph can be extended without drastic changes in the data model.

The path to a better data understanding

Wherever relationships between data points are the focus of interest, graph databases provide a solid basis for further analysis. They allow a more direct and flexible mapping of the problem domain than relational databases.

Furthermore, they show the way from the collected data to answering the category of questions about the cause of potential or current problems:
Why did this part fail?

- Why are there bottlenecks and price increases in the procurement of parts for production?
- Why does a therapy work better for a patient?
- Why is this server application overloaded? ◆

Use case: Supply chain modeling

Complex supply chains can be mapped and thus potential bottlenecks or dependencies on individual suppliers can be identified.

Use case: Pharmaceutical industry

Mapping the relationships between biological and chemical data can accelerate the development of new pharamceuticals.

Use case: Medicine

Graph databases can show correlations between patients' disease histories and the efficacy of therapies. Likewise, drug interactions can be identified.

Use case: Root cause determination for machine problems

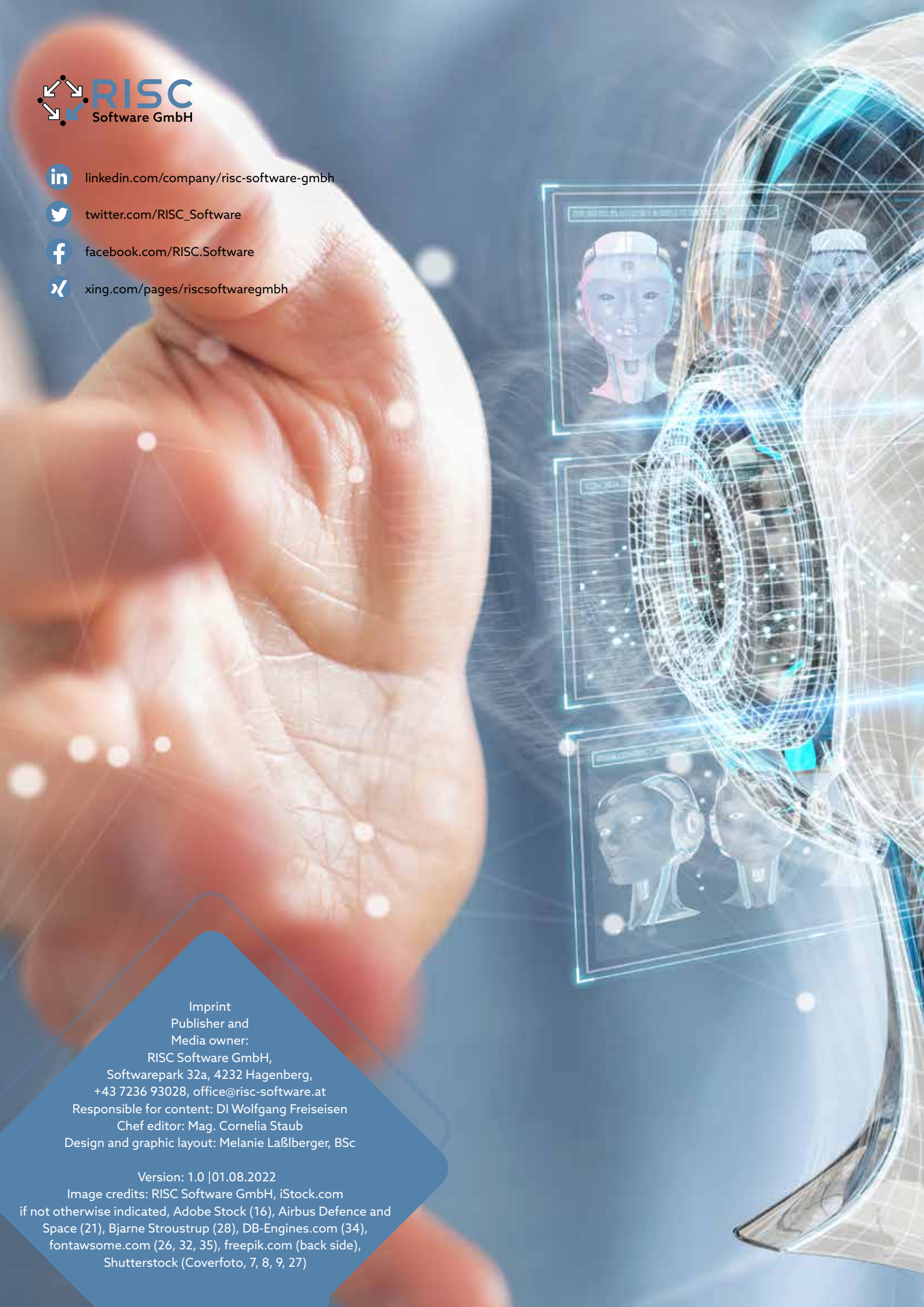
Correlations between sensor values and machine states are often only suspected or unknown. A collection and time-based linking of data in a graph database can reveal hidden correlations.

Use case: Traceability of circuits in complex systems

From traffic control systems to refineries and factories: control algorithms perform automated switching operations. Graph databases can help make them comprehensible to humans, detect errors and increase the efficiency of such plants.

Use case: Monitoring and optimization of IT assets

Parameters of servers and applications can be collected automatically. Based on this, graph databases can be used to detect transitive dependencies between services as well as overloads or critical elements in IT infrastructures.



RISC Software GmbH

linkedin.com/company/risc-software-gmbh

twitter.com/RISC_Software

facebook.com/RISC.Software

xing.com/pages/riscsoftwaregmbh

Imprint
Publisher and
Media owner:
RISC Software GmbH,
Softwarepark 32a, 4232 Hagenberg,
+43 7236 93028, office@risc-software.at
Responsible for content: DI Wolfgang Freiseisen
Chef editor: Mag. Cornelia Staub
Design and graphic layout: Melanie Laßlberger, BSc

Version: 1.0 |01.08.2022
Image credits: RISC Software GmbH, iStock.com
if not otherwise indicated, Adobe Stock (16), Airbus Defence and
Space (21), Bjarne Stroustrup (28), DB-Engines.com (34),
fontawsome.com (26, 32, 35), freepik.com (back side),
Shutterstock (Coverfoto, 7, 8, 9, 27)