

# Quantifying the effects of IT-governance rules

C. Verhoef

*Free University of Amsterdam, Department of Mathematics and Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*

Received 17 January 2005; received in revised form 12 September 2006; accepted 25 January 2007

Available online 14 April 2007

---

## Abstract

Via quantitative analyses of large IT-portfolio databases, we detected unique data patterns pointing to certain IT-governance rules and styles, plus their sometimes nonintuitive and negative side-effects. We grouped the most important patterns in seven categories and highlighted them separately. These patterns relate to the five fundamental parameters for IT-governance: data, control, time, cost and functionality. We revealed patterns of overperfect and heterogeneous data signifying reporting anomalies or ambiguous IT-governance rules, respectively. We also detected patterns of overregulation and underregulation, portending bloated control or no IT-control at all, both with negative side-effects: productivity loss, and too costly IT-development. Uniform management on time, cost or functionality showed clear patterns in the time and cost case, and more diffuse combined patterns for functionality. For these in total seven types of patterns, it was possible to take corrective measures to reduce unwanted side-effects, and/or amplify the intended purpose of the underlying IT-governance rules. These modifications ranged from refinements and additions, to eradications of IT-governance rules. For each of the seven patterns we provided lessons learned and recommendations on how to recognize and remove unwanted effects. Some effects were dangerous, and addressing them led to significant risk reduction and cost savings.

© 2007 Elsevier B.V. All rights reserved.

*Keywords:* IT-governance; IT-governance rules; IT-portfolio analysis; Quantitative IT-governance; Overperfect data; Heterogeneous data; Overregulation; Underregulation; Managing on time; Managing on budget; Managing on functionality; Time compression; Time decompression; Seasonality effects

---

## 1. Introduction

Many organizations are experiencing that information technology is becoming not only a significant expense, but also their largest production means and cornerstone of the organization. This implies for corporate management that a simple cost reduction strategy is no longer the obvious guide to decision making, since their real means to gaining competitive edge is IT. We see the emergence of enterprise architectures, software process and product improvement programs, software productivity benchmarks, business cases, and other indications that corporate executives want more insight in the IT-function. Signs of effort that is put into control are also available in abundance: the development of IT-dashboards and balanced scorecards [30], the installment of IT-portfolio management departments, the initiation of IT-review boards, the development of the IT-investment maturity model [39], and so on. IT-governance amalgamates this into a holistic more formal whole. You could say that IT-governance is a structure of relationships and processes

---

*E-mail address:* [x@cs.vu.nl](mailto:x@cs.vu.nl).

to direct and control an organization's IT-function in order to achieve its goals by adding value while balancing risk versus return over IT and its processes. Part of IT-governance is to design, apply, and evaluate a set of rules for governing the IT-function—the rules by which we play the IT-game. Just to mention an example that everyone is probably familiar with: the enterprise architecture. Once it is established we agree to use it all over the organization. There is no freedom whatsoever for ad hoc IT-investments or application of alternative technology.

But how effective are the controls in reality? Does such uniformity really deliver reduced costs and increase efficiency? And how to sift through all of these methodologies, strategies, models and outputs to ensure real added value and achieving the right balance of governance versus actual production? For instance, the effectiveness of such enterprise architecture groups can be brought seriously into question. Trying to get “all the boats rowing hard but also pointing in the right direction” is a worthwhile goal, but you have pockets of almost religious resistance, and there are truly needed variations [20]. Also, it can't be a one-size-fits-all, but instead a set of guidelines with different levels of commitment. Otherwise the enterprise architects are ineffective, or power struggles occur between developers supporting the business and the enterprise architect's perceived centralized ivory towers. Of course, you don't want to bog down the IT-organization with IT-governance. But what is the right level and combination? Such questions – let alone answers – are not part of most papers on IT-governance.

In this paper we quantify at least the effects of some important IT-governance rules and styles, so that insight is gained in such questions and answers are found in whole or in part. We will focus on issues regarding the five fundamental parameters of any IT-governance flavor: data, regulation, cost, time, and functionality. For, without a fact-based rationale IT-governance is null and void. More specifically seven patterns drew our attention: overperfect data, heterogeneous data, overregulation and underregulation, and a preoccupation with time, cost or functionality. These patterns reveal rules that deliberately or haphazardly style an IT-governance practice which easily induces unwanted negative effects. In each of the seven cases it was possible to take corrective measures to reduce these effects and/or amplify the intended purpose of the chosen direction. To scope the expectations for this paper, there is no best single approach. Instead, by continuous monitoring an IT-portfolio, we can often quickly spot unwanted side-effects of IT-governance, and by conservative modifications we can balance the level of governance to efficiency, the cost of data collection to its value, regulations to their effectiveness, and so on. Finding and keeping such an equilibrium is not easy, but flying blind will lead to asymmetry and instability [7]. In this paper we will not try to find this dynamic equilibrium but to steer away from the extremes to ensure better balance and allow organizations to rightly pursue their own equilibrium.

Of course, IT-governance rules are meant to efficiently and effectively carry out the IT-function. Most rules are a result of common sense, standardization, experience, and best practices. But many organizations lack even the most basic rules. For instance, Meta Group surveyed whether organizations make a business case for information technology. It turned out that [23]:

- 84% of companies either do not do business cases for their IT-projects at all or just do them on a select few key projects.
- 83% of companies are unable to adjust and align their budgets with business needs more than once or twice a year.
- 67% of IT-organizations are not “market ready”—benchmarking is done less frequently than once a year.
- 89% of companies are flying blind, with virtually no metrics in place except for finance (akin to flying a plane by monitoring the fuel burn rate).
- 57% of companies perceive they are balancing the pressures of cost cutting and IT-effectiveness—but perception is not necessarily reality.

In addition, budgeting for one year in itself seems already daunting, and sometimes takes more than five months alone. Frankly, it cannot get much worse than this survey illustrates, so if only there were a few rules, like having a solid business case for IT-investments balancing risk and return, the performance could improve drastically. One such improvement was attempted by the federal government of the United States with their Clinger–Cohen Act [22], regulating federal IT-spending. The basics from this act for major information system investments were summarized by Raines, then director of the Office of Management and Budget. His eight IT-governance rules are used as decision criterion by the government. The first four relate to capital planning, addressing the sore lack of such rules as surveyed by Meta Group. The fifth establishes the link between planning and implementation: the information architecture should align technology with mission goals. The last three rules establish risk management principles to assure a high level of confidence that the proposed investment will succeed (risk reduction is rarely seen in practice). All major

federal information systems should comply with what has become known as Raines' Rules [42]. Such investments shall:

- support functions that the federal government must perform;
- be made because there are no alternatives;
- be preceded by Business Process Reengineering [24,11];
- demonstrate a higher return on investment than if using alternatives;
- be consistent with existing federal information architectures;
- make use of risk reduction strategies;
- be implemented in as small as possible phased, successive chunks;
- employ an acquisition strategy (e.g., sharing risk, promoting competition, tying payments to accomplishments, maximizing commercial technology).

Although some of these rules, and our lessons learned, may seem like open door governance, compliance with this Act and Raines' seemingly obvious Rules turned out not to be easy. This was pointed out by an evaluation of adherence to the Clinger–Cohen Act, or better lack thereof [47]. We quote:

The report released today reveals what we feared the most—that the Administration is not enforcing the laws that Congress passed over four years ago [...] The report shows that 16 agencies neither developed nor submitted IT-management reports that included accomplishments, progress, and identification of areas requiring attention. One quarter of agencies listed projects that deviated significantly from cost or schedule goals. According to the report, agencies are not using sound business procedures before investing in information technology, so they are unable to improve program performance and meet their mission goals.

So once again what is the right level and combination of specific methods for each point made in, say, Raines' Rules? What is the real pay back? Exactly how to implement such issues? It all seems far from trivial. No wonder that the technology research company, Forrester Research claims that [10, p. 9]: “Almost any structure for IT governance will work as long as senior business execs are involved—appropriately”. This observation is understandable, given the deplorable state of IT-governance in the industry. But the crux is in the word *appropriately* as we will argue in this paper: some IT-governance rules turn out to have nasty side-effects. The word *appropriately* should cover roles, rules and processes. Structure as meant by Forrester and others [16,54,10,55], mainly denotes the roles. The summary “almost any IT-governance structure will do” [10, p. 2] pertains to the decision structure that is needed for good IT-governance: to the competencies needed within an IT-government. However, a preoccupation on roles within an IT-government could easily distract us from the exposure of unwanted negative effects that may accompany the explicit or tacit way organizations tend to deal with data, control, time, cost and functionality, the five fundamental parameters for IT-governance.

For the US Government the situation is improving. In the fiscal year 2004, the Office of Management and Budget required agencies to submit business cases for over \$34 billion in proposed spending (57% of the total IT-budget), up from less than \$20 billion in 2003 (38%). For those projects that made an adequate case, the quality of justifications has gone up [9, p. 43]. Total federal IT-spending across the government in 2004 is approximately \$60 billion (\$52 billion in FY 2003 [8,13]).

In a sense IT-governance as a theme and specific rules of thumb like Raines' Rules aim to fit information technology in the framework of good governance. The earlier mentioned balanced scorecard testifies of this [30]. IT is there part of the innovation quarter of the balanced scorecard. Even at the time of writing this paper, IT is often only seen as an innovation instead of gradually contributing to customer interaction, general work processes, worker productivity and customer convenience. Our approach can be seen as an attempt to apply established governance practice from other areas (human resource management, operations research, finance, accounting) to information technology. We illustrate that this viewpoint can contribute to IT, by exploring a rather basic set of data. For that we obviously need data, and this paper deals with organizations, that *do* collect data on IT-projects. We realize that this is not the majority of the organizations, and for those organizations in dire need to gain insight in their IT-function we refer to work where – despite the lack of internal information – decision making can be supported by a quantitative dimension, using industry benchmarks [50,49,52,51]. Collecting data is always preferable since often appropriate benchmarks are not known, and matching what data is present is an inexact science. Nevertheless, approximate benchmarks give you an idea of order of magnitude which is better than gut feel alone.

Table 1  
Summaries of actual and estimated working hours for 150 IT-projects

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Actuals	496.36	5539.70	10 462.76	10 422.49	15 800.74	19 975.33
Estimates	218.47	5487.76	10 257.58	10 187.10	15 491.96	19 968.16

By exploratory data analyses of information residing in large IT-portfolio databases we discovered characteristic patterns for certain IT-governance rules, whether they were stated explicitly, assumed implicitly, and irrespective whether or not they are adhered to. Moreover, we were able to advise on modifications to the IT-governance system in place to reduce or even mitigate some unexpected negative side-effects. We will discuss this in the paper as follows. First we devote our attention to data and regulation, the building-blocks of IT-governance. For both data and regulation, we discuss a number of extremes that we characterize as overperfect data, heterogeneous data, overregulation, and underregulation. These characterizations each take a separate section. Then we turn our attention to three important dimensions of IT-projects: time, budget and functionality. We will identify unique patterns for exclusively managing on time and patterns revealing a style of IT-governance with a preoccupation on managing on budget. This takes two sections. Then a section discusses the much more diffuse patterns of (the more rare) uniform management on functionality. By exploring IT-portfolios from these 7 viewpoints, we discovered sometimes large exposures in IT-portfolios, which could be reduced or mitigated. We finalize each of the above sections with a few important lessons learned. To protect the interests of the organizations involved, this study is made anonymous: the data patterns are simulations of the real-world situations that we encountered. The portfolio's that we investigated stem from various industries: financial, insurance, government, defense, telecom, and others. They contained a mix of different technologies, ranging from legacy-mainframe technology to modern approaches. The seven patterns were not tied to a single industry, albeit that uniform management on functionality was more often found in the systems industry than in other sectors. The seven patterns are illustrated by simulated versions to reveal their most typical appearance. In practice, such patterns can be less outspoken, often since more than a single effect is present in an IT-portfolio.

## 2. Overperfect data

We first turn our attention to data, which is one of the fundamental parameters of IT-governance. We will not focus in this stage on certain kinds of data, but address patterns in data that point to problems in the process of collecting them. In many organizations we encountered erroneous data points, but sometimes, we see the miracle of *overperfect data*, in other words, data that is too good to be true in the sense that the data is such that it suggests a perfect relationship between two or more variables, whereas there might be no relationship at all. True overperfect data is almost never found in applied statistics. On the contrary, we know from statistical practice that data collection is hardly ever fault-free. Routine data that is collected with little individual care, often under time pressure, can easily contain 10% stray values [26, p. 28]. So the chance that we will run into a true overperfect data set is close to zero in an uncontrolled environment—the practical reality in IT-portfolio management. A typical example of overperfect data is when the correlation between actual values and their corresponding estimates is unusually high. If this pattern is found, it is more likely that the estimates are retrofitted.

Often overperfect data is an effect of an IT-governance structure that allows for tampering the data, willfully or unwillingly. In Fig. 1 we depict a typical view of an IT-portfolio sample immediately revealing a case of overperfect data. This view describes the discrepancy between actual hours worked on 150 projects and their estimates. We summarized the data in Table 1. We explain the abbreviations in Table 1, which will come back in other tables as well. Table 1 is the result of a so-called summary statistic (a statistic is just some function of the observed values). Often used summary statistics are the five-point summary statistics made popular by Tukey [48,38]. We use a six-point summary, since we also include the most well-known summary statistic: the mean. The lesser known are the so-called quartiles. Let us explain them briefly. For a start, a *quantile* is any of several ways of dividing your observations into equally sized groups. An example of a quantile is the *percentile*: this divides your data into 100 equally sized groups. Likewise, *quintiles* divide into 5 equally sized groups, and *quartiles* divide data into 4 equally sized groups. You can obtain a fairly good idea of the distribution of your data by dividing it into quartiles [48,38]. We explain the abbreviations:

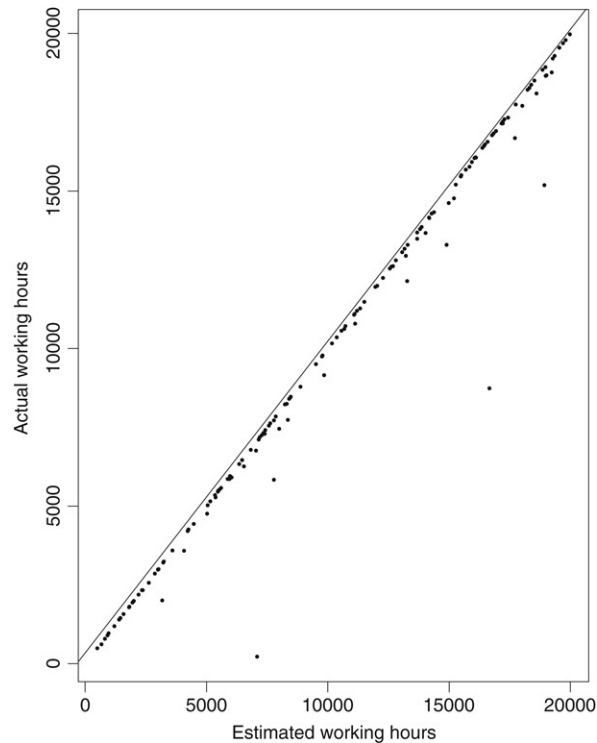


Fig. 1. Visualizing the actual versus estimated working hours for 150 IT-projects.

- Min. stands for Minimum: the smallest observed value.
- 1st Qu. stands for the first quartile. This marks the bottom quarter of the data: one-quarter of the observed values is below the first quartile.
- Median: or the second quartile marks the middle of the data, which is not necessarily the mean: 50% of the data is below that value and 50% is above that value.
- Mean: this is the well-known value that marks the center of your data. If the data has an asymmetrical distribution the mean and median will differ.
- 3rd Qu. stands for the third quartile. This marks the top quarter of the data: one-quarter of the observed values is above the third quartile.
- Max. means Maximum: the largest observed value.

Both from the numerical and visual aggregates it is clear that there is almost no discrepancy: the numerical summary shows large coherence between estimates and actuals, and the view shows this even more strikingly. A special aspect of the view is that there are no serious overestimations and only a few underestimations. The underestimations can be quite large, and they reflect data input by inexperienced managers not yet initiated in the unwritten rule to retrofit the data in the system. So they report actuals without “correcting” their initial estimates. Moreover if we carry out a linear regression (not shown in Fig. 1), we find a correlation coefficient of 0.9742, while a coefficient of 1 would have been a perfect fit.

If the resemblance is less striking, there are other ways to visualize a case of overperfect data. We mention them here. If two data sets have the same distribution, there must be a linear relation between both their quantiles (see e.g., [27, p. 244]). You can visualize that with a so-called Q–Q plot, where Q–Q stands for quantile–quantile. In Fig. 2 we give a Q–Q plot of the estimated versus actual data points. Indeed, this Q–Q plot shows a very nice linear relation, a strong indication that both data sets have the same distribution. Another visualization aid is to compare the cumulative distribution functions of both data sets. We depict that in Fig. 3. This clearly shows that both cumulative distributions are almost the same.

But how much different should data points be in order to conclude that they are not the same anymore? If the effect is less striking than with the data we used to illustrate the pattern of overperfect data, we can use formal statistical tests

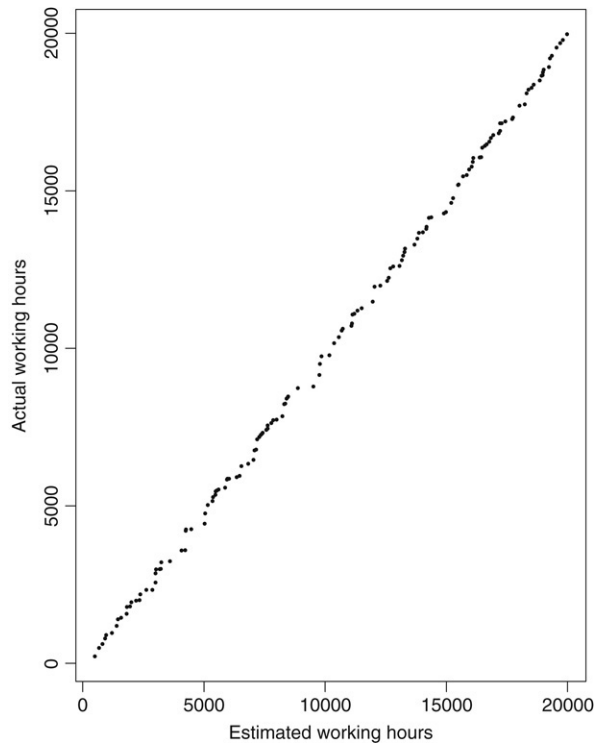


Fig. 2. A Q–Q plot of the actual versus estimated working hours for 150 IT-projects.

to decide whether there is overperfect data or not. Since in general we do not know anything about the distribution of the data, it would be best to use a test where this does not matter, a so-called distribution-free test. One such test is known as the Kolmogorov–Smirnov goodness of fit test, or KS-test for short [35,44,12]. The KS-test measures the maximal vertical distance between the cumulative distributions of two data sets. So, in Fig. 3, the test measures the maximal vertical distance between the solid and dotted lines. This distance is known as the KS-test statistic. For our example the KS-test statistic is 0.0333, meaning the maximal distance vertically between the two distributions is very small. Furthermore, the so-called  $p$ -value (for probability value) is 0.9999. If our null-hypothesis is that the two distributions are the same, we cannot reject that hypothesis with very high confidence (more than 0.001%). Of course, when data is somewhat blurred, the KS-test statistic is often larger, and the  $p$ -value less close to 1. But then you can decide with a standard confidence interval (like 95%) whether the distributions are different, or that you cannot exclude this. If we only have the retrofitted data, we will not learn anything and we lose the opportunity of predictive power, therefore, it is important to explore data for this type of pattern.

In order to develop predictive models you need historical data describing situations similar to the one you wish to have predictive power for. One of the ways to measure the quality of the historical data is to calculate the so-called *estimating quality factor* or EQF, proposed by Tom DeMarco [17] in the early 1980s. The idea is to divide some actual value (that we find in retrospect), by the difference with the estimates that were made before the actual value was known. So, suppose that  $a$  turns out to be the actual value, and  $e(t)$  represents the various estimates over time, and  $t_a$  is the time when the actual value is known. Then the following calculation gives us the EQF:

$$\text{EQF}(a) = \int_0^{t_a} \frac{1}{1 - e(t)/a} dt \quad (1)$$

Eq. (1) expresses that if the estimates are at all times exactly  $a$ , the estimating quality factor becomes infinite. So, in practice, the closer an estimate approximates the actual value, the higher the EQF will become. Indeed, the earlier mentioned overperfect data implies by definition a very high percentage of EQFs approaching infinite, whereas an EQF in the order of 10 (just 10% off) has never been observed by proponents of the EQF-methodology [36]. So it is more likely that if much better EQFs are found that in real-world examples that the data has somehow been adjusted.

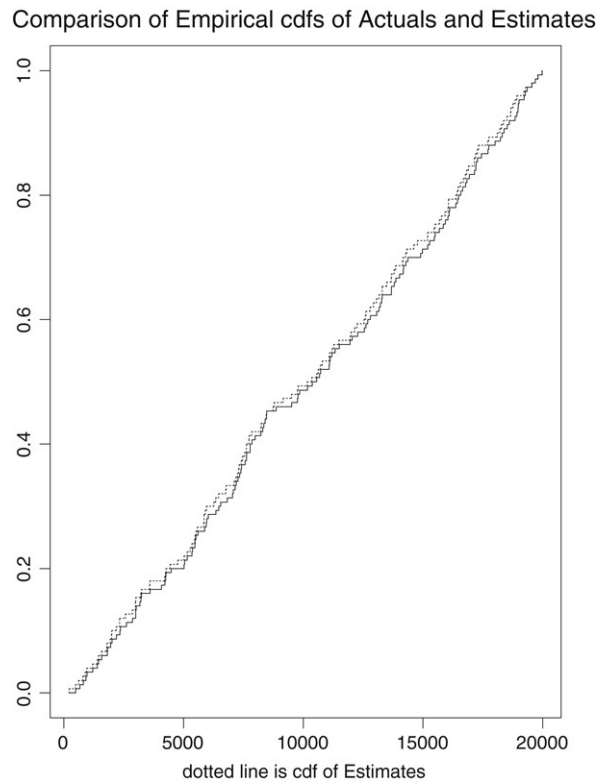


Fig. 3. Comparison of the cumulative distribution of both the actual and estimated working hours for 150 IT-projects.

One of the important data points for IT-portfolio management are durations of a project. Time-to-market is often an important aspect of a project, and control of deadlines can turn out to be crucial for reaping business opportunities. Underestimation of the completion time for projects is endemic, and analysis of real data as opposed to overperfect data, reveals that there is sometimes a fixed factor between actual completion time and the estimated time for a project. For instance, in the late 1970s Augustine [4] found a fairly consistent factor of 1.33 between estimated completion times and the actuals for about 100 official schedule estimates within the US Department of Defense [6, p. 320–1].

**Lessons learned.** Armed with this knowledge, we scrutinized the process of how and when the durations of projects were reported. In this case it turned out that the IT-portfolio reporting system used a destructive database field for project completion, that could be updated at any time. In effect, the managers used the reporting system in two ways: the initial data were keyed in to obtain the necessary approvals from higher management, and upon completion, they used it again to key in the actual data, while destroying the initial estimates. In this way, the historical estimates were no longer easily accessible, making a routine data analysis very difficult. When we recovered the back-ups from tape, we discovered that in some cases the initial estimates were not even made, since the data were not used by higher management. From all this a few important lessons can be learned:

- Never collect data that you are not going to use, because it offers more opportunity to alter data in retrospect.
- Always provide feedback to the people responsible for delivering data. This enables the creation of a regular positive feedback loop. Such a feedback loop is important for mutual risk–reward structures between IT-governors and IT-managers.

To solve the overperfect data problem, a simple change was made to the IT-portfolio management system: all data fields were made nondestructive with the possibility to correct for errors for a certain timeframe, say a week. From the historical data and the actual values, an estimating quality factor was calculated, giving managers more or less credit and freedom depending on their quantified estimating accuracy. In this way, the data that needed to be collected was used, and feedback was given by way of rewarding good estimators. A limitation of this simple pattern is that

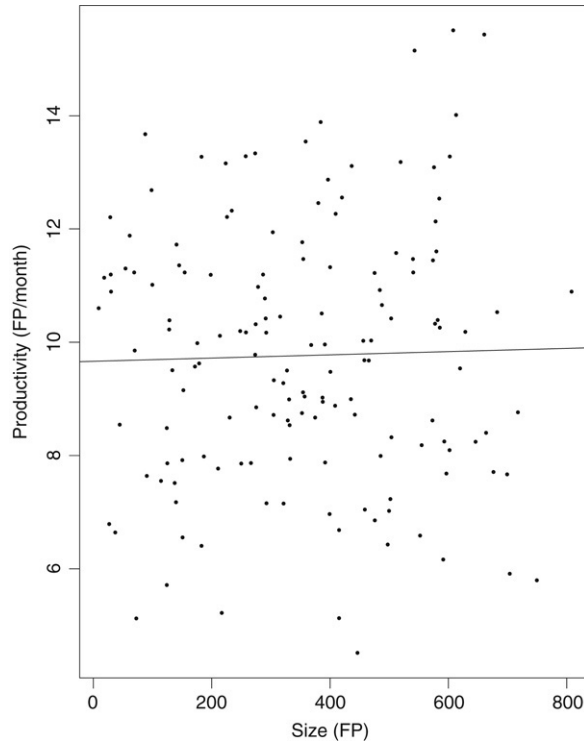


Fig. 4. A seemingly random scatter plot of productivity as a function of software size for 150 IT-projects.

it does not exclude people from booking to the estimate, so that aggregates look great but the data underlying the aggregates are fiction. To detect such patterns we need to analyze the micro-behavior of the cost data in correlation with duration and functionality. This requires sometimes sophisticated statistical analyses like (vector) autoregression techniques, that are out of scope for this paper. For more information on the plausibility of the micro-behavior of important IT-indicators we refer the interested reader to [53].

### 3. Heterogeneous data

We have seen overperfect data, and why it exists. Now we address another pattern in data pointing to problems in the process of collecting them. You might say the opposite of overperfect data, of which the correlations are unusually high. Namely, we also encounter data sets that show no correlations whatsoever, whereas they should. For instance, larger IT-systems often take a longer time to construct, so you would expect a relation between those two variables. But in some IT-portfolios, we cannot infer this from the data. We call such data sets heterogeneous. Often the data is pooled from a number of more homogeneous data sets, but put together they result in a (seemingly) arbitrary data view. Of course, such data can in addition be overperfect, but for explanatory reasons we will discuss heterogeneous data in isolation.

We show an example of seemingly random data in Fig. 4. In this view, we depict for a sample of 150 IT-projects their productivity in function points [1,2,19,34,33,21] per staff month, against their size in function points. From the view we can immediately spot that the linear regression line has no predictive power whatsoever. Indeed the correlation coefficient is 0.0005629, which is close to no correlation (then the coefficient would be exactly zero). In reality, there is a decrease in productivity when the software size increases [6,37,40,28]. For instance, when the size of software increases the number of people working on the project increases, and therefore the communication among the engineers increases, and overall productivity will be lower. But there are also other reasons why inevitably the productivity will be lower for larger systems than for smaller ones. This effect can be quantified: according to benchmark, this is a nonlinear relation. The sample viewed by Fig. 4 comprises mainly of in-house developed MIS

Table 2  
Summaries of productivity and accompanying size for 150 IT-projects

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Productivity	4.51326	8.01788	9.81368	9.76524	11.23224	15.50640
Size	8.9612	189.7929	354.4165	353.9136	501.1054	808.0263

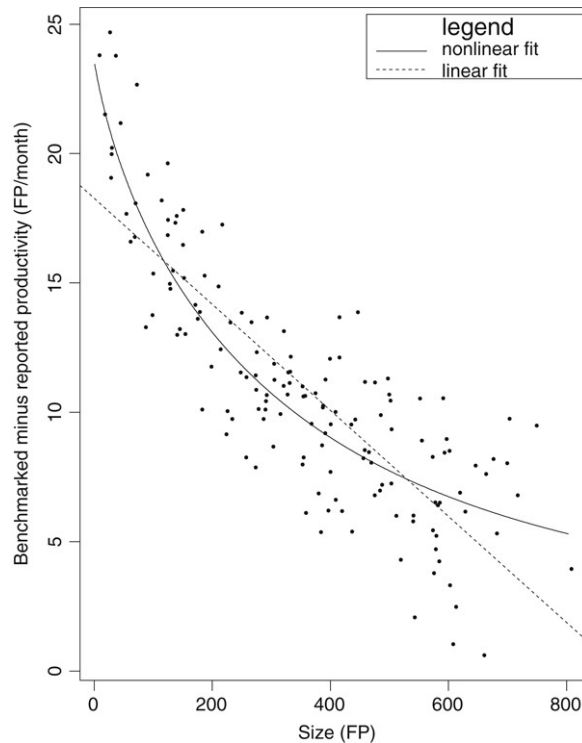


Fig. 5. Difference of benchmarked productivity and the seemingly random productivity as function of software size.

systems. For this type of development we use the following formula taken from [50, p. 61, formula (46)]:

$$p(f) = 1.627 + 38.373 \cdot e^{-0.06222733 f^{0.424459}} \quad (2)$$

This formula takes an amount of function points, and returns the productivity in function points per staff month according to benchmark for that particular software size. This formula is inferred from public IT-productivity benchmarks for in-house MIS development. Since the data does not seem to show any structure, we use Formula (2) as a fall-back scenario.

From Table 2, we learn that the productivity ranges between 4.5 and 15.5 function points per staff month. The mean productivity is around 10 function points per staff month. Moreover, there are extremely small sized projects of only a few function points up to medium sized 800 function points projects. We note that the numbers in both columns are not correlated, they just summarize minimal and maximal values, and a few crucial values in between.

This large range and the view in Fig. 4 are strong signs of heterogeneous data. In order to assess the arbitrarily looking data, we conduct an experiment. We wish to know what the difference is between benchmarked productivity and the reported productivity. The reason for this is, that in this example the function point sizes are reasonable. We note that this is more often the case; measuring the amount of function points is somehow easier than inferring the IT-productivity. The data of the sample portfolio contains tuples of the form  $(p_i, f_i)$ , where  $p_i$  is the reported productivity for a software system of size  $f_i$  measured in function points. To calculate the difference we use Formula (2):  $p(f_i) - p_i$ .

In Fig. 5, we depict the data cloud that results from taking the differences. This plot is clearly an improvement over the random plot (viz. Fig. 4). Next we carried out two curve fitting exercises. A linear regression, represented by the

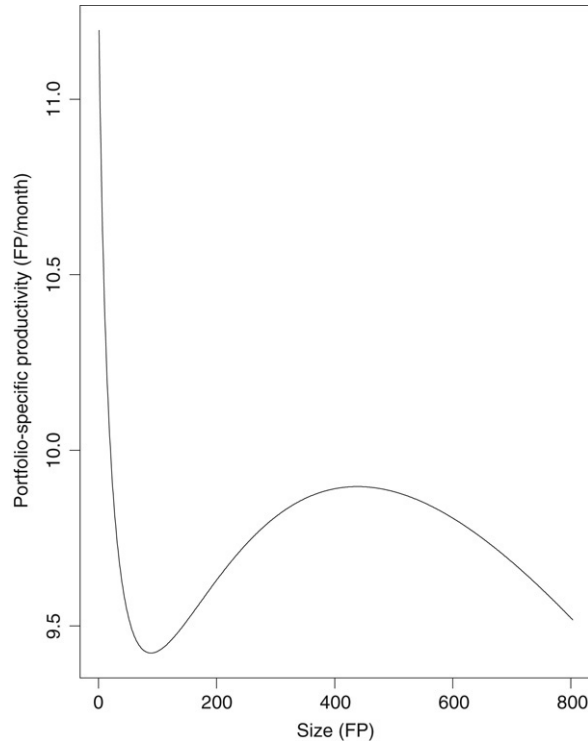


Fig. 6. IT-portfolio specific productivity formula based on public benchmarks and 150 IT-projects.

dashed line and a solid line which is a nonlinear fit. The linear fit has a much better correlation than the earlier fit with the raw data: 0.6889. Still we are not satisfied with the linear fit, since as we will see later on, it will not predict productivity in a natural manner: it will predict higher productivity for larger systems, which is not in accordance with reality. Therefore, we also conducted a nonlinear fit, and we assumed that the difference is similarly shaped to Formula (2). This assumption is based on our experience that productivity benchmarks are behaving somewhat like a logistic probability density function. We do not exclude that other families of curves are not appropriate, but we use the logistic family, which is often giving satisfactory results. This fitting exercise leads to the following formula:

$$c(f) = 2.065365 + 22.93463 \cdot e^{-0.01567961 f^{0.7208611}} \quad (3)$$

The  $c$  stands for *correction*, since Formula (3) describes a correction between the benchmarked productivity and the reported one. Using this correction, we can infer a formula that is specific for the sample IT-portfolio that gives a reasonable idea of the productivity:  $p'(f) = p(f) - c(f)$ .

In Fig. 6 we plotted our newly inferred productivity Formula  $p'(f)$ . What can be seen, is that for very small sizes, the productivity is high, then it slows down very quickly, and after a slight recovery, it slows down again. Investigation of the sample showed that there was a mix of small new development projects, small changes to very large systems, and minor and major enhancement projects. Since only the size of the change was reported, and not the context of the change, the data seemed arbitrary, but is in fact just heterogeneous. Based on an additional qualitative analysis we understood the dip in  $p'$ : these productivity figures represent the projects where relatively small changes were made to very large existing systems. For that category of changes it is known that their productivity is very low. Productivity ranges between 5 and 15 function points per staff month for well structured systems where typically enhancements are implemented without the need for internal modifications to the existing system(s) [28, p. 630], and 0.5 and 3 function points per staff month for the classic form of maintaining poorly structured, aging legacy systems [28, p. 633]. Obviously this kind of low productivity is totally different from the average productivity of 27.80 function points per staff month for in-house developed MIS systems [29, p. 191]. Therefore, the data of Fig. 4 showed such a wide variation that it looked random at first sight.

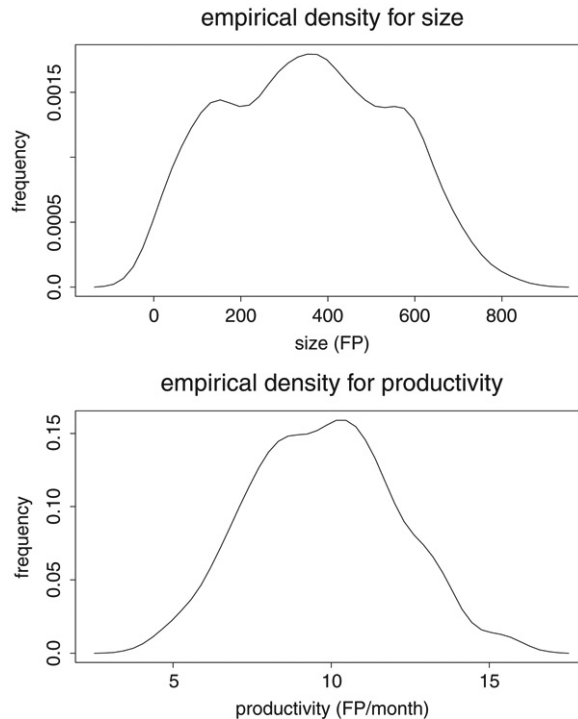


Fig. 7. Empirical density estimates of the distributions for size and productivity for 150 IT-projects.

We note that you can spot potentially heterogeneous data also by estimating the empirical probability density function. If this leads to more local maxima, this is also a sign of heterogeneous data. In Fig. 7, we provide such estimates. As can be seen there are more tops for the function point sizes, and these sizes seem to be the sum of maybe 3 density functions. But in fact, there is nothing wrong with that, since function point sizes often display asymmetric leptokurtic possibly heterogeneous distributions with heavy tails [53]. Also the productivity distribution does not reveal too much heterogeneity. Fortunately, the plot in Fig. 4 leaves nothing to the imagination in that respect: the data is heterogeneous.

As we already announced, we also tried a linear fit to correct for the productivity, but this turned out to be less natural. We compared both corrected productivity formulas in Fig. 8, and this clearly shows that for the larger projects the linearly corrected productivity increases monotonically. This is not in accordance with reality, since larger IT-projects have lower productivity rates than smaller ones. Therefore, we rejected the linear fit, and took the nonlinear fit that also behaved relatively well for the larger function point sizes. Again, also this fit is far from optimal, but at least it gives us insight in the status of the current IT-governance rules. This status is that too many different types of activities are pooled into one type of data, which leads to rather unclear views as depicted in Fig. 4.

**Lessons learned.** After discovery of the probable cause of the randomly looking views,  $p'$  was used for the time being within that business unit until the real solution was implemented. The real solution was to modify the IT-governance rules. These rules did not differentiate between types of IT-projects which was reflected in the reporting system where no difference was made between a development project and an enhancement project for a large existing system. When the rules were changed, the reporting structure changed, too, and we could isolate the various types of projects, and the random data effect was gone. We summarize a few important issues:

- Seemingly random data can result from too coarse-grained data requests.
- Make sure to categorize data so that randomness is avoided in the first place.
- When there are strong indications that the data set is heterogeneous, analyze the data to find the root cause of the effect, and reassess the IT-governance rules by which they are collected.
- In other words, prevent that IT-governance rules induce the pooling of homogeneous data sets into a larger heterogeneous data set, if the latter set hampers further analysis.

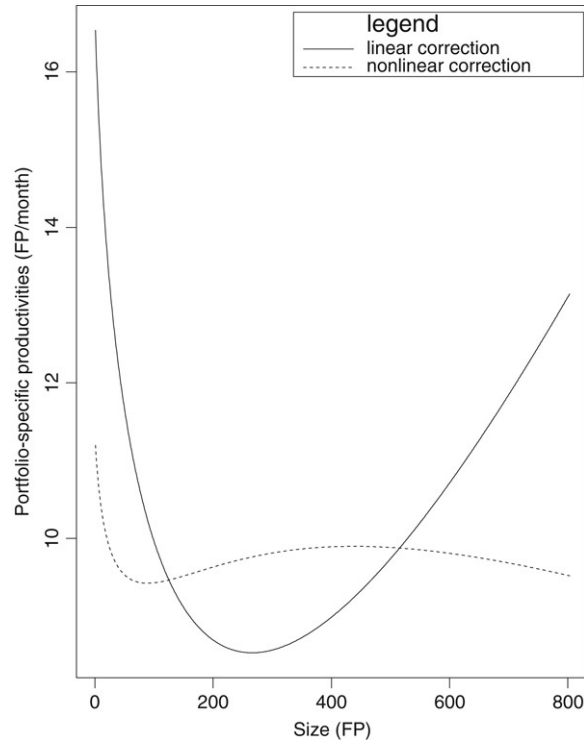


Fig. 8. Two productivity formulas based on public benchmarks and 150 IT-projects.

#### 4. Overregulation

We have seen two typical patterns when it comes to data. At this point we move to IT-governance rules, and dive into the subject of overregulation. Also here, we can see from the collected data what its effect is, and whether this is good or bad. For instance, in one case a benchmark among peers showed a too low productivity for IT-development within a large organization. How was this possible, while everything was perfectly under control? The answer lies in the word perfect, which is sometimes the enemy of the good. The productivity problems were due to overregulation.

As for comparison, we give an example of overregulation outside the software world. The US Government has many rules for all their acquisitions. One of those rules is called TINA, short for *Truth In Negotiations Act*. TINA implies that each contract must be accurate, complete, current, and certifiable. This holds for all cost and pricing data associated with each contract. Of course, TINA is meant to save costs, to prevent unnecessary high prices. But how much does TINA itself cost? The following case sheds some light on this question. After an initial acquisition of half a dozen F16s, six more turned out to be necessary despite the fact that they were declared to be superseded by other planes. The first acquisition was done according to TINA, but for the other 6 F16s a TINA-waiver was granted. This reduced the military specifications with 91%, the data deliverables with 61%, reduced the contract span period from 800 to 195 days, reduced proposal preparation costs with \$1.5 million, and a unit price reduction of \$0.3 million [43]. It is obvious that the rules to save costs, have a cost-increasing effect. When governance rules cost more than they deliver, we speak of *overregulation*. This happens also for software projects, and there are ways to detect cases of overregulation or control mania from data in an IT-portfolio.

One of the key-indicators to look for is the *approval duration* of IT-projects, this is the time (here measured in days) it takes from submission to approval of artifacts demanded by governance, e.g., all kinds of documents. In Fig. 9 we illustrate this. From a sample of 150 IT-projects we depicted the total approval duration for the most prominent deliverables that are submitted, reviewed, and approved during the project. In this case, a feasibility study, a requirements document, change control documents for each major deviation from the original plans, and a document dealing with the closure of the project, whether successful or not: the post mortem. For all but the change control documents, a project could only commence after a sign-off: no significant investment without an approved feasibility

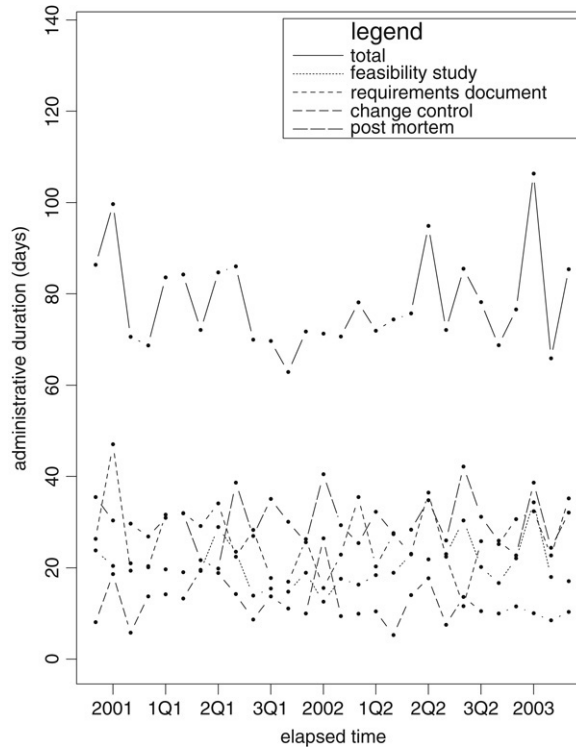


Fig. 9. Approval durations of IT-projects of a business unit from 2001 to 2003.

Table 3

Summary of the approval durations (in days) of IT-projects in a business unit between 2001 and 2003

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Feasibility study	12.59	17.42	19.47	20.31	22.27	34.34
Requirements document	11.59	21.88	26.02	26.36	31.71	47.02
Change control	5.27	9.78	10.79	12.31	14.05	26.43
Post mortem	19.81	25.95	29.86	30.14	32.88	42.14
Total	62.9	70.6	75.0	78.1	84.9	106.3

study, no design and implementation without proper requirements, and no deployment (or retirement) before the post mortem. The change control is not having such sequential impact since when some part needs change, other parts can progress. Still there is a small delay. We composed the total approval duration by adding the approval times of the three, and one-tenth of the change control approval time, thereby conservatively taking the approval duration of intermediate changes into account. As summarized in Table 3 the average total approval duration is over 75 calendar days, or more than 2.5 months of administrative delays. Also on average, the total average project duration was about 380 calendar days, so a little over a year, from which 78.1 days is just over 20%. Note that the row in Table 3 does not contain the sums of the columns, it is the summary of the totals. So, the approval process laid out in the IT-governance rules is responsible for 20% of total IT-project duration. Note, that the *creation* of the documents is not taken into account here. The 20% is only the time between submission and approval (this leads to time decompression, a subject we will discuss in another section). Such long waiting times not only stunt IT-productivity, but there is also an opportunity cost incurred. There is a longer time to market, and staff typically does not shift during unplanned delays. In such a situation, you start wondering whether no IT-governance at all pays off more than this fast-tracked governance (we will come back to underregulation in the next section).

**How bad is bad data?** As an intermezzo we address the question what the effect is of bad versus good data. We conducted several experiments to that end. Our experience was that raw uncorrected data did deviate from much

better data, but that in many cases some side-effect of IT-governance rules was visible with both raw and corrected data. Only the numbers were different, but the order of magnitude was similar. We will describe one experiment to give the reader an idea. The above 20% delay was calculated on raw uncorrected data. For this example we checked *each* data point by manual inspection and corrected faulty data by hand, which was labor intensive (13 person days). After all stray values were removed, erroneous data was corrected, and more, we found a lower average approval duration, but still a 15% delay on average for each IT-project. So, still a case of overregulation. Of course, better data is always preferable, since then analyses can be refined, and more conclusions can be drawn. But better data comes at a cost, so if effects can also be detected with data of less quality, this is more cost-effective. Of course, if you are looking for more refined issues than we are discussing in this paper, such as monitoring the progress of IT-development within a large outsourcing deal, it pays off to spend money on data collection, monitoring and evaluation. We have seen spending of 1 hour per week per 1250 function points of outsourced software.

**Lessons learned.** The 20% delay clarified a significant amount of the difference in productivity that was found during the benchmark study among peers. And with a less time-consuming approval process, much of the low productivity could be clarified. In this case, a few measures were taken: for small projects, the feasibility study and requirements were collapsed into one document, more preformatting was done for the post mortem document, and the requirement that IT should be taken into production only after signing-off the post mortem was abandoned except for the top 5 projects ranked by amount of investment. After that, the approval duration could be brought back to a more acceptable level below 5% of the total durations. All in all, a few important lessons can be learned:

- You know that things are in error when the administrative process around IT-governance rules stunts productivity with more than 10%.
- The time delay due to administration should not exceed a prescribed threshold of the total project time. For very critical projects, special thresholds can be set in addition to an overall threshold.
- Check now and then whether conclusions change if data is sanitized completely. If this is the case, improve the data collection process. Every year one or two important effects that you monitor will do. During audits irregularities will surface, and such efforts should not be carried out more than twice a year, unless there is a direct pressing reason.
- There must always be the possibility to waive IT-governance rules in individual cases, where this can be justified.

## 5. Underregulation

The opposite situation of overregulation is what we call *underregulation*: there are no IT-governance rules. This does not exclude the existence of governance rules at all, just the fact that there are no specific rules to govern information technology. Lack of such rules gives rise to patterns in data collected for IT-portfolio analyses. We will illustrate a few effects testifying of the lack of rules. In fact you could say that people assume certain rules, even if they are lacking, and what we see in IT-portfolio analyses is the reflection of such implicit rules.

In one organization we found the following effect when looking at three important indicators: budget, time to market, and delivered functionality. Due to the lack of rules, technical personnel optimized towards functionality. Both budget and time to market were sacrificed for delivering the full solution. Furthermore, budget took precedence over time to market: first, deadline extensions were used to optimize to full delivery, and only as a last resort more budget was asked for. You could say that the IT-department optimized towards quality—an attribute that characterized the mission and strategy of this organization.

We know that optimizing to functionality is only important in a few cases, for instance, when obligatory changes to existing systems have to be made, you cannot afford yourself to deliver 80% of the solution. But in some other cases, the speed to market a solution is more important than having all the functionality, for instance to ascertain a market share. In yet different cases budget is a leading indicator for creating a business case: the perfect solution is way too expensive, but a partial solution often creates enough value for money.

An interesting phenomenon that is often seen in organizations lacking IT-governance rules is what is called *seasonality*. Almost all organizations have financial governance, and this has been traditionally organized around years and quarters. Some governance rules are also organized around fiscal years, that do not need to coincide with calendar years, but the effect is the same. The effect is that information technology becomes automatically organized around the financial “seasons” in this case.

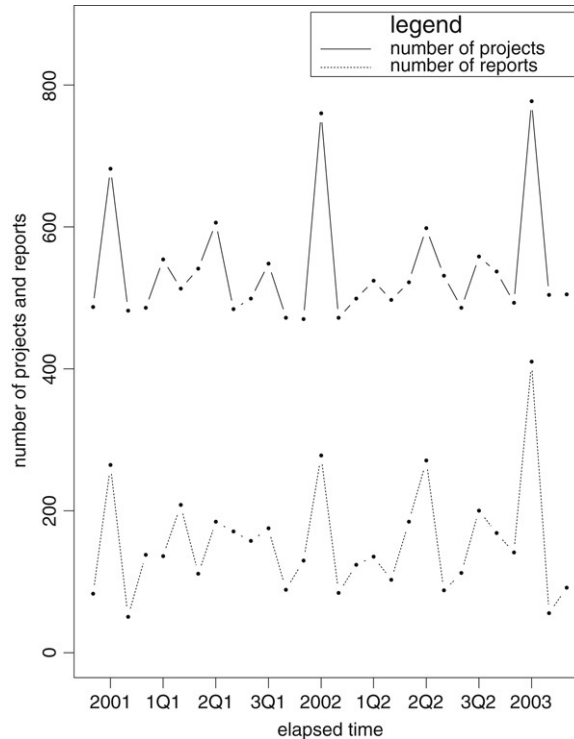


Fig. 10. Amounts of IT-projects and their reports of a business unit from 2001 to 2003.

Table 4

Summary of the number of IT-projects and the generated reports in a business unit between 2001 and 2003

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Number of projects	470.0	486.8	509.0	538.8	549.5	777.0
Number of reports	50.3	99.9	136.9	155.1	184.5	409.9

This implies for IT that project management, budget allocation, and review processes are also organized around the seasonal deadlines that apply for general governance. We can see such effects in IT very clearly. For instance, in Fig. 10 we depicted two lines: the solid line is the number of ongoing IT-projects, and the dotted line is the number of accompanying reports about the projects. In Table 4 we summarized the data for both the projects and their reports. The long-term means of both time series show that there are on average 500 + IT-projects and about 150 reports, at each time. However, from the lines in the figure we can see that there are short-term peaks around *magic dates*, like January 1st, July 1st, and smaller peaks around the 1st and 3rd quarter. We can clearly spot the effects of seasonality here.

**Lessons learned.** At this point the reader might ask: “So what? You need to govern IT as other artifacts, so what’s wrong with this?” This is a very legitimate question and we elaborate on this later on. For now it suffices to say that we can detect lack of IT-governance rules in various ways, and the typical patterns of seasonality are among them. Moreover, the certainty of no explicit IT-governance rules is that some implicit set of unwritten rules will be applied regardless the nature of the problem. This probably will not lead to an optimal IT-investment strategy. In summary:

- General governance rules do not take the immature nature of information technology into account, with all its potential consequences.
- For the key indicators budget, duration and delivered functionality, often a uniform order of precedence is culturally inclined, which is usually not the best approach. For instance, an organization that positions itself in the high-quality

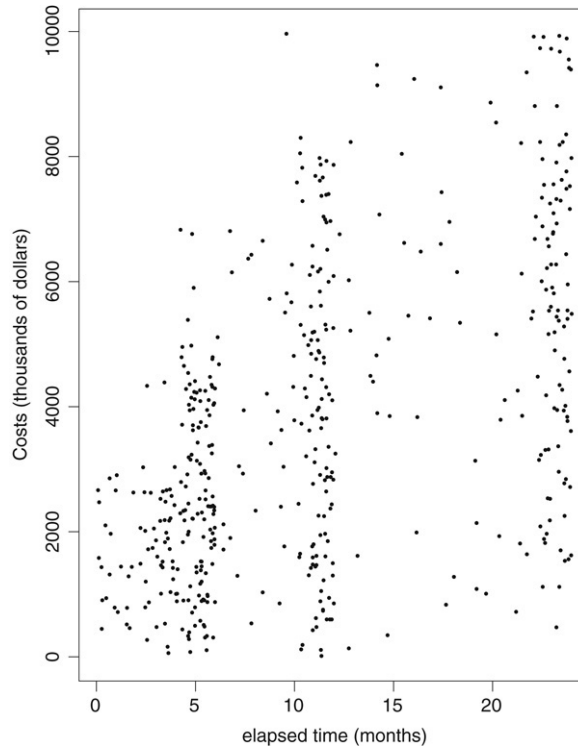


Fig. 11. Cost–duration view of a sample IT-portfolio containing 495 projects, at a total cost of 1.8 billion dollar.

range, is inclined to sacrifice budget and cost constraints in favor of functionality, also when this is not the best choice.

- Lack of IT-specific rules often leads to following the financial regime within the organization, which can be detected by seasonality effects. Of course, it is not a bad thing to subject IT to financial accounting, but it should not hamper productivity.

We do not recommend governance around financial seasons, since they often emphasize on deadlines for delivery of financial data, which transposes to IT-delivery. We do however, recommend to take decisions on IT-portfolios on a regular basis, rather than a few projects each time. A formal submission process enables governors to give IT-projects a different priority, or to weigh projects against each other based on different criteria to obtain an optimized IT-portfolio with a proper risk/return and according investment policy.

## 6. Managing on time

We already alluded to the notion of seasonality, and more general there can be IT-governance rules that take the aspect of time into account. One of the things that keeps coming up is time-to-market. There is often a strong pressure on as short as possible durations of IT-projects. In fact, this is often viewed as a cost-control. But as we will see, managing on time can also increase costs. We can spot those effects in data collected during IT-portfolio analyses, and we can also measure the effect this has on the IT-budgets.

In Fig. 11 we depicted a cost–duration view of a sample IT-portfolio containing 495 finalized IT-projects, with a total investment of \$1.8 billion. On the vertical axis we set out cost in thousands of dollars ranging from IT-projects of an insignificant cost to large IT-investments in the millions of dollars. Horizontally, we depicted the absolute duration over time of these projects, but we refrained from giving actual time series: we start with 0 which represents January 1st, of some recent year. In this sample we see vertical concentrations of data. Such data clouds are characteristic for managing on time.

To obtain a better view of such data clouds it is a common technique to depict the same data on a log–log scale. We have done this in Fig. 12. The difference between Figs. 11 and 12 is that the logarithm of the cost, and the logarithm of

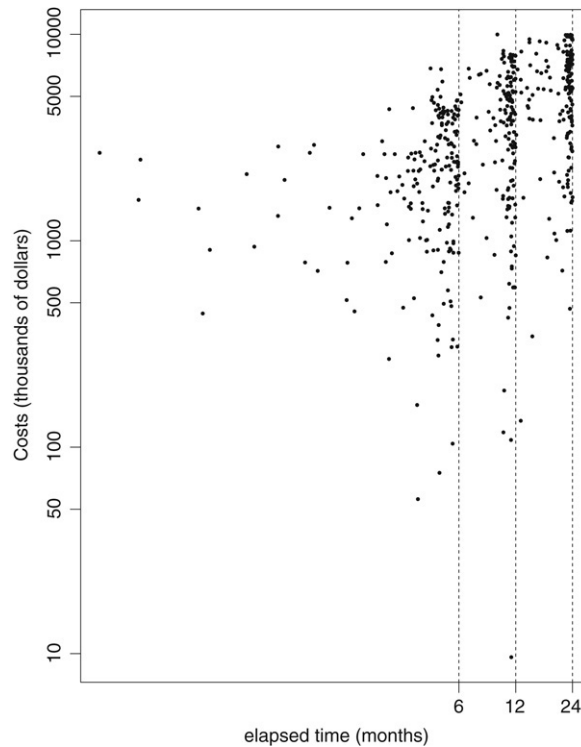


Fig. 12. Log–log view of the same sample IT-portfolio as depicted in Fig. 11.

the time are plotted. The vertical and horizontal scales are therefore not linear anymore. A logarithmic scale clusters data points having the same order of magnitude. We put three vertical dotted lines in Fig. 12: we call them tear lines. As can be seen, a lot of projects have fixed deadlines: 6 months, 12 months, and 2 years. In this case, when these deadlines were not met, it turned out that they jumped to the next tear line.

Another issue that is worth mentioning is that per tear line there seem to be almost arbitrary costs. To that end, we clustered the data around the three tear lines and summarized the result in so-called box and whiskers plot [48,38], or abbreviated a boxplot. We depicted three of them in Fig. 13. A boxplot is just a visual form of a five-point summary. The shaded box is limited by the first and third quartile, and the white line inside the shaded box is the median so that skewness of the data can be spotted right away. So the shaded box encloses the middle 50% of the observed values. Extreme values (minimum and maximum) are highlighted by so-called whiskers and, if present, outliers are shown as well. From these boxplots, we indeed see that for very small timeframes there is an enormous variation in the cost range. Moreover, this cost range is not wide due to outliers, or potential outliers: the middle half of the data is responsible for the large spread.

After we found this data pattern, we discussed it with the involved organization to learn more about their IT-governance rules. It turned out that they managed heavily on time: everything they were doing was time-critical and it was of the utmost importance to market applications in short timeframes, so that customers could be served and profits be made. A question that we worked on subsequently, is the following: what is the trade-off between speed-to-market and IT-development costs? In other words, a shorter timeframe for an IT-development project brings additional costs, but this also enables earlier usage of the software, which can bring in profit (or market share for that matter). The idea was to quantify the extra costs, so that management could make a calculated decision on the benefits of time-to-market.

**Time compression.** Time compression of a software project is doing more work in a time frame than you would normally do from a pure technology viewpoint. We can quantify the cost of time compression using the following relation between time and effort, which is taken from [41,40]:

$$c \cdot d^{3.721} = \text{constant} \quad (4)$$

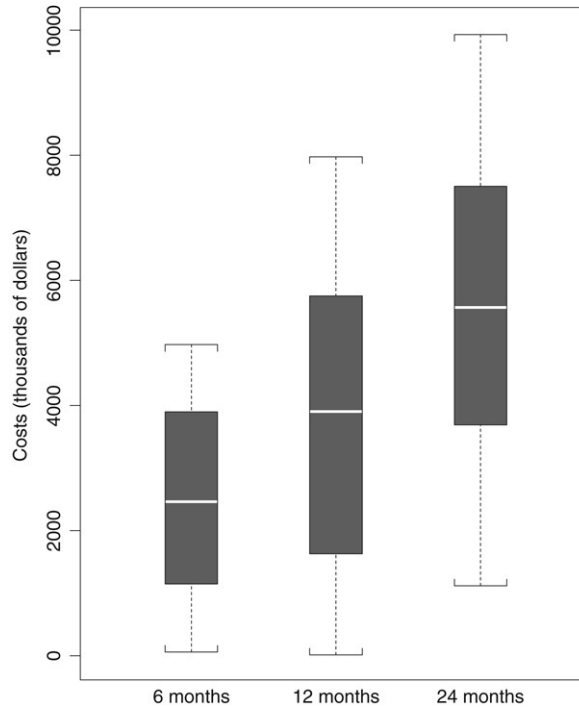


Fig. 13. A boxplot of clusters of data clouds around the three tear lines of Fig. 12.

where  $c$  stands for cost, and  $d$  is the duration of an IT-project. We use this empirical relation between time and cost to estimate the effect of a shorter time-to-market than a normal technology-driven deadline would cost. Eq. (4) indicates that when we try to *compress* time just a little bit, the *pressure* on the cost increases drastically. This is similar to fluids, where a minimal compression of its volume results in a significant increase of its pressure. Therefore, we sometimes refer to Eq. (4) as the *hydraulic software law*. A reasonable range for Formula (4) is that durations do not deviate more than 35% from nominal values; these can be found using your own data, or estimated via public benchmarks.

In Fig. 14, we depicted the effects of time compression. Let's explain this. Suppose there is an IT-development project that costs a little over a million dollar (\$1037174 to be precise). And suppose that the time-to-market for this system should be 12 months according to the business. We depicted this IT-project with a single dot in a cost–duration view; see Fig. 14(a). For this IT-project, we calculate the constant of Formula (4). This amounts to:

$$10751895005 = 1037174 \cdot 12^{3.721}.$$

Now we can vary for this IT-project the cost and time along the line that is defined by the following function:

$$h(t) = 10751895005 \cdot t^{-3.721}$$

The abbreviation  $h$  is short for hydraulic, and quantifies the effects of taking more or less time for an IT-project on the costs. In Fig. 14(b) we depicted  $h$ , together with the dot representing the 12 month IT-project. Next we recall a formula taken from [50, p. 18]:

$$tcd(d) = \frac{rw}{1800} \cdot d^{3.564} \quad (5)$$

where  $tcd$  is short for *total cost of development*,  $r$  is the daily burdened rate in dollars,  $w$  is the number of working days per annum, and  $d$  is the duration of an IT-project in calendar months. Formula (5) is based on public benchmarks, and thus gives an indication what an IT-project will nominally cost. In this case, we had internal data, so we did not use this formula, but another one. Basically, the formula above is constructed like this:

$$tcd(d) = \frac{rw}{12a} \cdot d^{3.564} \quad (6)$$

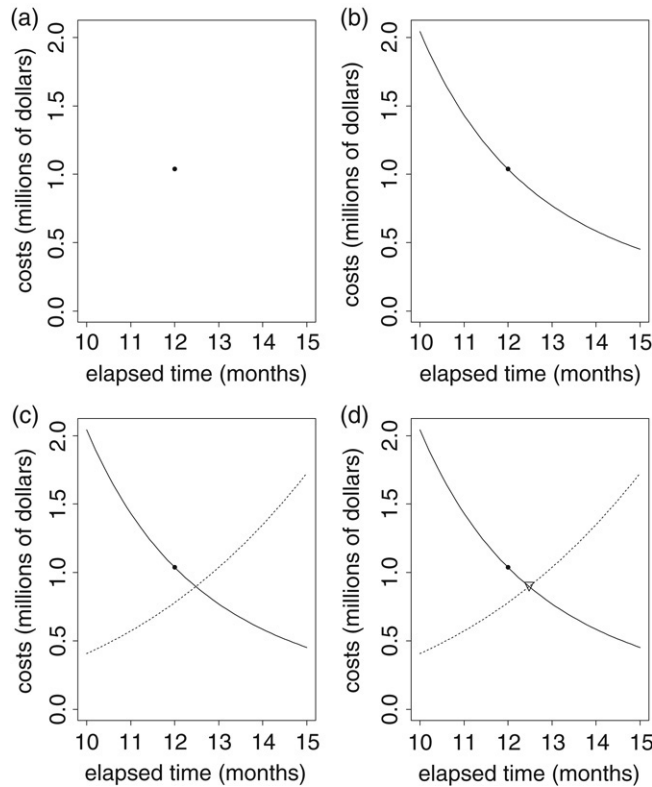


Fig. 14. An example of time compression.

where  $a$  in Eq. (6) is short for assignment scope. An assignment scope for a certain activity is the amount of software (measured in function points) that you can assign to one person for doing that particular task. Note that the assignment scope is relatively size independent. In Formula (5) we took  $a = 150$ , representing the activity: IT-development. If we set  $a = 750$ , this is for the activity: maintenance. Both numbers were taken from [28, p. 202–203]. Since we had data, we could carry out a regression to estimate  $a$ , which turned out to be a value between 150 and 750. This reflected the organization’s governance rule that no distinction was made between new development, maintenance and renovation projects. For now it is only important to realize that you can come up with some internal benchmark with which you can compare IT-projects. For the sake of the example we use an industry benchmark to explain the methods we are using. So, in Fig. 14(c) we added a dashed line representing Formula (5), to the hydraulic line determined by the IT-project visualized by the single dot.

To answer the question what this IT-project would cost if the deadline was according to public benchmarks, we have to combine Formulas (4) and (5): we have to travel alongside the hydraulic line  $h$ , until we meet the benchmark line  $tcd$ . The intersection is marked with an open inverse triangle—see Fig. 14(d). In this case we have analytic relations, so that we can easily find an algebraic solution for the duration  $d_0$  that is both on line  $h$  and according to benchmark. It is the following relation:

$$d_0 = \left( \frac{1800 \cdot constant}{r \cdot w} \right)^{0.1372684} = 12.47924. \tag{7}$$

In Fig. 15 we depicted an exploded view of Fig. 14(d), in order to visualize the time–cost trade-off more prominently. As can be seen, the original deadline of 12 months should for the price of about a million dollars be a little longer, namely about half a month longer. To calculate the cost reduction, we calculate  $tcd(d_0) = 896544.1$  dollar, so that the reduction is 140629.8 dollar, so approximately \$141K. The horizontal line segment in Fig. 15 represents the half month time delay, and the vertical line segment in Fig. 15 stands for the potential \$141K cost-reduction. Of course, you should then not work with fixed staff, since then delay equals added cost. Apart from the

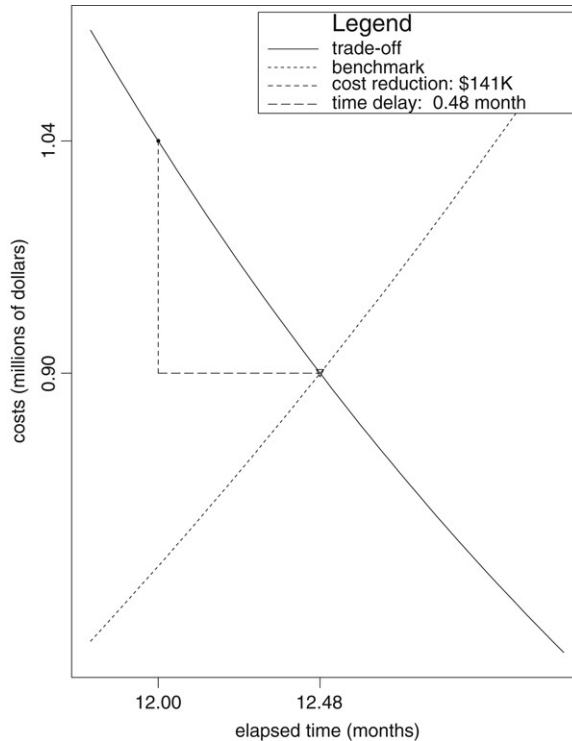


Fig. 15. Exploded view of Fig. 14(d), illustrating how to calculate cost reduction at the cost of some time delay.

financial calculations for time compression and decompression, you must deploy proper capacity management, so that the relaxed time frame, and its freed effort is moved to other projects.

With this result, the business can reflect once more on their business case. Namely, is an extra cost of far over a hundred thousand dollar justified, for a time reduction of a small half month? If this is the case, then the IT-project must make more than \$141K in two weeks or some other strategic goal must be met, e.g., to prevent permanent loss of market share as a consequence of not being the first mover. And can we deploy the extra effort on another project? Summarizing, at least now we can make a calculated consideration when it comes to speed-to-market, and what this brings as benefits to the business.

Of course, it is interesting to look at individual projects, and make such time–cost trade-offs. But what happens if we apply the above exercise to an entire IT-portfolio that is managed on time? In Fig. 16 we depicted a sample of 50 IT-projects from a business unit within an organization. The dots above the dashed line are the actual project data. The solid line segments represent the same solid curves as in Fig. 14. They are the hydraulic lines that belong to the original data points of each individual project by applying fifty times Formula (4), and finding the  $h$  function that belongs to the particular project. The dashed line is again Formula (5), of which we only need one: this is a line that represents what the relation between cost and duration is according to public benchmarks. Now if we apply Formula (7) fifty times, we find all the intersections of the time-compressed IT-projects with their benchmarked estimates. This leads to the fifty intersection points on the dashed benchmark line.

Next we calculate all 50 time delays, as depicted in Fig. 15, and we calculate the 50 cost reductions as well. In Table 5, we summarized the data points, and their totals. We see on average that the time compressed schedule is 17.58 months, and that the total effort in time is 879 calendar months. The benchmarked durations are a bit longer, since this IT-portfolio sample is clearly time compressed, and take 19.68 months on average; an increase of 2.1 month per project, for 50 projects. This amounts to an increase in the number of calendar months of 105 over the 50 projects. For the total of 984 calendar months this is an increase in schedule of 11.95% at the IT-portfolio level. Likewise, the planned cost of these projects is on average about 8 and a quarter of a million, whereas the benchmarked costs are much lower: \$5.3 million, so a decrease on average of 2.96 million dollars per IT-project. Of course, the results in the

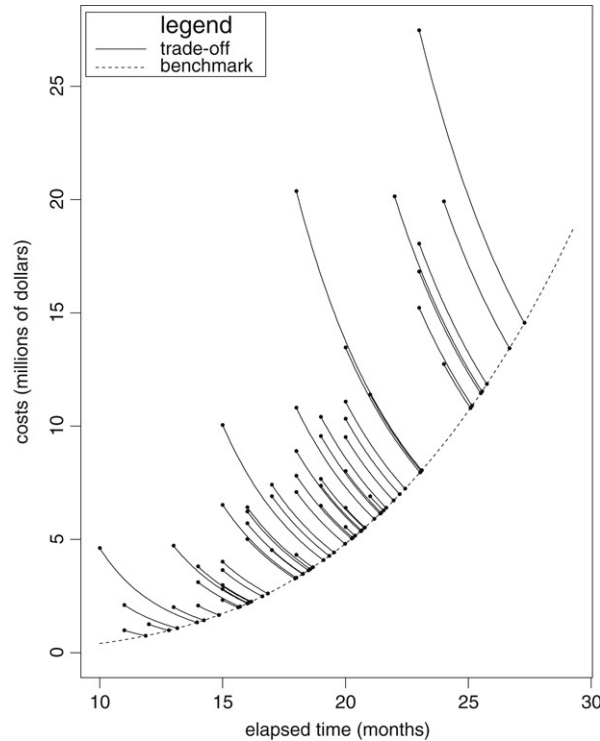


Fig. 16. Sample IT-portfolio of 50 time-compressed projects plus alternative benchmarked scenarios.

Table 5  
Summary of the 50 time-compressed IT-projects and their benchmarked alternatives

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Total
Compressed time <sup>a</sup>	10.00	15.00	18.00	17.58	20.00	24.00	879
Benchmarked time	11.86	16.67	20.12	19.68	21.88	27.28	984
Time delay	0.396	1.264	2.147	2.100	2.606	5.103	105
Planned cost <sup>b</sup>	0.99	4.37	6.91	8.26	10.39	27.47	41.32
Benchmarked cost	0.75	2.52	4.92	5.30	6.64	14.56	26.50
Cost saving	0.24	1.06	2.42	2.96	3.51	12.92	14.82

<sup>a</sup> Time in calendar months.

<sup>b</sup> Cost in millions of dollars.

rows for time delay and cost saving are not calculated by subtracting the two rows above them, these numbers are the result of a standard summary statistic on the differences for all the data.

All in all, the IT-portfolio cost is a little over 41 million dollars, its benchmarked cost – at the expense of longer schedules – is 26.50 million. We call the difference between actual and benchmarked costs the *IT-portfolio time compression risk*. This time compression risk amounts to almost 15 million dollar, which is 35.86% of the total IT-budget. In this case, time compression forms a severe exposure for the organization. Once the indication for a serious time compression exposure is given, we have to assess individual time-compressed IT-projects to see whether the time-constraints can be relaxed such that they approximate nominal schedules.

**Lessons learned.** Of course, not all the IT-project schedules can be relaxed as indicated in Fig. 16. But, the assumption that *all* IT-projects are time-critical, is also an extreme point of view. Often this eagerness to put (too much) pressure on schedules for IT-projects is performed via governance rules. A well-known rule is to only provide money for 12 months, and if you need more, ask after a year for more. As a consequence, people cram – whatever it takes – a project in a timeframe of 12 months, with significant time compression as a consequence. If this rule is

applied organization-wide, a significant time compression risk is the consequence. In addition, budget for next year's maintenance and operational costs is almost never in the business case, see [50] for estimating operational costs of IT-investments, and [51] for how to quantify the value of an IT-investment where operational costs are taken into account.

Also the sample IT-portfolio depicted in Fig. 15 clearly indicates the potential to create havoc in the IT-budget. Much more resources are necessary than anticipated, or with just a little more time we can deliver the same functionality for a much lower cost. In the case described, we advised to change the IT-governance rules slightly, so that default time-critical IT-development was abandoned. Instead all projects were assessed whether they were time-critical or not. In case they were not, other governance rules were used to monitor progress, and in case there was a time-critical component, the costs of speed-to-market were weighted against the benefits of having earlier access to an operational system. From these analyses a few important lessons can be learned.

- Be very careful with IT-governance rules that rely on uniform time constraints, like budget for a certain timeframe, since it can lead to endemic rushing inducing a large time compression risk.
- Make sure to install IT-governance rules that assess the time-criticality of IT-investments, and if time plays a crucial role, think of ways to mitigate time compression risks as much as possible. For instance, think of starting earlier, taking more time, phased development and deployment, etc.
- Regularly scan IT-portfolios for potential time compression risks, so that you can take immediate action.
- A common cause for time compressed projects are the nondiscretionary ones; projects that are regulatory, unavoidable. In many cases there is no executive attention for such issues, since they do not require strategic decision making. But this is not entirely true. Namely, a consequence of lack of executive interest can cause a lack of executive support. This leads to procrastination of the work and only if no time-delay is possible anymore, the technical staff can attract the attention for these mandatory IT-projects. But then it can be too late to carry them out without time compression. Such projects often turn out to be very expensive, and the IT-staff is blamed for it. But in fact this is due to unnecessary time compression.
- Do not make the common mistake to take nondiscretionary IT-projects off-radar. Often cost avoidance is possible by preventing time compression effects.
- Put a uniform maximum to the amount of death-march IT-projects. These are mission-impossible projects with a serious time compression exposure, where schedules are compressed in the order of 25% or more. Of course, you'd want to avoid this category altogether, but we all know that such projects pop up more than is good for any organization.
- Put a uniform maximum, like a percentage of the annual IT-budget, on the time compression exposure that is acceptable at the IT-portfolio level. This maximum will differ from organization to organization, but in general 40% is a very serious risk for almost every organization.

## 7. Managing on budget

Another uniform way of managing IT is nowadays often seen: managing on cost. While in some cases this may resort the correct effect, cost-leadership can have negative consequences, as we will show in this section. Just like managing on time, we can detect managing on cost by cost–time views. In Fig. 17, we depicted a case of managing on budget: we see clouds of data around various cost-bandwidths. To improve the view, we also plotted a log–log view so that numbers with the same magnitude clutter together. This is clearly visible in Fig. 18. As can be seen, the costs asymptotically approach certain magic lines, and then leap to the next level where another asymptote is found.

We collected the data clouds, and depicted them via boxplots in Fig. 19. There we see a large spread of different time intervals, that are not in accordance with nominal time intervals for IT-projects of such price tags. One of the possible effects of managing on cost, is that some projects are systematically underfunded. When you do not put enough resources into an IT-project in the end it will cost more than with enough resources. We call this effect *time decompression*. Again, a limitation of this simple pattern is that it does not exclude people to game the system, so that aggregates look great but the data underlying the aggregates are arbitrarily scattered. We recall that in order to reveal such problems we need to investigate the microscopic properties of the underlying variables with more complex statistical techniques, like vector autoregression. More information on such analyses in the realm of IT-audits can be found in [53].

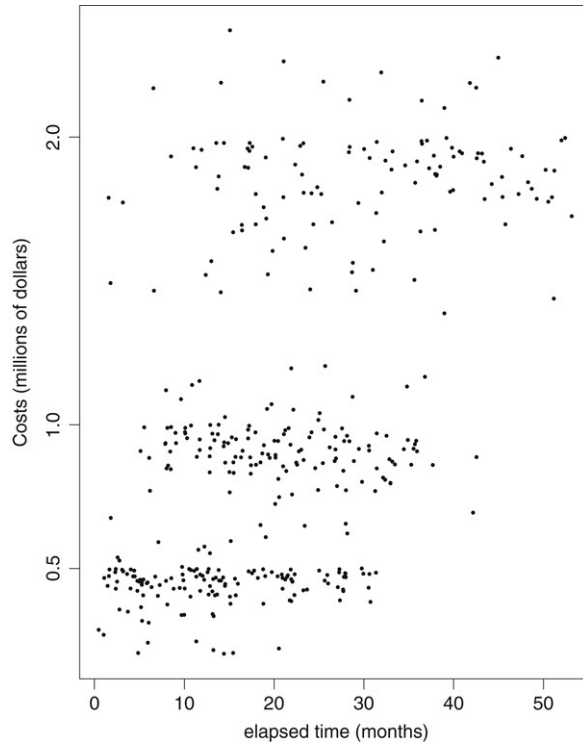


Fig. 17. Cost–duration view of a sample IT-portfolio containing 400 projects, at a total cost of 286 million dollars.

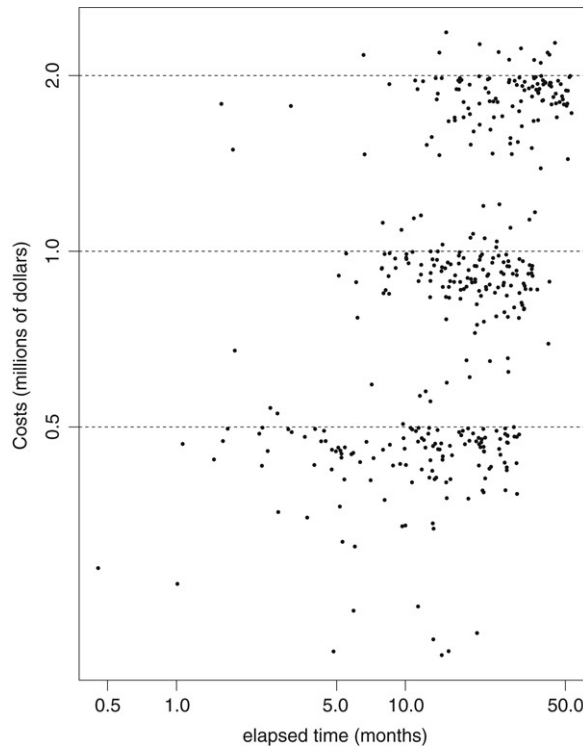


Fig. 18. Log–log view of the same sample IT-portfolio as depicted in Fig. 17.

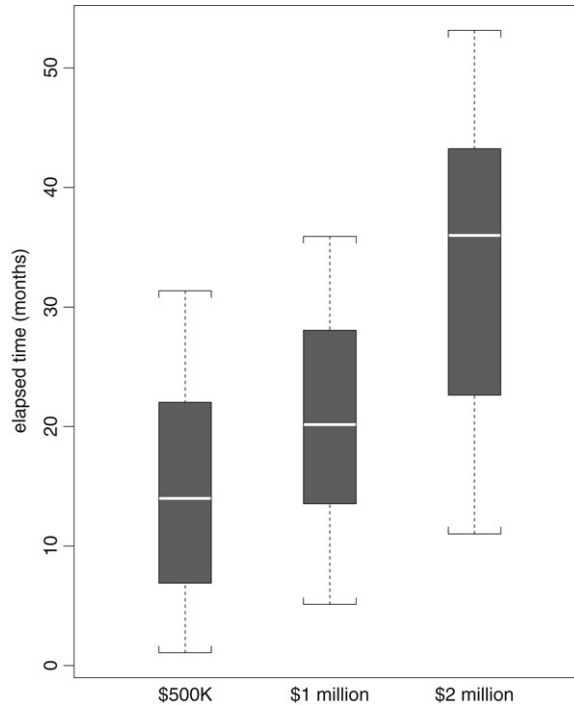


Fig. 19. A boxplot of clusters of data clouds around the three tear lines of Fig. 18.

**Time decompression.** Time decompression of a software project is doing less work in a time frame than you would normally do from a pure technology viewpoint. According to Eq. (4), stretching out the time will lead to lower costs. We believe this to be true, but not indefinitely. Since taking much more time will lead to wasting time and losing knowledge, which in turn will boost the costs. In [6, p. 472] we can find several empirical relations between time and effort for off-nominal schedules. In the time compression case, we took the worst case, which is Eq. (4). For time decompression we also take the worst case. This is the simple relation that when the schedule stretches out  $x\%$ , the effort increases with  $x\%$ . So we can quantify time decompression as follows:

$$c = \frac{c_0}{d_0} d \quad (8)$$

where  $c$  stands for cost, and  $d \geq d_0$  is the stretched duration for the IT-project. Furthermore,  $c_0$  is the nominal cost, and  $d_0$  is the nominal duration. For the nominal relation between cost and duration, we take Formula (5) (but you can also take another, e.g., internal benchmark for nominal cost–duration relations). This leads to the following relation:

$$c = d \cdot \frac{1800c}{d_0} = \frac{drw}{1800} \cdot d_0^{2.564}. \quad (9)$$

So, suppose we carry out a million dollar project in  $d = 14$  months. Then, we can calculate the nominal duration, and nominal cost as follows. Using Eq. (8) we find:

$$c_0 = \frac{c}{d} d_0.$$

By substituting  $c_0$  in Eq. (5) using the above we find:

$$d_0 = \left( \frac{1800c}{drw} \right)^{0.39}$$

which gives us a nominal duration of 12.44987 months. The nominal costs that belong to this duration are: \$889276.4, using Eq. (5). So, on this project we can save some time: 1.55013 months, and some cost: \$110723.6. Note that for

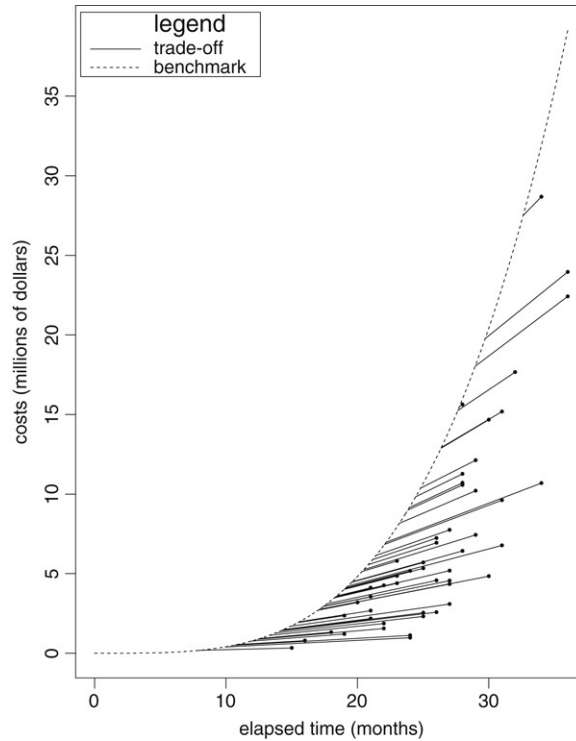


Fig. 20. Sample IT-portfolio of 50 time decompressed projects plus alternative benchmarked scenarios.

Table 6  
Summary of the 50 time decompressed IT-projects and their benchmarked alternatives

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Total
Decompressed time <sup>a</sup>	15.0	22.2	26.0	25.8	28.8	36.0	1289
Benchmarked time	7.9	15.1	18.7	19.1	22.2	32.6	953
Time gain	0.2	4.2	5.7	6.7	8.9	14.0	336
Actual cost <sup>b</sup>	0.3	2.6	5.0	7.1	10.1	28.7	353
Benchmarked cost	0.2	1.8	3.8	5.7	6.9	27.5	283
Cost saving	0.14	0.68	1.25	1.39	1.74	4.38	69.3

<sup>a</sup> Time in calendar months.

<sup>b</sup> Cost in millions of dollars.

the sake of ease we refrained from discounting time-value for money. More involved calculations appraising IT-investments that do deploy discounted cash flows are presented elsewhere [51].

We can make such calculations at the IT-portfolio level as well. In Fig. 20, we give an example of an IT-portfolio that is managed on budget. The dots to the right of the dashed line are the actual data points: real cost, and duration. The dashed line, is our benchmark formula giving the nominal cost–duration view. The line segments show the linear relation between time-stretchout and cost, as found by Eq. (8). We note that although it looks good to have many IT-projects under the nominal schedules, this is deceptive. If the majority of the IT-projects is under benchmark, this is often an indication of time decompression, as a consequence of uniformly managing on budget. Of course, when time bookings are manipulated to fit available budget, we can spot this with more involved analyses, for which we refer to [53].

In Table 6 we summarized the numbers. On average, the time-stretched IT-projects take 25.8 calendar months, and the total amount of calendar months for this IT-portfolio amounts to 1289 calendar months. The benchmarked schedule times are shorter: on average 19.1 calendar months. The total amount of nominal times sums up to 953 calendar months. This saves in total 26.1% calendar months with the original IT-portfolio. Since we are in the stretch-

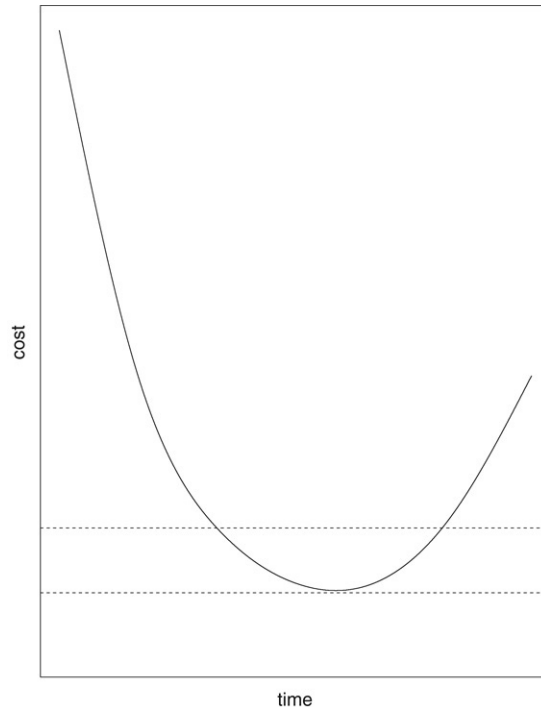


Fig. 21. Example of an empirical time-effort trade-off curve, based on internal data.

out zone, taking less time will also save costs. The actual costs are on average 7.1 million dollars per IT-project in an IT-portfolio that costs 353 million in total. The nominal costs are on average \$5.7 million, with a total cost of 283 million. Again, the results in the rows for time gain and cost saving are not calculated by subtracting the two rows above them, these numbers are the result of a standard summary statistic on the differences for all the data. Summarizing, if the projects were done in the nominal time, this implies a cost saving of 19.6% at the IT-portfolio level. We call this the *IT-portfolio time decompression risk*. As with time compression, to mitigate this risk, we have to look for individual underfunded projects, and assess their business-criticality. Sometimes you will be surprised to find fairly critical projects that are underfunded since the project owners were overshadowed by politically shrewd managers, whereas a business-criticality check would put priorities entirely different. Time decompression is a way to spot such projects, likewise the super time-critical projects can turn out to be less critical than suggested by some ad rem managers.

**Empirical time-effort trade-offs.** We have seen time compression and time decompression. We used published formulas to give the reader an idea of the effects, and they can be used if there is no internal data available to analyze. In organizations where a longer measurement tradition is in place, it is possible – and more accurate – to establish internal time-effort trade-offs, so that time compression and decompression risks can be calculated based on internal data. Such internal time-effort trade-offs sometimes look like asymmetric bath-tub curves, as depicted in Fig. 21. For such curves, there is normally no simple analytic function, as in the used formulas. By collecting time and cost for similar projects it is possible to take out a few obvious cases of time compression, normal schedules, and time decompressed ones. Based on the data of these projects, you can use cubic splines [5,25] to find an empirical time-effort trade-off curve. In Fig. 21 the solid line is a spline through a number of similar enough IT-projects, and the dashed lines mark an area that we consider to be nominal values. At the left-hand side of the curve we see a steep rise, somewhat comparable to Formula (4), indicating time compression. At the left-hand side of Fig. 21 the rise is less steep, somewhat in accord with Formula (8). Fig. 21 resembles the shapes of a figure given in [6, p. 472], where four different time-effort trade-off curves are compared.

The meaning of Fig. 21 is in fact as follows. The minimum of this curve actually represents the most appropriate manner of production: if you look at the IT-project from a pure technological production viewpoint, this is the way to

do it, at the most appropriate cost and time. Hence our term nominal cost and nominal duration. If we change time or cost beyond certain thresholds, indicated by the dotted lines, we are no longer capable of constructing the IT-system in the most appropriate manner, leading to higher costs. These higher costs can be analyzed via benchmarked formulas as given in this paper (see formulas (4) and (8)), or with internally derived relations based on proprietary data. It is possible to derive such formulas internally (and externally, obviously), since during the construction of similarly typed systems we can assume serial-piece production: products that are identical technically and therefore considered identical economically. And then we can use approximately the same relations for new investments as well, to estimate the time and cost representing the most appropriate manner of production, and if that is impossible, the consequences in cost and time for that particular investment. In this way we can balance business-criticality, time-to-market, and economics of IT-projects.

**Lessons learned.** From all this, some lessons can be learned. As with time compression, some IT-projects need to be managed on budget, so in practice you cannot avoid all added costs due to time decompression. But the viewpoint that all IT-projects have to be managed purely on budget is too extreme. When economy is on a low tide, we see more time decompression than otherwise. Maybe then the major driver is cost-leadership, instead of time-to-market. No matter the state of the economy, the eagerness to put (too much) cost pressure on IT-budgets takes place via IT-governance rules. Well-known rules are to provide one budget for every project, or to insist on many signatures above certain budget thresholds. We emphasize a few important issues relating to time decompression.

- Be careful with IT-governance rules that put uniform budget constraints on IT-projects, since this can lead to systematically underfunded IT-projects, and high costs due to time decompression. We note that the common idea that some time pressure often leads to smart alternatives is debatable, see [3] for details, so also for this reason, unnecessary rushing is not recommended.
- Overregulation can cause long waiting periods for the IT-people. This stunts not only productivity, but also induces time decompression, adding to the costs.
- Regularly scan IT-portfolios for time decompression risks, so that you can take immediate action.
- Put a uniform maximum on the amount of projects that are managed on cost only. This to prevent severe time stretch-out situations.
- Put a uniform maximum on the number of different projects IT-knowledge workers can be working on. Reason is that it takes time for knowledge workers to get in flow [14]. If they to do too many different things, no work is effectively being done [18]. This leads to time decompression, even in the case that schedules are nominal. More than 2 different major projects simultaneously is already an exposure.
- Put a uniform maximum, e.g., a percentage of the annual IT-budget, on the time decompression exposure that is deemed acceptable at the IT-portfolio level. Time decompression is in general not as damaging as time compression, but a time decompression exposure of 25% is problematic in many organizations. Better effort allocation is often reaping benefits immediately.

## 8. Managing on functionality

There is a danger that if you no longer uniformly manage on time or budget, that managing on delivering full functionality is a consequence. Optimizing towards functionality implies higher costs, longer schedules, but not always the corresponding value creation. Management on functionality is easily spotted if the following combination is present: no IT-governance rules, and a high-quality positioning in the market. Both characteristics can be found without any data collection. Lack of IT-specific governance rules and quality are readily detectable aspects of an organization.

If the above patterns are not present, it becomes more complex to spot this type of management by inspecting IT-portfolio data, since there are no “pathognomonic”<sup>1</sup> IT-portfolio views revealing this. Many different minor indications together can diagnose exclusive management on functionality. When seasonality effects are present in the data it can be an indicator of lack of following IT-governance guidelines. Another sign is a relatively low percentage of low-cost projects, and a relatively low amount of projects with short time frames. Usually, the top 5–10 IT-projects in

<sup>1</sup> Distinctively characteristic of a particular disease.

a business unit consume 60%–80% of the total annual IT-budget. The rest is spent on a large number of small and medium-sized projects. As a rule of thumb, if not 20% of the IT-projects take 80% of the IT-budget (or 80% of the (smaller) IT-projects take 20% of the IT-budget), this is a sign of uniform solutions delivery management. Another sign is that a significant number of projects is perpetual, that is, there is no delivery date planned, but continuous evolutionary cycles are used to improve the IT-function in production. Yet another indication is that such production systems comprise an abundance of options and features, that are hardly ever executed in reality.

If there is detailed information available, it becomes possible to measure requirements creep as an indicator. Requirements creep (and churn), is the compound monthly growth rate of software after the requirements have been set. For instance, if you decide to build a 1000 function point information system, and in the end it turns out to be in the range of tens of thousands of function points, the requirements have grown enormously. This growth is much higher than is common for a 1000 function point information system. Namely, the benchmarked monthly growth-rate for in-house developed information systems is 1.2%, and the maximal rate reported in the literature is 5.1% [29, p. 186]. For 1.2% and a development schedule of say 8 months for a 1000 function point system after the requirements phase, we find  $1000 \times 1.2^8 = 4300$  function points upon delivery. Although this is also high, it still deviates enormously from the example of tens of thousands of function points. So apparently, requirements are adapted continuously, often both by developers and business, inducing unrestricted and undirected growth of the software. And this in turn is an indicator for exclusively managing on functionality.

Of course, all indicators on their own provide only circumstantial evidence: there can be other reasons that clarify the found patterns satisfactorily. But if you spot the majority of these somewhat diffuse signals simultaneously, chances are big that uniform management on functionality is in place.

**Lessons learned.** Some projects need to be managed on functionality. But managing all IT-projects like that is too extreme, and consuming too many resources. Directions to solve this are partly in IT-governance rules, and partly in software development methods. A few important lessons can be learned.

- Design a set of criteria for IT-projects that must be managed on functionality. This can be regulatory projects, safety-critical projects, etc. where the functionality is a given.
- For those projects use a much more rigid requirements engineering process than usual.
- Make sure to deliver the functionality in phases, delivering the most crucial functionality first (if that is possible).
- For other projects, think of good-enough scenarios. The business must use time-boxing or money-boxing depending on time-criticality or cost-leadership strategies that fit best for these projects.
- Set a uniform threshold on managing on functionality. If more than 40% of the IT-budget is spend on a small part of the projects (say 20%), this is a serious exposure in many organizations.
- Make sure not to establish a single IT-development method, so that a one-size-fits-all approach is induced naturally.

**DSDM.** DSDM, which stands for Dynamic System Development Method [46,45], is a good example of a dynamic methodology where within the same framework, various focal points can be emphasized. We have seen organizations using this methodology, or parts thereof, to successfully mitigate uniform management on functionality. Of course there are other methodologies aimed at preventing requirements creep, but DSDM's focus on delivering what is most important first is enticing. DSDM advocates the best practice known as the MoSCoW principle. MoSCoW stands for must have, should have, could have, won't have. This, combined with time-boxing carried out by the business intends to deliver the most critical functionality first. Methods that support this are DSDM's MoSCoW principle, but also the more recent architecture based CBAM (Cost–Benefit Analysis Method) [31,32]. A time-box can of course induce uniform management on time, but this type of time-box should be used to set requirements, not to set the entire schedule. With DSDM, you can set out at the beginning the most important emphasis: time, cost, or functionality, and then those can be traded-off in accordance with the actual business needs. By comparing different business units in a large organization, we found that a unit that was using DSDM and software cost estimation tools, had a much lower time compression exposure (<10%) than a business unit that lacked these tools. In that business unit time-to-market was the overall strategy, leading to a high time compression exposure in the order of 35%. Of course utilizing DSDM and professional software cost estimation tools, also comes with a price, so we cannot easily subtract and conclude that this saves more than 25%. But since time compression is fairly expensive, and inducing high risks, a lower exposure at the cost of using some methodology will pay off: not only in investment costs for software process

and product improvements, but also in terms of operational risks. For more information on a gauge of price/return for large software process improvement programs including the implementation of DSDM we refer to [53].

## 9. Conclusions

In this paper we accumulated our experience with a number of exploratory data analyses of large IT-portfolios. We were able to reveal several IT-governance styles and their effects by specific data views. On the basis of such views it was often possible to detect unwanted side-effects. By changing the IT-governance rules (sometimes only slightly), we could establish significant improvements. These improvements were in cost savings, time savings, or risk reduction (sometimes even mitigation). We proposed 7 focal points in such an analysis. They are: overperfect data, heterogeneous data, overregulation, underregulation, and a preoccupation with unvariable management on time, on cost, or on functionality. One important conclusion is that the governance style should be flexible, in the sense that sometimes it is necessary to manage on time, or on budget, or on functionality but if one uses such management styles without assessing the need for them, counterproductive effects ensue, some of which can be revealed using the methods discussed in this paper.

In real IT-portfolios we often find combinations of the above 7 patterns. A typical example is an IT-governance structure where the small projects are relatively free of governance as long as they are under a certain threshold, say \$100000. If you are above that threshold, more governance is mandatory, and apart from more formal approvals, budget is given for 12 months. So this is a governance style that mixes management on time and on cost. First managers try to break up projects to sub threshold size, and if this is not possible they cram the work into 12 months. As a consequence, in cost–time views of such a governed IT-portfolio we will see a combination pattern of horizontal and vertical tear lines. This indicates a preoccupation with management on budget around the horizontal \$100000 cost-line, and persistent management on time indicated by a vertical 12 month-line. Moreover, due to the large amount of formal approvals, we detect patterns indicating overregulation, which is not as clear as it could, since there are also some underregulated smaller projects. But by removing the smaller ones, the effect becomes more evident. And by assessing the smaller projects, underregulation patterns become visible. These four effects together lead to a combination of time compressed and time decompressed projects, requiring a number of steps to be taken to solve the problems with the IT-governance rules.

For the sake of explanation, we separated such combinations in this paper into seven unique and recognizable patterns. An important lesson we learned is that management exclusively on time is endemic in some organizations, which does not lead to more benefits: the effects of uniform time-constraints in IT-governance rules more often cause time compression. This is easily solved by translating time constraints for non-time-critical projects into cost or functionality constraints. For instance, a project needs be finalized in 12 months, and it is not time-critical. Then it is better to provide resources for what a 12 month IT-project nominally costs in your organization. Then you reach the effect of the 12 months approximately, without the effects of time compression. If in the end, the project's scope should have been larger, then the side-effects of time compression are mitigated. On the other hand if exclusively was managed on cost, this can lead to time decompression, which is costly as well since knowledge loss is the consequence.

So, there is no best single approach. But continuously monitoring the IT-function reveals really unwanted side-effects of IT-governance, and by modest redirections we can balance the level of control to efficiency, the cost of data to its value, the amount of governance to its effectiveness, and more. Any set of straightforward guidelines for IT-governance will be an oversimplification, and it is highly unlikely that it will lead to the desired dynamic equilibrium. But one way of reaching this is to keep steering away from the seven found patterns to enable a better balance between the fundamental parameters of IT-governance (data, control, time, budget, and functionality). Mastering IT-governance is like mastering the art of juggling—if you are not keeping all the variables airborne you are not juggling anymore. But by continuously monitoring and measuring them you can gradually improve and in the end, juggling running chain saws—the reality of managing the IT-function.

Indeed, we realize that it is already a challenge to initiate IT-governance. If you manage to start such an initiative, it will not be optimal in the first year. The results in this paper can aid in assessing the current performance of the IT-governance rules, and will hint towards improvements of the rules. Also, the lessons learned will give you an idea of IT-governance rules that align best with your organization. Repeated assessments, or continuously monitoring your IT-portfolios, will spot some potentially dangerous undesired effects quickly. With (sometimes) conservative modifications of the IT-governance rules you can evolve them into a suitable governance structure. This will result in

an IT-function causing less trouble in the future than it is doing now, where in 2003 about 290 billion dollars were spent on failing IT-projects [15].

## Acknowledgements

This research was partially sponsored by the Dutch Ministry of Economic Affairs via contract SENTER-TSIT3018 *CALCE: Computer-aided Life Cycle Enabling of Software Assets*. Furthermore, this research received partial support by the Dutch *Joint Academic and Commercial Quality Research & Development (Jacquard)* program on Software Engineering Research via contract 638.004.405 *Equity: Exploring Quantifiable Information Technology Yields* and contract 638.003.611 *Symbiosis: Synergy of managing business-IT-alignment, IT-sourcing and offshoring success in society*. The author thanks the anonymous organizations who provided confidential data on their IT-function for this research. Furthermore, a number of individuals are acknowledged for lively discussions about the subject of IT-governance, and for their input and comments on a draft of this paper. In particular, we would like to thank in no particular order, Rob Peters (formerly at ING Group, The Netherlands), Menno van Doorn and Jaap Bloem (Sogeti, The Netherlands), Adri van der Wurff (Cordares, The Netherlands), William R. Taylor (Department of Housing and Urban Development, USA), Eric Onderdelinden (Capp Gemini, The Netherlands), Mark van der Pas (Vodafone, Germany), Clark Thompson (Delaware Investments, USA), David Faust (Stratasys Consulting, USA), Ton Tjink (CIBIT, The Netherlands), and Rene Brozius (Department of Justice, The Netherlands).

## References

- [1] A.J. Albrecht, Measuring application development productivity, in: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83–92.
- [2] A.J. Albrecht, J.E. Gaffney, Software function, source lines of code, and development effort prediction: A software science validation, *IEEE Transactions on Software Engineering* 9 (9) (1983) 639–648.
- [3] T.M. Amabile, C.N. Hadley, S.J. Kramer, Creativity under the gun, *Harvard Business Review* 80 (8) (2002) 52–61, 147.
- [4] N.R. Augustine, Augustine's laws and major system development programs, *Defense Systems Management Review* (1979) 50–76.
- [5] E.K. Blum, *Numerical Analysis and Computation Theory and Practice*, Addison-Wesley, 1972.
- [6] B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
- [7] S.L. Brown, K.M. Eisenhardt, *Competing on the Edge — Strategy as Structured Chaos*, Harvard Business School Press, 1998.
- [8] Budget of the United States Government, Fiscal Year 2003, 2003, Available via: <http://www.whitehouse.gov/omb/budget/fy2003/pdf/budget.pdf>.
- [9] Budget of the United States Government, Fiscal Year 2004, 2004, Available via: <http://www.whitehouse.gov/omb/budget/fy2004/pdf/budget.pdf>.
- [10] B. Cameron, R. Shevlin, J. Meringer, M. Child, *Transforming IT governance*, Technical Report, Forrester Research, Cambridge, MA, USA, November 2002.
- [11] J. Champy, *Reengineering Management — The Mandate for New Leadership*, Harper Business, New York, NY, 1995.
- [12] W.J. Conover, *Practical Nonparametric Statistics*, 3rd edition, in: *Probability and Mathematical Statistics*, John Wiley & sons, 1980.
- [13] Federal CIO Council, *A summary of first practices and lessons learned in information technology portfolio management*, Technical Report, Federal CIO Council, Best Practices Committee, 2002. Available via: [http://cio.gov/Documents/BPC\\_Portfolio\\_final.pdf](http://cio.gov/Documents/BPC_Portfolio_final.pdf).
- [14] M. Csikszentmihalyi, *Flow — The Psychology of Optimal Experience*, SOS Free Stock, 1991.
- [15] D. Dalcher, A. Genus, Introduction: Avoiding IS/IT implementation failure, *Technology Analysis and Strategic Management* 15 (4) (2003) 403–407.
- [16] T. Davenport, *Management agenda: A vote for IT federalism — The concepts that shaped our country should also apply to the IT world*, June 26, 1995, Available via: <http://www.informationweek.com/533/33uwtd.htm>.
- [17] T. DeMarco, *Controlling Software Projects — Management Measurement & Estimation*, in: Yourdon Press Computing Series, 1982.
- [18] T. DeMarco, T. Lister, *Peopleware — Productive Projects and Teams*, Dorset House, 1987.
- [19] J.B. Dreger, *Function Point Analysis*, Prentice Hall, 1989.
- [20] D. Faust, C. Verhoef, Software product line migration and deployment, *Software: Practice & Experience* 33 (2003) 933–955. Available via: <http://www.cs.vu.nl/~x/pl/pl.pdf>.
- [21] D. Garmus, D. Heron, *Function Point Analysis — Measurement Practices for Successful Software Projects*, Addison-Wesley, 2001.
- [22] United States Government, *Clinger Cohen Act of 1996 and Related Documents*, 1996, Available via: <http://www.c3i.osd.mil/org/cio/doc/CCA-Book-Final.pdf>.
- [23] META Group, *The business of IT Portfolio management: Balancing risk, innovation, and ROI*, Technical Report, META Group, Stamford, CT, USA, January 2002.
- [24] M. Hammer, J. Champy, *Reengineering the Corporation—A Manifesto for Business Revolution*, Harper Business, New York, NY, 1993.
- [25] R.W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd edition, McGraw-Hill, 1973.
- [26] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, W.A. Stahel, *Robust statistics — The approach based on influence functions*, in: *Probability and Mathematical Statistics*, John Wiley & sons, 1986.

- [27] R.V. Hogg, J.W. McKean, A.T. Graig, *Introduction to Mathematical Statistics*, 6th edition, Prentice Hall, 2005.
- [28] C. Jones, *Estimating Software Costs*, McGraw-Hill, 1998.
- [29] C. Jones, *Software Assessments, Benchmarks, and Best Practices*, in: *Information Technology Series*, Addison-Wesley, 2000.
- [30] R.S. Kaplan, D.P. Norton, *The Balanced Scorecard – Translating Strategy into Action*, Harvard Business School Press, 1996.
- [31] R. Kazman, J. Asundi, M. Klein, Quantifying the costs and benefits of architectural decisions, in: *Proceedings of the 23th International Conference on Software Engineering, ICSE-23, 2001*, pp. 297–306.
- [32] R. Kazman, J. Asundi, M. Klein, *Making architecture design decisions: An economic approach*, Technical Report CMU/SEI-2002-TR-035, Software Engineering Institute, 2002.
- [33] C.F. Kemerer, Reliability of function points measurement — a field experiment, *Communications of the ACM* 36 (2) (1993) 85–97.
- [34] C.F. Kemerer, B.S. Porter, Improving the reliability of function point measurement: An empirical study, *IEEE Transactions on Software Engineering* SE-18 (11) (1992) 1011–1024.
- [35] A. Kolmogoroff, Sulla determinazione empirica di una legge di distribuzione, *Giornale dell' Istituto Italiano degli Attuari* 4 (1933) 83–91.
- [36] T. Lister, Becoming a better estimator — an introduction to using the EQF metric, 2002, Available via: <http://www.stickyminds.com>.
- [37] B. Londeix, *Cost Estimation for Software Development*, Addison-Wesley, 1987.
- [38] F. Mosteller, J.W. Tukey, *Data Reduction and Regression*, Addison-Wesley, 1977.
- [39] United States General Accounting Office, *Information technology investment management — a framework for assessing and improving process maturity*, 2000, Available via: [http://www.gao.gov/special.pubs/10\\_1\\_23.pdf](http://www.gao.gov/special.pubs/10_1_23.pdf).
- [40] L.H. Putnam, W. Myers, Measures for Excellence — Reliable Software on Time, Within Budget, in: *Yourdon Press Computing Series*, 1992.
- [41] L.H. Putnam, D.T. Putnam, A data verification of the software fourth power trade-off law, in: *Proceedings of the International Society of Parametric Analysts — Sixth Annual Conference*, vol. III(I), 1984, pp. 443–471.
- [42] F.D. Raines, Memorandum for heads of executive departments and agencies — funding information systems investments, 1996. Available via: <http://www.whitehouse.gov/omb/memoranda/m97-02.html>.
- [43] J.R. Riemer, The truth in negotiations act — what is fair and reasonable? *Program Manager* 26 (6) (1997) 50–53.
- [44] N. Smirnof, Sur les écarts de la courbe de distribution empirique, *Matematicheskij Sbornik. (Novaya Seriya)/Recueil Mathématique* 6 (48) (1939) 3–26. In Russian with a French summary (pp. 25–6).
- [45] J. Stapleton, *DSDM — Business Focused Development*, 2nd edition, Addison-Wesley, 2003.
- [46] J. Stapleton, P. Constable, *DSDM — Dynamic System Development Method*, Addison-Wesley, 1997.
- [47] F. Thompson, *Investigative Report of Senator Fred Thompson on Federal Agency Compliance with the Clinger–Cohen Act*, 2000. Available via: [http://www.senate.gov/~gov\\_affairs/101900\\_table.htm](http://www.senate.gov/~gov_affairs/101900_table.htm).
- [48] J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.
- [49] C. Verhoef, Getting on top of IT, 2002, Available via: <http://www.cs.vu.nl/~x/top/top.pdf>.
- [50] C. Verhoef, Quantitative IT portfolio management, *Science of Computer Programming* 45 (1) (2002) 1–96. Available via: <http://www.cs.vu.nl/~x/ipm/ipm.pdf>.
- [51] C. Verhoef, Quantifying the value of IT-investments, *Science of Computer Programming* 56 (3) (2005). Available via: <http://www.cs.vu.nl/~x/val/val.pdf>.
- [52] C. Verhoef, Quantitative aspects of outsourcing deals, *Science of Computer Programming* 56 (3) (2005). Available via: <http://www.cs.vu.nl/~x/out/out.pdf>.
- [53] C. Verhoef, *Quantifying software process improvement*, 2007. Available via: <http://www.cs.vu.nl/~x/spi/spi.pdf>.
- [54] P. Weill, M. Broadbent, *Leveraging the New Infrastructure — How Market Leaders Capitalize on Information Technology*, Harvard Business School Press, 1998.
- [55] P. Weill, J.W. Ross, *IT Governance — How Top Performers Manage IT Decision Rights for Superior Results*, Harvard Business School Press, 2004.