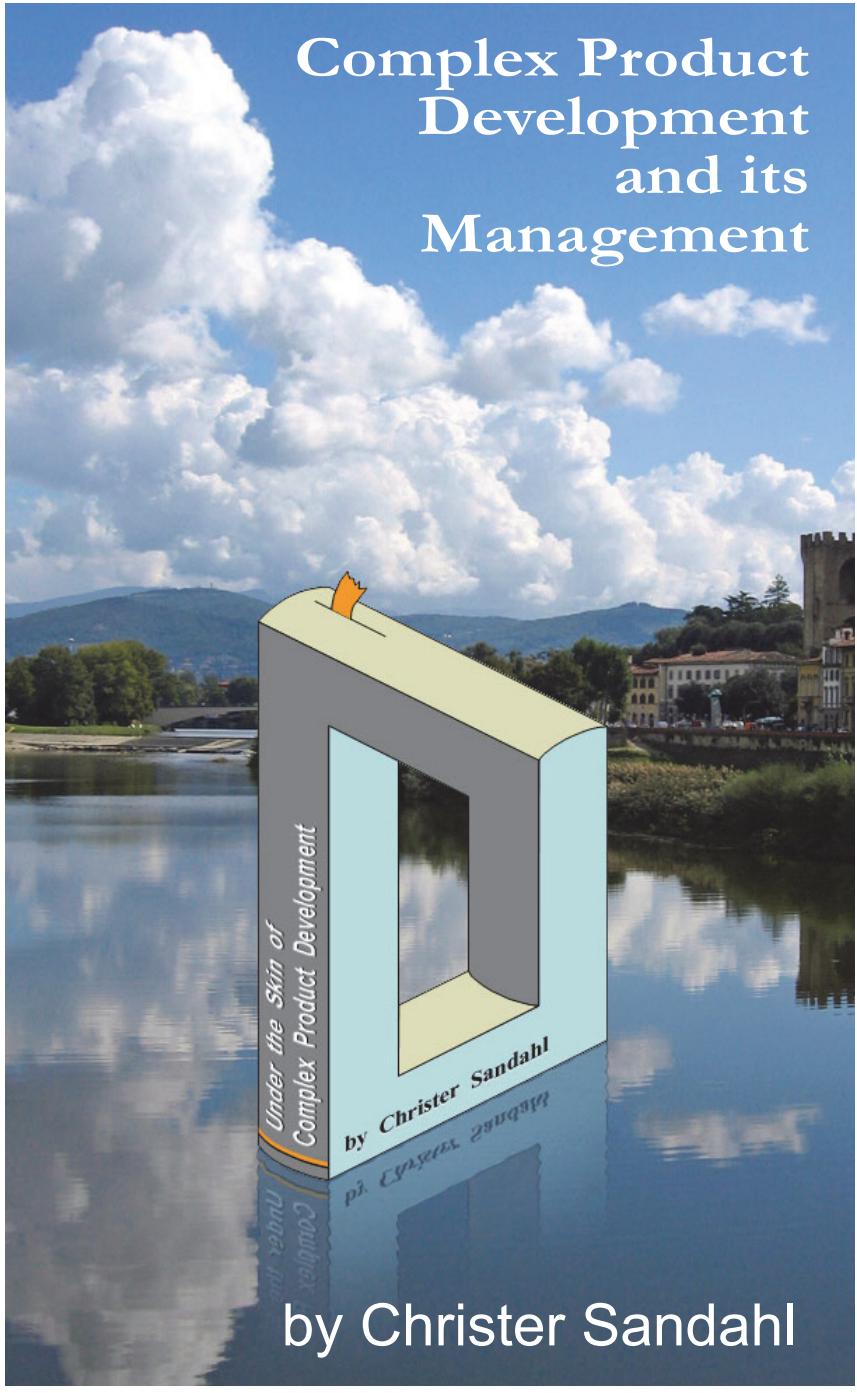

Complex Product Development and its Management
Under the Skin of Complex Product Development
by Christer Sandahl
by Christer Sandahl

This Book

Dedicated to my dear family,
for taking proper care of me,
when I fade into book coma.

To be printed and marketed by AuthorHouse

Revision of 23 May 2011

Overview Table of Content

Detailed Table of Content

Overall CHAPTER 1

Warning Bells

**"There is a lot of noise in the jungle,
you must only be aware of the dangerous"**

Why nailing with a cudgel ?

1.1 What has gone wrong ?

In development organisations, it is not seldom seen that people work backwards like nailing with a cudgel (primitive club). Why ? You never get a carpenter to your house with a cudgel to nail with, do you ?

Why are skilled people behaving like nailing with a cudgel ?

Product development is nowadays in many respects an established and ordinary business. For example, house and bridge development are several thousands years of age. Other fields of product development are much younger, for example dealing with software begun in the decade of 1960.

However, in all development fields, there are still products which fail to satisfy end users. In some newer fields like software, trouble looks to be the standard, but in other fields it is a bit better. And note, there is no conspiration behind this, no supplier like to disappoint an end user. So what is the problem ?

Uncontrolled complexity risk to emerge, when a business are getting scaled up.

If digging deeper in the development business in order to find the root cause to why products fail to satisfy end users, the most common reason seems to be that up-scaled development introduce multiplied levels of complexity, which in turn cause capability and competence problems when organising development of these products.

Two thousand years ago, the Pantheon building in Rome was an ultimate complex house construction on the very front of development knowledge at that time, but should nowadays be a rather modest target for a mid sized construction company. A complex building of today is a several hundred meter high sky scrape, which need high tech solutions and sophisticated calculations for strength of construction materials, and several hundred of workers within many disciplines must be well organized to make the building being raised. Complexity seem to have increased over time, which implies that it is certainly possible to coop with larger and larger complexity.

To analyse complexity a bit further, imagine the two diametrically opposed ends of complexity, the "ordinary low end", and the "utmost high end".

1.1.1 Success from the low complexity world

Houses have been constructed with success for very long. A normally handy person (with some drive) can, for example, extend his private house with some new space. He may have to contact experts to sort out problems beyond his competency and hire specialists to help him build, but in the whole, this is not more complex than he can lead the construction work and also take part in the craftsmanship. The extended space will be of desired cost and quality, and will function as planned. What possibly fails can afterwards most often be repaired at modest costs.

This scene is true for many ordinary scaled business in our modern society. (Let be that the house business recently face new complexity, when being target for a mas-

sive energy saving requirements or when competition press prices for house building below what is reasonable for persistent quality.)

1.1.2 Success from the high complexity world

There might be air plane crashes and medicines with severe side effects, but to travel by air or follow a doctor's subscription is generally very safe. In these cases, the high complexity of developing aircraft or medicine are undoubtedly handled with success. Obviously, in these fields, the scaling up of complexity has worked out very well, even if not totally clean from disasters.

1.1.3 Transition from low complexity to high

However, not all business and companies has manage to make this transition in a proper way. Computers often hang up and spoils large amount of work, consumer electronics fall into pieces and must be expensively repaired, kitchen equipment barely keep together until warranties are expired, etc.

Some companies retain success and others fail when getting into complexity.

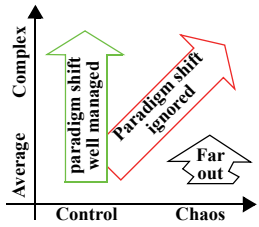
In these unsuccessful cases there are of cause a lot of extenuating circumstances, like everything must be developed in a rush because the market change quickly, testing is not given enough time and is forwarded to the end users, money is spent on commercials rather than development etc. And in the software discipline, typically one after another line of code is added, and the scale-up come very creeping and invisible, and all of a sudden has ruined the structure of the system.

Many unsuccessful companies might argue, that it is not really their problem if they fails to deliver satisfying complex products. Who hasn't heard "the customer simply gets what they pay for". But most often an analysis would have showed, that poor products costs more than they save for both producer and customer. These products are often in the poor end of the quality scale, and in fact it would have been more profitable to develop them better from the beginning (at least when consider the full life cycle of the product).

1.1.4 Is there any trick with scaling up ?

Scaling up is like 1 + 1 = 3. At some point more than the size has changed, a paradigm shift has occurred.

It is recognised for long that when things dramatically change in scale, the thing is not just anymore the same but with another size. It get more like 1 + 1 = 3. Sometimes this is referred as "at some point when increasing quantity, there is a change in quality" or in our case "at some point along the scaling-up there must be a change in approach, sometimes referred as a "paradigm shift" is approaching.



If the paradigm has shifted, the old methodology and approach must be replaced, and a radical new way of thinking must be applied. For example, house stairways must be replaced by elevators when houses get higher, growing software must be partitioned into smaller pieces separated with clear interfaces, slide rules get replaced with digital calculators, key hole surgery being far more efficient than open big wounds etc. The world is full of (smaller and bigger) paradigm shifts.

And back to the initial question, why nailing with a cudgel. Developers might nail with a cudgel because their approach hasn't been scaled up to the actual complexity facing them. A paradigm shifts has occurred but was ignored. The cudgel was successful against primitive enemy tribes, but has got very inappropriate for nailing.

1.1.5 Short about complexity

With complexity means, when a set of system parts have relations to each other, in a way that forms a total system that is hard to understand and predict.

Let's look at the solar system. When copernicus placed the sun in the middle, Isaac Newton was able to describe the motion of all planets with his "laws of motion". This is rather ordinary mathematics, referred as the "n-body problem", which can be analytically handled.

But complexity reappear at this stage. It was rather simple to solve the equation for n = 2, e.g. two planets like the sun and the earth being alone in the solar system. It took several hundreds of years to solve it for n = 3, e.g. three planets like the sun, the earth and the moon being the only planets. For n greater than 3 it is still not completely analytically solved, but the challenge has lead to a lot of chaos research.

This is in short what happens when scaling-up. Very soon the system parts form a total system that possibly might be described, but gets hard to understand and predict. Often such systems are referred as systems in chaos. The system itself doesn't know that it is in chaos, of course it is our understanding that is not good enough.

The general way to treat complexity is to make research, in order to enough understand the complexity. If still too complex to be handled, some mitigation can be tried. One way might be to limit the degrees of freedom and accept a lesser accuracy of understanding, for example by approximations (the moon has no influence on the sun), or to freeze some relations (the sun is assumed fixed in the centre).

Have you ever reflected over why houses preferably have right angles between most building elements. Do you get the point? Simply because this lowers the complexity and makes a house easier to understand, predict and build. The Beijing Bird's Nest, not having two similar angles anywhere, has such large complexity that it had been impossible to handle in the slide rulers era, but could be mastered by designing with powerful computers.

1.2 When should you hear the warning bell ?

A company having problem with mastering its complexity, shows many symptoms from this if anybody care to watch. And most often these symptoms has continued for a long time. Scaling up is a slowly creeping effect, and problems may be small and silent in the beginning, but hitting hard after a while if not cured.

And there are reasons to watch out. The first company to identify an approaching paradigm shift and succeeding to overcome the challenge, is coming out very strong and competitive. Instead of being gradually slowed down by growing legacy complexity, they can capitalize on their new way of mastering it, and in short time get ahead of their competitors.

The most obvious examples are paradigm shifts in warfare, which could even change the balance of power between whole countries. A lot of effort is spent on intelligence and reconnaissance in order to watch the enemies effectiveness.

Disorder during development, result in that this disorder is also built into developed products.

One may perhaps argue that it is not a big problem if a development organisation is internally messy and in disorder, because this will not trouble their customers. But this is often wrong, because this disorder will as well be built into their products, which will get the same lack of structure and quality, and will thereby finally hit their customers.

To identify problems with mastering complexity in development organisations, watch out for the following 13 warning-bells:

1. "EXAMPLE: Unrealistic campaigns continually restarted" (page 28).
2. "EXAMPLE: Management not being accountable" (page 28)
3. "EXAMPLE: Root cause analysis being suppressed" (page 29)
4. "EXAMPLE: New wrapping with same content" (page 30)
5. "EXAMPLE: Ego people being change agents" (page 30)
6. "EXAMPLE: Sub optimization within local organizations" (page 30)
7. "EXAMPLE: Scapegoats of a blame game" (page 31)
8. "EXAMPLE: Products driven by engineers" (page 31)
9. "EXAMPLE: Product structure being degenerated" (page 32)
10. "EXAMPLE: Impossible principles applied" (page 32)

11. "EXAMPLE: Artificial complexity being pushed" (page 33)
12. "EXAMPLE: Intrinsic complexity being ignored" (page 33)
13. "EXAMPLE: Devils in the details being ignored" (page 34)
14. /* EXAMPLE: Continually decision reset */

1.2.1 EXAMPLE: Unrealistic campaigns continually restarted

In this case, improvement campaigns are being broadcasted from management and the message from them might look like Figure 1-1 below.

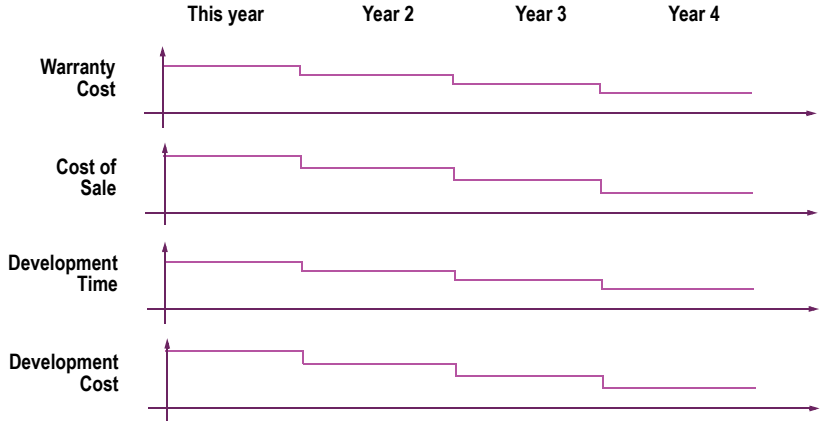


FIGURE 1-1 Improvement campaign goals

Why are many organizations running improvement projects, one after another, without sustainable results ?

Probably you have seen such programs sometimes passing by. Most of the energy is put on cheering and making noise, and less on analysing, understanding and implementation of changes. Very often these campaigns are very intense in the beginning but are fading out as time pass. You may also have ended up with the feeling, that not much have been achieved at the end of the campaign (even if graphs are circulated proving the contrary). And for that so little is delivered, seldom anybody are found responsible.

When a new manager enter the organisation, the campaign is restarted again to show drive and energy. But of course with different names, concepts and symbols, but with the same type of unrealistic campaign.

1.2.2 EXAMPLE: Management not being accountable

Managers are the most important group of employees, when to establishing good and efficient working models. Over and over again, it has been found that making improvements and establishing working models is more or less impossible without

active support from managers. Keep a watch on the following criteria, which often prevent from setting a sound company culture:

- Managers not specially interested in how his inferiors are working, and claiming that they are expected to sort out that themselves.
- Managers thinking it is more important keeping pease in his organisation and reporting to superior management that everything is working fine, instead of risking noise from solving severe problems.
- Managers not ever acting pro actively to problems. Their habit is always to wait until the failure is a matter of fact, before they take action.
- Managers pretending to be interested in work models, but just desire to silence their conscience. They might engage persons to document and improve, with the hidden agenda to archive the results in binders in order to be forgotten.
- Managers acting grandiose, and claim that they already have everything under control. For example, imagine that this book were shown to them. They would answer that here is nothing new they didn't already knew. It's a great book, but we already work according to it.

It is seldom possible to influence on which managers there are in a company, but yet it sets the level for improvement success. It is waste of energy to try to improve companies with inappropriate management.

"The management culture" is the most influential on how complexity is handled and improved.

1.2.3 EXAMPLE: Root cause analysis being suppressed

Sometimes a frenetic "improvement wave" can be spread over an organisation. One after another wants to be the best on improving the way of work. It may even happen that upper managers tries to beat each other with efficiency programs and rewarding improvement proposals.

Their eagerness admit no time to structure the improvement work, and improvements get started on every imaginable spot of the organisation. Everything are object for reparation and improvement. Current work models are declared insufficient and are discarded (and forgotten), like throwing the baby out with the bath water, in favour for new bright improvements to come.

An objective and honest root cause analysis might be embarrassing and inconvenient, but is most essential for serious improvement management

Unfortunately, sufficient analysis are not being made, pointing out the poor parts being most urgent targets for improvements, and what parts in fact is "good enough", at least for a while. Neither it is planned in what order the poor parts need to be fixed. Often organisation are very well aware of existing real problems and bottle necks, but courage are lacking to present these facts and get these problems visible. It is much more convenient to sweep the most ugly problems under the carpet and report about more harmless shortages. They are not so embarrassing and they are much easier to fix.

1.2.4 EXAMPLE: New wrapping with same content

Even if development is complex, there is certainly a limited amount of fundamental ways how to organize product development. This fact is troublesome for methodology consultants, salvation authors (myself being an exception:-) and other confidence trick makers. But like in the fashion business, this is solved by change the wrapping and reintroduce old things as being the latest inventions, that are urgently needed by everybody.

Trousers can not be designed in so many different ways. But the fashion business succeeds, over and over again, to surprise us with yet new trousers.

People too young in the development business was maybe not there the last time these things were in fashion. And managers may not have time enough to penetrate and disclose all "package" tricks. Like fashion consumers, people in general fear the risk to be regarded uninformed and old fashioned.

When these arguments arise from consultant sales persons or by improvement proposals from inferiors, it easily happens that it is decided to acquire similar things (but differently named and described) that might already be acquired and even might be in place.

1.2.5 EXAMPLE: Ego people being change agents

Many persons love to start up new things, to be inventive persons, to get a lot of attention and look busy, and to be hang-arounds to influential managers. They use their charisma to sell in solutions to anyone in need for anything.

But after a while, when it gets harder to deliver and show promised results, these persons pop up somewhere else in the organisation with other newly started improvements. If follow-up from management is poor, these persons are never made responsible for what they promised but not delivered, and can proceed to jump around.

Charisma people are very useful, but never appoint them in expert positions.

Needless to say, this example is a disaster for improvement activities and the organisation moral. Much more of this will be discussed in the chapter "Meta CHAPTER 18 Improvement & Assessment", at page 251.

1.2.6 EXAMPLE: Sub optimization within local organizations

When organizations get that large, that everybody don't meet each other face to face any more, there will appear more and more individuals not being in contact with each other. This is nothing wrong in itself, in a big company everybody can not work together with everybody else.

To still bring employees to share the company culture and value chain, there is now an emerging need to work with formal documentation and improvements on many abstraction levels. As a consequence of this, it can often be seen that groups internally works very efficient and structured. But if looking on how these groups contribute together for the company result, it might be very inefficient, or the groups might even destroy each others work results.

Even if a group performs very well, it is not thereby given that it brings any substantial value to the organisation

The higher up in an organisation the bigger are the effects of problems and rewards from efficient solutions. But in many companies this fact is not recognized, and even the revers may occur. On low level there might be a dedicated improvement work ongoing, but the higher up in the organisation, the more uninteresting managers get for improvements and an efficient way of working.

1.2.7 EXAMPLE: Scapegoats of a blame game

If crisis of any kind hits a development company, it is quite natural that everybody tries to protect themselves, and managers tries to protect their organisations. One way to protect yourself, if you can make believe the cause of the crisis are outside of your own domains, is to declare that you and your organisation was only an innocent victims. Often a company executive group get pressure from the boards and owners, and must present drive and improvement programs to overcome the crisis.

Altogether, there is a high desire to point out scapegoats in order to try to hide own responsibility. And in the same pattern as common mobbing mechanisms, the weakest parts of the company organisations is less dangerous to attack.

The value chain and people working with the value chain, are often targets when searching for scapegoats of a crisis.

Not seldom seen, is that executives point out the engineering value chain to be the cause for the crisis. There are examples, that even if the crisis obviously was caused by outdated product portfolios delivered by incompetent market organisations, it ends up with the management declares that developers are working in the wrong way.

A good way to eliminate such mismanagement, is to demand root cause analysis. But the opposite is more common, it is easier and more controllable to point out a weak scapegoat, than to launch a root cause analysis which may find skeleton in anybody's cupboard.

1.2.8 EXAMPLE: Products driven by engineers

Many product markets (even if being very technical products) are not different from the fashion market. It is the price tag and the appearance of the product that is most important, and the rest of the product characteristics must only reach above a normal "hygiene standard".

Engineer driven development is often carefully watched. But to ensure it will not happen is hard.

If a company is governed by mostly technical people, the reverse might occur. Then the technical systems within the cabinet get most important and heavily improved but the outlook is kept as boring as forever. User perceived quality may slip because feedback from the market is ignored. A high return rate and warranty cost is often the result.

A company may have a lot of market driven people and managers, but despite this might still be engineering driven, because a strong channel might be missing to convey market driven requirements into the centre of the engineering departments, for more details about this see the chapter "Detail CHAPTER 6 Requirements", at page 109.

1.2.9 EXAMPLE: Product structure being degenerated

Product structure and architecture are often the most misunderstood of all development concepts. It is very strange, because for the house construction business the architect is both important and well understood. In hardware development it is fairly well understood, because the components are tangible and can be likened to rooms in a house.

A software structure might be as degenerated as an allotment-garden cottage. A lot of rooms, nooks and corners, but nowhere to live.

But when coming to software, it might be totally impossible to make analogies to rooms, flats, floors etc. A software construction that has been uncontrolled extended for long time, may have an architecture very similar to a allotment-garden cottage. Small rooms has been added to the house body every summer but the body itself has never been reconstructed. This ends up in a cottage with a large amount of rooms, nooks and corners, but nowhere any continuous space for living. More about understanding architecture will be discussed in the chapter "Detail CHAPTER 7 Architectures", at page 151.

1.2.10 EXAMPLE: Impossible principles applied

To build a company on strong principles are generally a good sign, for example like the successful companies Toyota and Ikea. But be aware of, that if wrong or impossible principles are selected, then the damaged will be equally strong as the success would have been.

What a competitive edge it would be for a company which succeed to "suspend gravitation". But what are the chance ?

A striking example on implying impossible principles is the construction of the Swedish warship Wasa. Sweden was in war with Poland, and needed better fire power in their navy. Thus, there was a heavy force from the Swedish king to equip the ships with as many cannons as possible. Consequently the ship Wasa had too many cannons compared to its size and ballast, but despite this the king ordered the launch of the ship. After ten minutes in fine weather, the ship capsized and sunk in 1628.

Another example is a company having big problems to handle their system, which grew bigger and bigger for each day, resulting in uncontrollable organisations. The system engineering department was not in control and was even seen to be unneeded. However, a big improvement project was started and the upper management announced "the principle of de-coupling", meaning that different parts of the system should be developed separately from each other and handed over from smaller organisations for integration and assembly.

The "only" annoying problem was that the architecture of the system was far from being decoupled, and no sub-system interfaces were described, visualized or managed. Few of the persons in the management understood, that if the system wasn't partitioned in parts, it was useless to partition the organisation. A lot of time, energy and money was thrown away when to divide the company into decoupled organisations, but without decouple the technical system parts from each other. More of this will be addressed in the chapter "Detail CHAPTER 7 Architectures", at page 151.

1.2.11 EXAMPLE: Artificial complexity being pushed

One of the best examples on this, even if far from the product development world, is the old solar system model with the earth in the centre. The solar system model showed the movement of all planets assuming that they were connected on spheres that were rotating at different speed. But some observations were really difficult to explain with this model. For example, it was not so seldom seen that a planet moving over the sky, suddenly reversed its direction and retrograded for some time, before reversing once again and proceeding in the original direction. The more irregularities of planet movements that was observed, the more epicycle spheres was put into the solar system model.

Complexity in this model was actually added by humans, and consequently most of it disappeared when Copernicus released his model with the sun in centre (by the way, a substantial paradigm shift).

Things should be made as simple as possible - but not simpler.
- Albert Einstein

In product development it is easy to end up with too many "spheres" and everything gets more and more impossible to understand. Every model builder is not as brilliant as Copernicus, but the warning bell should be ringing, when working models gradually gets too complex to understand. It might be the consequence of a model that has evolved beyond its initial simple context, and must be replaced by a more well-reasoned one.

1.2.12 EXAMPLE: Intrinsic complexity being ignored

This is the opposite to the preceding example and might ring the warning bell as well. Product development can be really complex, du to very complex products that are being developed in shorter time that is possible (so to say).

One good example of this is when a system to be developed has varying lead times in different architectural parts. For example, turn around time for making a new

software release might take a couple of days, while making a new ASIC may take some years to prepare.

Again, things should be made as simple as possible - but not simpler.
- Albert Einstein

If you neglect the different lead times for the software and the ASIC, you never succeed to get requirement management started in the right time for different architectural parts. That implies that different parts will not be ready for integration at the same time, which implies that some parts must wait for other parts to get ready. Or that some parts must be used despite they are not yet finalized.

The point is, that if the complexity of different lead time is not taken care of in the work model, the requirement management will never succeed.

1.2.13 EXAMPLE: Devils in the details being ignored

Of course it is important that top management are supporting improvement work, but it might be dangerous to drive and organize everything strictly top-down. Some nasty details can be as hard to overcome as a law of nature. An improvement without experts checking in advance for devil's details may come out to nothing.

A good example is when totally development lead time must be shorten. That means to do the same things like before, but on shorter time, which in turn often results in doing more and more things in parallel. To understand that the synchronisation between parallel work will still be possible in practice, a critical path analysis must be performed on the new parallelism to apply.

The saying "little strokes fell great oaks" might be terrible decisive in complex product development.

At some point in trying shortening the lead time, it is not possible to further increase the degree of parallelism, because some details of the devil sets the limit. For example, it take some minimal time to get an ASIC (silicon chip) produced after you have submitted all ASIC drawings and this time isn't realistic to shorten beyond a certain duration. For analyses of critical path it is essential to involve experts of details, knowing the critical path and where unconditional limitations sets in for each activity in the path.

1.3 Your way out of this

Above chapters describe symptoms when complex development are deviating in wrong ways. It is very important to diagnose such symptoms in an early stage, when it is still easy and cheap to cure. But how to carry out this treatment? This book give one clear answer to it - the development organisation must better master their complexity they face.

This book do help you out. (In this book, important statements are shown like this)

To learn about complexity is hard and there is certainly no short cuts. This book intends to give you a palate of aspects on complexity, that can help you to discuss, understand and come up with improvement proposals that really help.

(If you find this book being impolite when claiming that you need help, but in fact you don't, please instead review this book and send back your comments to the author).

This is not just an academic book that only enumerate all plausible facts of complex development, but it is instead a book covering real experience and long time learning. It contain a almost endless amount of examples from various relevant development worlds. But this book doesn't stop even there, it also put all these examples into context of the whole lot, in order to capture and illustrate the complexity behind. And that is in fact what you are in big need of, but seldom get from other books. This book will take you underneath the skin of complex development, and prevent you to fall into any of the pit-falls presented above. Good luck!

Glossary of terminology

Term	In this book	Out there
Architecture	How the structure of an artefact is assembled by its parts, including immaterial artefacts like software.	About the same
Artefact	Something developed or produced by humans, including immaterial artefacts	About the same
Black-box (BB)	Demarcation which not disclose anything of its internal content, but only showing properties of its surface, and behaviour via its interface. Properties and behaviour are described by requirements.	About the same
Component	A sub-system being explicitly developed or extracted from a finished system, in the way that it can be easily understood and reused without significant rework. Electronic components are typical, even if they often have small granularity.	About the same
Decompose	To split up an architecture black-box into its white-box, containing design elements (some of which can be underlying black-boxes).	About the same
Description	Information about an artefact. Always be careful when referred to an artefact, is it the artefact itself or is it its description being referred.	About the same
Design (noun)	See design element.	
Design (verb)	Transform black-box requirements, to design elements in resulting white-box.	Rather similar
Design element	Anything showing up when decomposing a black-box, e.g. underlying black-boxes, electronics, software commands etc. Design elements can be developed and finished without a new layer of formal requirements.	Many definitions
Development	Translate an product idea (satisfying a market demand), into descriptions and prototypes, making a production possible. Be aware, to not mix production and development together.	About the same
Formal	Comply to concepts that are shared by colleagues.	More vague
Granularity	Size of components (compared to size of assembled system).	About the same
Interface	Interactions with a system (or sub-system) are through its interfaces. If the system (or sub-system) is judged a black-box, the interface requirements must be described and managed.	About the same
Life cycle status	Regarding a named artefact to only change maturity states, during its life from birth to death. The life cycle status reflecting the maturity might be stated after the name.	About the same
Line (management)	The static and most often hierarchal structure of managers to lead a company.	About the same
Managed information	Information is said to be managed, if it is uniquely identifiable, documented, obeyed and updated to always reflect the reality.	About the same
Module	A sub-system without clear interfaces to its system. It can have a explicit name, but it is not easy to understand how it interacts with the system or how it is demarcated from the system.	Unclear distinction between module and component.
Object	A self-contained and autonomy artefact. Close in its meaning to a small black-box.	About the same
Product	A product is an artefact, created by somebody, from raw materials, to finished goods, for a market, to satisfy a need.	About the same

Term	In this book	Out there
Product life cycle (PLC)	Time period, from a product idea is invoking a development, until the product is out-phased on the market, including eventual warranties.	About the same
Production	To realise and multiply construction in volumes, based on developed construction documentation and prototypes. Be aware, to not mix production and development together.	About the same
Project	A temporary structure in a company, to accomplish an assigned objective, and restricted by limited conditions (time, cost etc.).	About the same
Project leader	The manager of a project, reporting to the project sponsor.	About the same
Project office	A line manager with a pool of project leaders	About the same
Project sponsor	A line manager ordering and controlling a project.	About the same
Property	Observable characteristics on the surface of a system (or sub-system). If the system (or sub-system) is judged a black-box, the property requirements must be described and managed.	About the same
Refine	Narrow in black-box requirements to match contained underlying black-boxes.	Can mean whatever
Requirements	The behaviour and properties of an artefact.	About the same
Specify	Users capturing wishes from a system.	Much more vague
Stakeholder	Anybody that have interest in what are specified by requirements. A stakeholder might be an end user of a system to be developed, an orderer paying for the development, a verifier testing a developed system against its requirements, etc.	About the same
Sub-system	Partition or part of the system to be developed. If a sub-system is judged complex, it should be considered as a black-box with requirements to describe it.	Much more vague
System	The precise target demarcation for what is to be developed, no less, no more.	Much more vague
Value chain	The factual way that work are performed in a company (to add value to it), regardless if it is understood or described.	More vague
White-box (WB)	Demarcation of an amount of visible design elements. When opening a black-box it gets a white-box containing embedded black-boxes and other design elements. Be aware that a white-box always has requirements, that has been updated to still satisfy the chosen white-box design.	About the same

Term	In this book	Out there

Index

W

Christer Sandahl has for over 40 years been living in the engineering world. Beginning with photo and chemistry in his teenage, over to electrical engineering in university, and into computer design as professional.

In his early working life Christer has all by himself several times constructed large computer systems, both hardware and software.

When computers got large and complex, he has for long periods managed software groups in successful local companies as well as in large world wide combines, such as Sony Ericsson, and Axis Communications.

Christer has grown up in the "Gnosjö area" of Sweden, the origin of entrepreneurship. His family has a large transportation company, he and his brother has quality wine production in Hungary, and Christer has of cause taken on this way of life in his engineering profession.

- Why is malfunctions common in computer products?
- Why is it frustrating to operate our everyday products?
- Why do most development projects get out of hand?

It become more and more evident that behind those problems, is mainly the ever growing complexity, which is not enough understood and mastered.

If deciding to master complexity, there are no single easy cures. You need generalists, you need specialists, but most essential, you need people that are both, to establish efficient bridging between all different areas of development.

This book describes all those concrete elements of complex product development, and ties them together to a uniform conception. With you in mind, all descriptions comes with plenty of real examples and illustrations, from prestigious board rooms down to the plain floor.

Don't ever tell me you didn't get a chance.