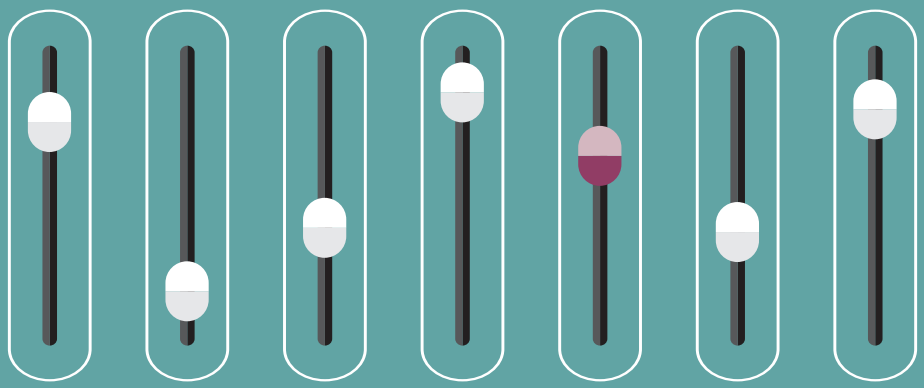
Insight driven design : balancing production speed and development learning in smart industries

INSIGHT DRIVEN DESIGN

balancing production speed and development learning in smart industries



Alex Alblas
Fränk Zwaans
Sander Schepens

INSIGHT DRIVEN DESIGN

balancing production speed and development learning in smart industries

INSIGHT DRIVEN DESIGN

balancing production speed and development learning in smart industries

Alex Alblas
Fränk Zwaans
Sander Schepens

Colophon

Insight driven design
balancing production speed and development learning in smart industries

The authors gratefully acknowledge the support of:

- Prof. dr. Fred Langerak

Special thanks to contributors:

- Rens Beelen
- Sarah Gelper
- Tobias van der Kaaden
- Niki de Kadt
- Jeroen van Olffen
- Sanir Pašalić
- Nico Spoelstra
- Henk Stadhouders
- Martijn Voogd

Expert Insight: Fränk Zwaans
Text: Tekstbureau Skribo, Derk Ederveen
Design: Op de millimeter, Mechteld Ballintijn
Production: Proud Press

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-4201-7

Second print, January 2017

Contents

PREFACE

The ability to learn is a shared characteristic of all living beings. The speed at which an organization can learn makes all the difference. That is the speed to accumulate knowledge and experience, to classify and analyze, and to share within communities with the goal to initiate any kind of action or reaction.

In high tech industries, the real competition is about continuously delivering innovations on time. It is therefore critical to understand the factors that can influence this speed, and act on them.

This book is based on the hands-on experience of introducing many new products in ASML. It provides a renewed vision of our collective learning mechanisms, highlighting unexpected parameters with high impact in many areas, and then offering a new field for managers to improve our performance.

Frederic Schneider-Maunoury

Executive Vice President and
Chief Operations Officer, ASML

Management summary

This book is the culmination of two years of research, on ten years of company data, including development, manufacturing, design iteration and design debugging data. The resulting conclusions are backed by dozens of interviews with company experts.

Unravel the B time mistery

In smart industries, managers are struggling to keep cycle time in check, and non-planned time (B time) in particular. B time is currently not only the most uncertain, but also the most prominent component of production time. From an academic point of view, the factors determining B time length were unclear up to now. The research results presented in this book clarify a considerable part of that B time mistery.

The largest percentage of B time is due to issues during assembling and testing of a new product type: time spent on solving Design Iterations and Design Debugs. A Design Iteration introduces functional characteristics to the product, and each iterative step takes the product closer to perfection. A Design Debug emerges when something does not go as planned during assembly or testing of a particular product, which may lead to production being temporarily halted.

Focus on learning curve management, not merely on output

Managers are required to recognize the importance of learning curve management. So ask yourself the question: why should I focus on learning curve management over production output? There are many reasons provided in this book. But the most simple answer is that learning in the transient phase of new product development is mainly driven by experience in design iteration and debugging on the first series of products, and not by experience in production output. Optimal cycle times can only be achieved by proficiency in managing design iterations and debugs. If you are convinced of its importance, you can get in control of cycle time, and manage the progress in B time.

Seven management principles, to balance learning and speed

We not just analyze B time, we also provide optimization tools. Managers can adjust seven management principles to balance learning and speed. Chapter 3 shows that learning is inevitable: without learning, cycle time goes through the roof. With a learning rate of, say, 20%, after some time a factory can quadruple production with the same resources. But the balance between learning and speed strongly depends on the phase. For example, learning is more important in the proto phase, while speed should be paramount in the volume phase.

Nine process measures, four outcome measures

The management principles operate on nine process measures we distinguish, either boosting or supressing them. The process measures themselves can have a positive or negative impact on learning, on cycle time, or both. This is visualized in our quadrant model in paragraph 3.3.2. The subsequent chapters use that quadrant model to score all process measures against four outcome measures. Chapter 4 explores process measures related to Design Iterations and delta's, and chapter 5 explores process measures related to Design Debugs and downtime. At the end of both chapters we list tangible advice that managers in smart industries can use to steer the balance between learning and cycle time, and thus optimize production.

One framework

To provide a better overview, in Illustration 3 of paragraph 2.6 we compare the above framework with a sound system. The mixing console (the management principles) is at the heart. It filters the properties of the input channels (process measures), to impact the output channels (outcome measures). The result is a combination of mixing console settings, optimized for a particular type of music. The summary table, at the end of this book, presents an overview of optimal management principle settings per phase.

Why this book?

This book presents research breakthroughs in managing innovative development and production processes. Innovative, complex, iterative processes are typical of the growing branch of smart industries. Their processes are incomparable with those of the 'traditional' industries. That is why smart industries require special attention.

This chapter shows what's so special about smart industries, and gives a preview of what their managers can do differently to optimize success. We will show why it is important to balance production speed and development learning, and why it is necessary to analyze cycle time types. Finally, we introduce our sound system framework, which puts managers in control of B time, the most important cycle time type of all.

2.1 SMART INDUSTRIES ARE SPECIAL

But first: what exactly is a smart industry? It is about deploying Information and Communication Technology (ICT) in manufacturing in the broadest sense: big data, the internet of everything, 3D printing, robotization, artificial intelligence, cloud computing, you name it. A Dutch project team defined it as follows:

> *Smart industries have a high degree of flexibility in production, in terms of product needs (specifications, quality, design), volume (what is needed), timing (when it is needed), resource efficiency and cost (what is required), being able to (fine)tune to customer needs and make use of the entire supply chain for value creation. It is enabled by a network-centric approach, making use of the value of information, driven by ICT and the latest available proven manufacturing techniques.*[1]

Both the National Research and Development Strategy in the United States, and the Horizon 2020 program of the European Union, highlight smart industries such as semiconductor manufacturing and biotechnology, as crucial for economic growth and industrial development. In the Netherlands, institutions like TNO, VNO-NCW and the Ministry of Economic Affairs have taken initiatives to advance smart industries.

Because of our unique characteristics...

So why are smart industries special? Because they share some unique characteristics. First of all, an international outlook. Secondly, they deploy and create state-of-the-art technology, by using conceptual innovation and out-of-the-box thinking.

Another important aspect is smooth and open collaboration, not just internally, between different disciplines, but also externally with proactive value chains: suppliers with specialist knowledge and technology, such as ASML, Fokker, ten Cate or VDL ETG. In this way they enable competitive industry sectors like high-tech, chemicals and logistics.

And last but not least, smart industries focus on both product and customer. They develop integrative products that meet customer needs, and combine them with services, like training and maintenance. The focus on product quality is extreme, for example by using sensors to monitor production, or self-learning to improve production automatically. Also, product customization never stops: products are highly diversified and customer participation is actively encouraged, for example by using apps for customer feedback.

... we face distinguishing challenges...

The above characteristics can be summarized into a unique combination of five challenges:

1. high variability
2. low production volumes
3. knowledge-intensive products
4. capital-intensive products
5. production is driven by an ongoing process of Design Iterations and Design Debugs

Those are the main challenges that smart industries face, to effectively design and manufacture their products. These challenges are important, because they show that smart industries fundamentally oppose the traditional conveyor belt manufacturing industry. Henry Ford did not offer options: *"You can have any colour as long as it's black."*. He produced exactly the same product in huge numbers at the lowest possible price, requiring no knowledge or changes because the product had already been perfected.

But the world has changed; people demand personalization. In smart industries, every product may be unique due to tons of options, and new products are still introduced at the speed of the old conveyor belt. This trend is irreversible, and it requires non-traditional manufacturing and management principles to produce unique, innovative, expensive and customized products.

... and have to balance speed and learning

The five challenges of smart industries listed above require us to rethink strategy and performance. On the one hand, these companies require high speed, to meet extreme time-to-market demands. On

BACKGROUND INFORMATION

DESIGN ITERATIONS AND DESIGN DEBUGS AS VALUABLE LEARNING SOURCE

A Design Iteration (DI, also known as an Engineering Change) introduces functional characteristics to the product. Each iterative step takes the product closer to the final goal: to deliver high quality products, with low costs and low cycle times. In 1999, Jarratt defined a DI as follows:

> *An engineering change is an alteration made to parts, drawings or software that has already been released during the product design process. The change can be of any size, span or type; the change can involve any number of people; take any length of time and can be initiated throughout the product life cycle by any source.*

A Design Debug (DD) emerges when something does not go as planned during assembly or testing of a particular product, which may lead to production being temporarily halted. If an operator comes across an issue, for example a part that doesn't fit or a test which doesn't meet specifications, he registers a Design Debug. It contains information on the error, description, cause, who was contacted, and ultimately also the solution and the interruption time.

[1] *Smart Industry, 2014*

the other hand, they require optimal learning, to prevent expensive development and production errors. In paragraph 3.2 we will show that these two requirements are difficult to combine. Let's therefore have a closer look at each one in the next two paragraphs.

2.2 WE NEED SPEED

Firstly, for speed, cycle time (CT) is a true Key Performance Indicator to manage innovation in an unpredictable business environment, especially during new product development[2]. CT is a better determinator of success of an innovation than focus on output, because it impacts other performance indicators at multiple (lower) levels:

- development and production costs
- time-to-market, ramp-up capability and supply reliability
- technical product quality
- product competitive advantage
- customer-based and financial outcomes

BACKGROUND INFORMATION
CYCLE TIME
Cycle time is the time it takes to create a product from start to finish. In some cases, the entire process is included: feasibility, system design, detailed design, proto, pilot, and volume phases.
In this book cycle time means 'factory time': the time it takes from assembly to start of shipment.

Wrong, or too late?
But CT also influences the way activities are executed. Circumstances changing down the road will influence cycle time management practices. For example, designers and operators must balance a quick fix and progress, versus taking the time and getting to the root of the problem. Customers are waiting for their product, so simply swapping a faulty sensor saves time. Finding out why that sensor broke in the first place would be better because it may prevent faulty sensors in the future, but the customer will get impatient. In other words: *rush and be wrong, or wait and be late*.[3]

2.3 BUT WE ALSO NEED LEARNING

Secondly, we look at learning. Because of the first two challenges mentioned in paragraph 2.1 (high variation and low volume), it is a serious undertaking to meet the third: acquire and keep the required knowledge. But in addition, activities typically:

- are executed concurrently and iteratively
- are complex and specialized
- must be finished yesterday

Those circumstances make it even tougher. That is why optimal policies, to enable integrated learning, can result in significant improvements. Learning curve management is crucial in this respect. We use it to identify the characteristics that drive learning[4].

Use learning to accelerate production
A learning curve (Illustration 1) explains how performance (cycle time, on the vertical axis) improves with increased experience (time, on the horizontal axis). So the more times a task has been performed, the less time is required for each subsequent iteration.

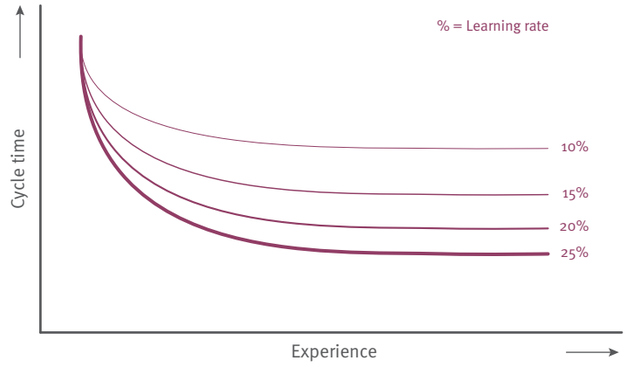


ILLUSTRATION 1
Learning curves for four different learning rates.

The CT gradually improves over time. To be precise, the CT performance improves, with a rate of improvement that generally declines over time[5]. The learning rate, which is part of the learning curve formula, determines how big the improvement is (the 'steepness' of the learning curve). In Illustration 1, the top learning curve has a rate of 10%, and the bottom curve 25%. The vertical end of the top curve is at roughly 50% compared with its starting point. In other words, the cycle time has been cut in half! In production terms, this means for example:

> *With a learning rate of just 10%, after some time a factory can double production with the same resources.*

A learning rate of 20% (the third curve) reaches double production much sooner, and finally reaches quadruple production. In short, learning curve theory helps companies to balance speed and learning with an eye on the future: more experience results in more speed.

In fact, this is the opposite of the usual reflex: save CT by cutting corners. This will only have a short term effect. In the long run, it will result in more errors, which cost more time to fix than the saved CT.

BACKGROUND INFORMATION
EXAMPLE
A production cell for a product may costs millions of euros per year. An unfinished product may represent dozens of millions worth of Work in Progress. Because CT is an important WIP factor, it has been calculated what an improved learning rate would mean in terms of people and production cells. Suppose that 1000 FTE and 100 cells are required if the learning rate is 10% and the CT is 100. If the learning rate is improved to 25%, then the CT would drop to 28. As a result, only 548 FTE and 28 cells would be required!

[2] *Alblas & Langerak, 2015; Cankurtaran et al., 2013*
[3] *Loch & Terwiesch, 2005*
[4] *de Kadt, 2015; Alblas & Langerak, 2015*
[5] *Argote, 2013; Alblas & Langerak, 2015; van Olffen 2015*

So the smart thing to do is to facilitate learning, because a higher learning rate not only accelerates production, it also takes less people. But this focus on the learning curve requires understanding of design driven manufacturing: handling lots of changes, caused by DD's and DI's.

2.4 USE LEARNING MANAGEMENT FOR EVOLVING INSIGHTS

One significant improvement opportunity is in the last challenge mentioned in paragraph 2.1: to effectively manage design iteration and debugging processes during production.

For many companies, it is impossible to anticipate the consequences of their designs due to technical and market uncertainty. In addition, companies use concurrent engineering (parallel development processes) to cut down the time to market. As a result, new insights emerge during development and production, in the form of Design Iterations (DI's) en Design Debugs (DD's). These DI's and DD's, which we defined in paragraph 2.1, are necessary to fix mistakes, to integrate new parts or to tweak the product towards perfection. They should result in improvements, as well as in learning.

Moreover, DI's ripple through the product. Each DI takes the product closer to the final goal of finishing the design, to enable delivering a product of high quality, low cost and low cycle time. But because DI's can't solve all problems, they have a propagative nature, where one change ripples through to other parts of the design. In turn, this fuels the need to solve other problems, previously unforeseen.

Prior research has shown that late problems can cost a hundred times more than early ones[6]. Moreover, time delays can result in substantial profit loss.

It is important that managers recognize the value of iterative design. Instead of regarding testing as part of the verification process, the process of design iteration and debugging can be viewed as an integral part of product development and production. In this process managers, engineers, and operators strive side by side to separate and manipulate cause and effects variables that gives rise to learning. Due to constantly changing environments, the structure of this network of causes and effects is uncertain and evolving. Companies must therefore harvest the knowledge and experience that emerges from evolving insights. Their ability to learn is shown in a learning curve. Smart industries can use learning curve management, to guide managers in coping with these evolving insights, and learning from them. That is what this book is about.

BACKGROUND INFORMATION

EXAMPLE

DI A may fix problem 1 in part X, but this may also, inadvertently, affect parts Y and Z. As a result, three new problems could arise, two in part Y and one in part Z. These must be solved with three new DI's, B, C and D. These new DI's may, in turn, also affect other parts.

[6] *Boehm, 1981; Terwiesch et al., 2002*

2.5 FOCUS ON B TIME FOR CT MANAGEMENT

In the preceding paragraphs we saw that cycle time is an important metric, not just for management, but also for our research. To analyze the performance of insight driven design, we need to measure the effects on cycle time. That is only possible if we split the cycle time into several parts; one that is typically planned, and mostly related to production, and two that can not be planned, and are both affected by learning.

So for our framework for cycle time and learning curve management, we distinguish three different types of cycle time in Illustration 2:[7]

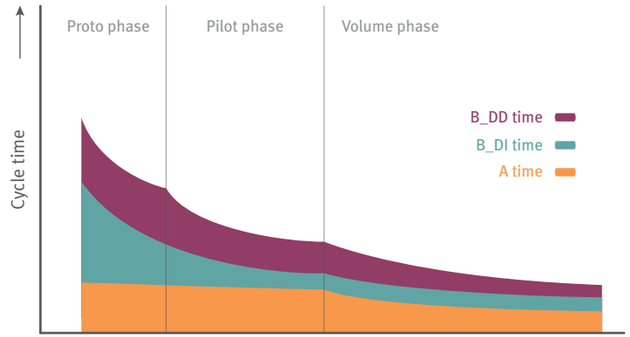


ILLUSTRATION 2

Breakdown of cycle time (vertical axis) into three types, as experience (horizontal axis) increases.

- **A time** is all progress on a product type that is defined in the disturbance free production sequence (blueprint). This is a product type property, which is expected to decrease linearly over time during product maturity, but only minimally, as a relatively small part of the overall CT decrease. A time decrease starts during the volume phase (at the second vertical line in Illustration 2). A time can for example be improved by doing production steps in a different order, so a particular part is more easily accessible.
- **B_DI time** (Design Iteration B time) is all non-planned time required to implement and test DI's. B_DI time decrease starts during the proto phase (until first vertical line in Illustration 2). B_DI time decreases slowly, but now and then briefly goes up when there are product variants.
- **B_DD time** (Design Debugging B time) is all non-planned work or issue time experienced during production. B_DD time decrease starts during the pilot phase (from first to second vertical line in Illustration 2). B_DD time starts out horizontally, because no struc-

[7] *We did not include C time in Illustration 2, because it is not relevant for this book. This is planned non-progress on products, which is quite constant, and includes for example nights, operator breaks and holidays.*

tural time improvements are made during the proto phase. The line starts to follow a learning curve when the design becomes more stable, and less time is spent on DI's. The organization learns how to handle typical problems and solves DD's in a structural way.

The biggest cycle time improvement potential is in B_DI and B_DD time, because they account for a large percentage of the total cycle time, in particular during the proto and pilot phases. Strangely enough, this book is one of the first to focus on B time instead of on A time, while in many smart industries (because of their innovative nature) A time improvement mostly starts when the product design is finished.

In addition, both B times are most affected by learning, so by improving the learning rate it is possible to reduce the cycle time. The question then becomes, what are the controls that managers can operate to influence these cycle time types?

2.6 SLIDE THE SOUND SYSTEM CONTROLS

This is where our framework comes in! We developed a straightforward set of principles that managers can use to improve cycle time, learning, or both. Illustration 3 shows the entire framework as a sound system, with a mixing console at its heart. The manager uses his management principles, the slides on his mixing console, as a filter to produce the right sound for the event: a small jazz tryout (proto phase) or an orchestra in a music hall (volume phase).

In turn, the settings of the mixing console provide operators, engineers and designers with the conditions to perform. Experimental activity requires settings that enable flexibility and improvisation, while repetitive activity requires stability and sheet music.

No jazz band consists of 100 musicians, and no symphony orchestra plays without sheet music. In the same way, smart industries need smaller teams capable of improvisation in the beginning, and later on a larger group of operators using strict work instructions.

1. **Process measures**
   Statistic tests have shown that these nine process measures have an effect on cycle times and on learning. They are values that can be measured exactly. These are the properties of the input signals for the mixing console, like balance, speed or frequency range.
2. **Management principles**
   To achieve cycle time reduction and learning effects, managers can operate a handful of controls: the slides on the mixing console. These controls act as a filter on the properties of the individual input signals (the process measures), and the combined settings

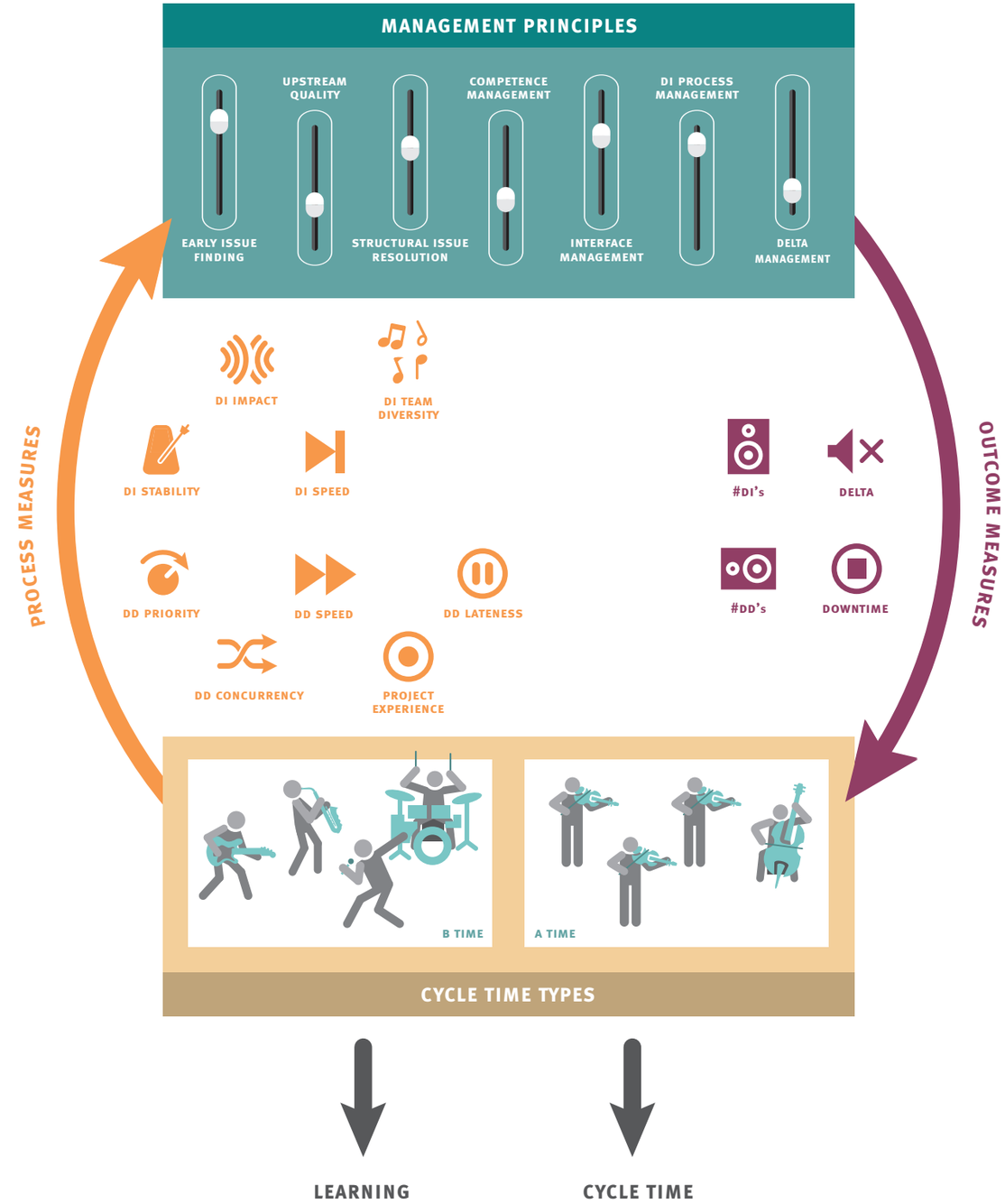


ILLUSTRATION 3
The cycle time and learning mixing console, at the heart of the sound system framework.

influence the output channels (the outcome measures) for a particular type of music. For example:

- Early Issue Finding will affect DD Lateness
- Upstream Quality will influence DD Concurrency, but also DD Lateness
- Structural Issue Resolution will impact DI's, and the number of people involved in them (DI Speed)

3. **Outcome measures**
   The result is sent to the sound monitors and speakers: the output channels:

   Number of Design Iterations (DI's). This strongly relates to product delta's. One could think of DI's as the soundcheck prior to a concert: the sound engineer uses the concert hall speakers to optimize the settings.

   Number of Design Debugs (DD's). This strongly relates to product downtime. DD's strongly resemble feedback during the concert: the musicians use their stage monitor speakers to check their sound.

4. **Cycle time types**
   The result of all of the above is a combination of mixing console settings, optimized for a particular type of music. A jazz combo or a blues band requires settings that enable improvisation, walking around, and now and then a solo. In other words: they need settings that focus on B time. On the other side of the spectrum is predictability: a string quartet making a studio recording using sheet music, or a band like The Eagles with quite static microphone positions on stage. These types of music lean more towards A time.[8]

5. **Cycle time and learning**
   Finally, the A time and B time have an effect on both the cycle time and on learning. A lower cycle time has a direct effect on one single product, while learning affects the cycle time of all subsequent products.

This book focuses on B time, because its improvisation characteristics provide the best opportunities to learn.

Shifting management principles

This book discusses the framework outlined above, but in chapter 6 we zoom in on phases. Because as the product advances (from prototyping, via pilot production, to volume production), the required management principles must shift accordingly. In other words: the mixing console settings need constant attention and adjustment. That's why there is a feedback loop in the framework: the arrow from cycle time types, via process measures, back to the management principles. For example, managers can fade out Early Issue Finding, and fade in Delta Management.

The same mixing example can be applied to these phases, where the mixing console settings depend on the development phase of the music:

- In the proto phase, the best mixing console settings still have to be optimized for the specific audience and music style (tryout).
- During the pilot phase, mixing console settings have been optimized, but still have to face a larger, official test (premiere).
- A concert tour corresponds to the volume phase: all settings have been finalized for all songs on the set list, and the mixing console has been labeled accordingly.

THE ULTIMATE GOAL: TO KEEP B TIME UNDER CONTROL

While for the A time a well-founded estimate can be calculated, the B time is currently not only the most uncertain, but also the most prominent component of production time. In some smart industries, the A/B time ratio currently is 1:20, while according to various managers the acceptable ratio is 1:1.[9] In other cases the ratio is 4:1 for assembly, and 1:1 for testing.

So, B time is the largest component of cycle time. That's why it is also an important factor for learning curve drivers when producing a new product type. The largest percentage of B time is due to issues during assembling and testing of a new product type: time spent on solving DI's and DD's.

From an academic point of view, the factors determining B time length were unclear up to now. The research results presented in this book clarify a considerable part of that B time mistery.

BACKGROUND INFORMATION

PROTO, PILOT AND VOLUME PHASE

Phases relate to the maturity level of the products, and their identification is often based on the number of products produced. However, the numbers are branche dependent. In some industries there is only one prototype, and after a handful of pilot products the volume phase starts at seven or eight products. Other industries need dozens of prototypes and hundreds of pilot products to reach the volume phase. In this book, the definitions are as follows:

Proto phase: The first series of products, with focus on design. This is the phase with a high cycle time, in which the large amount of technological and design uncertainty is reduced.

Pilot phase: The next series of products, with focus on process improvement. Technologies and designs have been proven to a point that the products are shipped to customers.

Volume phase: All subsequent products. Manufacturing is fully responsible and involvement of development is minimal.

[8] *C time is left out here, because it is predictable and constant. It is not influenced by management settings.*

[9] *Voogd, 2014*

Learning for the long distance run

As we saw in the previous chapter, smart industries tend to have high variability and low production volumes. One might therefore conclude that learning is useless, and that focus should be on production efficiency only.

This chapter uses the quadrant model and the seesaw effect to show that it pays off to take time for learning and product improvement, especially in smart industries. The cycle time will eventually go down, and with much more than the time spent on learning. We also show that learning is driven by design iteration and debugging.

3.1 BALANCE CT AND LEARNING CURVE MANAGEMENT

Let's first consider a fundamental difference between cycle time management and learning curve management. The former focuses on improving the CT of an individual product. The latter improves the CT of all subsequent products. Illustration 4 shows a stylized learning curve.

- With **cycle time management**, one dot will move down in the graph. All other dots will remain unchanged.
- With **learning curve management**, the current dot will go up, but all dots to the right of the current product will move down!

In the early phase of the learning curve (on the left side), there is ample room for redesigning processes and cutting slack from inefficient processes. In later phases however (more to the right), these opportunities are reduced and, consequently, improvements

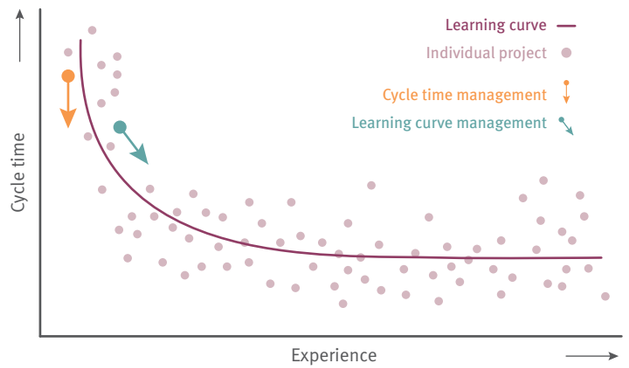


ILLUSTRATION 4
The learning curve: as more products are produced, cycle time decreases at a decreasing rate.

EXPERT INSIGHT
FIRST PRODUCT ON TIME, OR ALL ON TIME?

"The first few products always have unforeseen issues. You never know how many there will be or how complex they are to fix. The smart thing would be to take a step back to find the root cause. That takes time, but it will prevent those issues in future products. It improves learning and reduces the cycle time of all subsequent products.

But those products have been promised to customers who want them on time. So to save time, in many cases a workaround is used or a part is swapped. Properly fixing the issue is postponed to the next product, where the same dilemma will occur.

Output purists do not want to search for the root cause. The product must be shipped on time, no matter what. Some even believe that issues get fixed sooner by simply setting an ambitious delivery date. Learning curve purists prefer to stop production to fix it once and for all. I remember one time when we stopped the first product to properly fix five DD's. That was a huge investment, but it paid off in all products after that. It takes balls to put a product on hold, the customer doesn't care about future products, he wants this one as soon as possible. It is a difficult tradeoff between early shipment or to hold shipment: it is the voice of the current versus future customers. The dilemma is hard to explain to them.

Another case was when we had a downturn after finishing only six products of a new type. I was worried about the upturn moment, because we did not properly complete our learning curve. But it turned out that we managed to define a reasonable list of issues to be solved and we had ample time to address them during the downturn. By the time the upturn came, almost everything was under control. I learned that even a small number of products can provide a solid learning curve base."

in work processes stem more often from autonomous learning by employees.

Learning curve management balances the immediate output (CT) and structural CT improvements (LC). This means that companies trying to achieve cycle time reductions of their projects, must reach this goal with an eye on the future, balancing speed and learning effects.[10]

This dilemma is illustrated in the expert insight *First product on time, or all on time?* Juggling speed and learning requires knowledge of the seesaw effect, which we'll discuss next.

BACKGROUND INFORMATION
APPROACH AND METHODOLOGY

This book is the result of a research project that aimed to investigate drivers and mechanisms of learning in a high-tech product development and production setting. The research project took place from 2014 to 2016 in the development and production departments.

Data was collected by using several sources. The research team inspected the company data archives for records best describing the properties of design debugging and design iteration activity over time. Data on project cycle time, number of design debugs and design iterations formed the bases of our database. Furthermore 100.000s archival records of debugging incidents and 10.000s engineering change reports were used. These data were all combined into a single database that comprises over 500 NPD projects that were commercialized between 2005 and 2014.

To scrutinize the drivers and mechanisms of learning we tested statistical models using software such as SPSS. The aim was to find common patterns of learning, and because the data set comprised a variety of different product families and product types, dozens of models were tested and compared, based on scientific validity and reliability criteria.

In addition, we used semi-structured interviews with several key informants to validate our research findings. The qualitative data was transcribed and analyzed by using software such as NVivo. In addition, we corroborate our findings with anecdotal evidence and expert insight that are also used for illustration purposes in this book.

[10] *de Kadt, 2015*

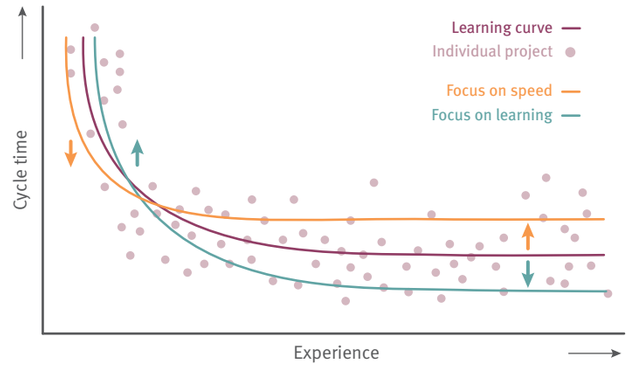


ILLUSTRATION 5
Learning curves with focus on learning (–) versus speed (–).

3.2 THE SEESAW EFFECT: BALANCE LEARNING AND SPEED

To balance CT and LC management, you need an understanding of the seesaw effect of the learning curve. As we saw earlier, we have two competing scenarios:

- One with focus on **learning**
  Use early products to learn and improve

  Learning takes time, for example to find structural solutions to issues, or to find the root cause of issues. This will mean longer cycle times, especially in the beginning. But as more products are built, cycle times become shorter very quickly.
- One with focus on **speed**
  Finish early products as soon as possible, no matter what

  An early pressure on cycle time hampers learning opportunities. As a result, unforeseen problems may emerge in a later phase, which could have been discovered in an earlier phase.

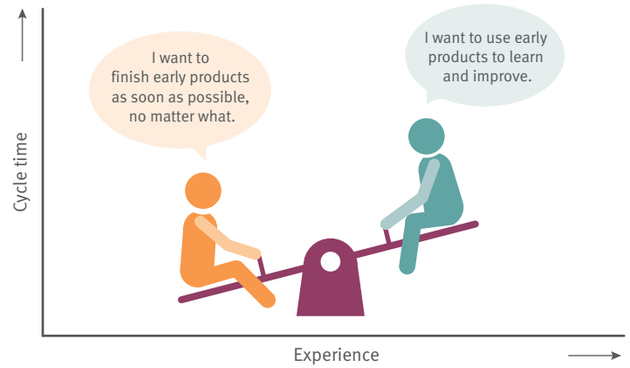


ILLUSTRATION 6
The seesaw effect: balancing output and quality.

BACKGROUND INFORMATION
LEARNING CURVE, PROGRESS RATIO AND DRIVERS

Time to introduce some equations of the underlying math. The classic form of the learning curve (Illustration 4) can be expressed as follows:

Equation 1 $CT_i = CT_0 X_{i-1}^{-b}$ *(where CT_i is the number of labor hours per product, b is the learning rate, X_{i-1} is the cumulative number of products produced through time period i-1, and CT_0 is the number of labor hours required to produce the first product.)*

From learning rate to progress ratio

Learning curves are characterized in terms of a progress ratio (p). Equation 1 describes that with each doubling of cumulative output, the product CT is reduced with a certain percentage (p). Parameter b in Equation 1 is related to the progress ratio, p, as follows:

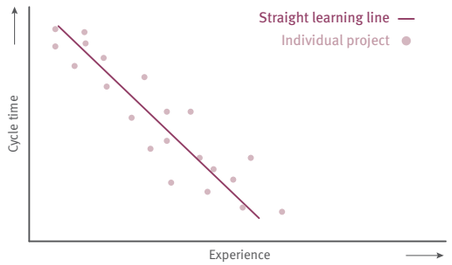
Equation 2 $p = 2^b$

For example, if the first product was produced in 100 hours (CT_0), and if this is the fifth product (X_i), and the learning rate is 0,15 (b), then the number of hours for the fifth product is $100 * 5^{-0.15} = 78,5$ hours. The progress ratio is then $2^{0.15} = 1,1095$.

From a curve to a line

The classic learning curve becomes a straight learning line, if Equation 1 is converted to a logarithmic scale. This is the most widely used model. This is expressed as follows:

Equation 3 $\ln CT_i = ai + bi \ln X_{i-1}$ *(where a_i is constant and b_i = the learning rate.) Equation 3 describes the decrease in cycle time when the cumulative number of products increases.*



Including drivers

Finally, Equation 4 (the learning curve) can be expanded to include driver effects. The driver is one variable influencing both learning rate and cycle time, and therefore responsible for the deviation of the line. The equation then becomes:

Equation 4 $\ln CT_i = a + b_1 \ln X_{i-1} + b_2 * DRIVER + b_3 \ln X_{i-1} * DRIVER$ *(where b_1 =is the learning rate and b_2 is the impact on cycle time).*

According to Illustration 5, best results would be achieved if the early phase would focus on speed (—), while still providing learning opportunities (—). In the next phases, those lessons learned can then be used to keep cycle times in the volume phase short.

In other words: each phase requires specific priorities, to keep the balance, and to prevent the 'seesaw' from tipping over to one side (see Illustration 6).

3.3 USE THE QUADRANT MODEL FOR CT AND LC MANAGEMENT

During our research we tested many different models. To present the results in a comprehensible way, we used a quadrant model. The model is used to score the process measures (the input channel properties in our sound system framework).

The following paragraphs will introduce the model, and systematically list and explain the scoring differences.

3.3.1 CYCLE TIME AND LEARNING DRIVERS

Let's first summarize some important concepts from the background information Learning curve, Progress ratio and Drivers on page 25:

- learning curve (Equation 1)
  The function that expresses the decrease in CT when the cumulative number of products increases. To predict the CT of the last product, three values are required: the CT of the first product, the cumulative number of preceding products, and the learning rate.
- driver (used in Equation 4)
  A variable that influences both CT and learning rate.
- progress ratio (Equation 2)
  From the learning rate it is possible to calculate the progress ratio, which provides insight in the reduction percentage of CT, when the cumulative production doubles.

In Equation 4 of the background information we introduced a driver. It impacts CT directly (b_2 * DRIVER), and has a moderating effect on the learning rate (b_3 ln X_{i-1} * DRIVER).

This is a very important distinction, and it is crucial to successfully reduce CT using:

- issue resolution - the process of decreasing the number of Design Debugs that occur during production
- learning and self-reliance - the ability of an organization to handle new situations and to solve them independently
- delta management - the process of determining which parts can be redesigned (Design Iteration) and which parts should have a design freeze

3.3.2 CYCLE TIME AND LEARNING QUADRANTS

Building on the previous paragraph, we can combine the two effects in the two-dimensional graph of Illustration 7. On the vertical axis, the impact on CT is plotted: the higher the better. On the horizontal axis, the moderating effect on the learning rate is plotted: the more to the right the better.

This gives us four quadrants, where the top right corner is optimal (good (■) for both learning and CT) and the bottom left corner reflects a combined worsening effect (bad (■) for both).

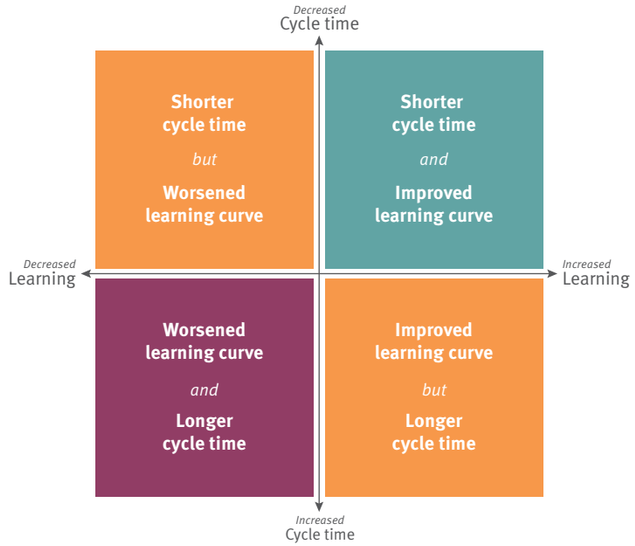


ILLUSTRATION 7
Impact on CT is plotted vertically. Moderating effect on learning rate is plotted horizontally.

Two interesting results stood out from our research: some product families learn faster than others, and learning differs across phases. The following paragraphs provide more insight in the reasons for these phenomena.

3.3.3 CYCLE TIME AND LEARNING IN THE REAL WORLD

This paragraph will show the diversity in learning and cycle time across product families and phases. Illustration 8 contains an example based on actual company data. The research data spans a period of ten years, including two product families, each with three products.

All products of product family A are located near the center of the graph; they vary little in CT and learning effects. The third product of that family (A3) is used as reference for the next generation of products (B), where learning effects are good for B2 and B3, but CT increases for B2. For B1 there is a decreasing learning effect, but CT is much better.

If a learning curve is drawn for each of the six products, some of the products show a considerable number of deviations. These deviation points appear above the learning curve, indicating a longer cycle time than expected.

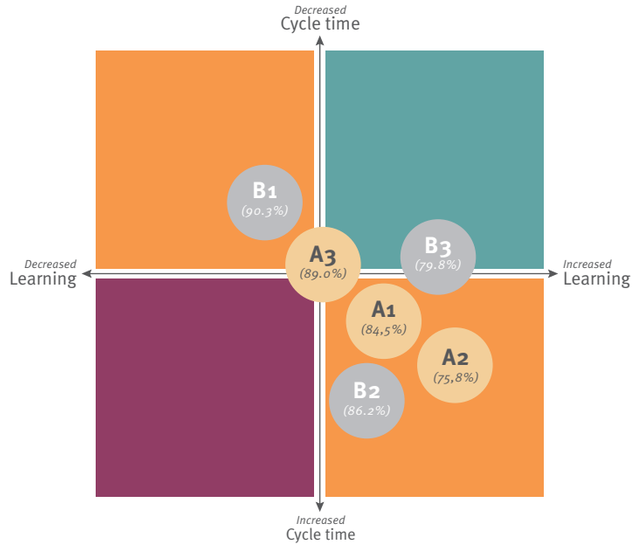


ILLUSTRATION 8

The effect on cycle time and learning for two product families, including progress ratios.

The data becomes even more interesting when we calculate the progress ratio, which is also shown in the illustration. As product family A develops from A1 to A3, the progress ratio increases from 76% to 89%. But for product family B, it decreases from 90% to 80%! One possible conclusion is that learning was less in A, and better in B. But there are also learning deviations per phase within the product families, as Illustration 9 shows.

Product family A shows an almost equal CT for all three products for the pilot phase, but the A2 proto CT is almost twice that of A1 and A3.

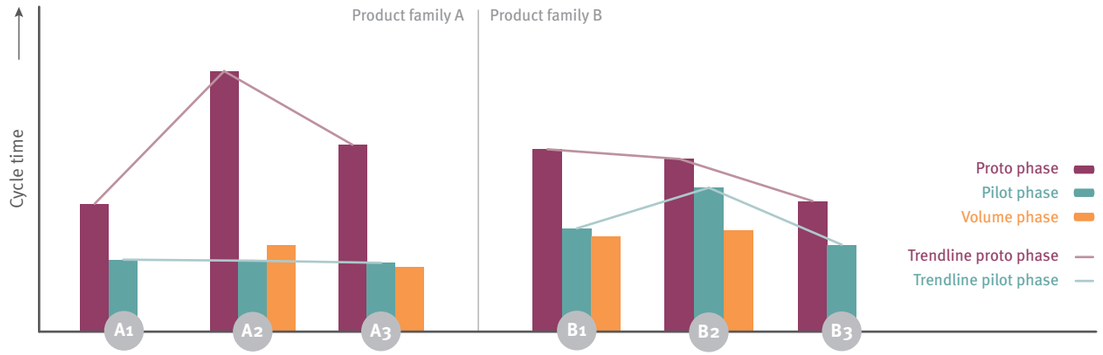


ILLUSTRATION 9

Average CT per phase, for each product of the two product families.

EXPERT INSIGHT

PRESSURE ON CYCLE TIME: CUSTOMER DRIVEN STARTS PER WEEK

*"It's difficult to attach cycle times to phases, but it pays off to take time for the first few products. The first pilot product may take a year. Our volume cycle time, on the other hand, is often customer driven: if they ask for two products per week, we simply start two per week. Because the work in progress (WIP) is starts per week * cycle time, the WIP increases linearly (1:1) with the starts per week. To keep the complexity, predictability and output of the factory under control, only CT can be manipulated, because reducing the move rate would result in less products for the customer, and that is not an option. We usually start calling it volume manufacturing when we reach a cycle time of 20-25 weeks, but only when we reach 16 weeks we are really in control.*

Also the proto cycle time may be customer driven. The customer needs our latest product to produce their next generation of articles. In some cases they wanted that prototype so badly that they demanded to have it on site, no matter what, even when it did not yet work properly. Then it takes teams of dozens of development engineers to get it to work on site. We support our customers well, we have a high customer focus score. It is part of our company culture."

For product family B, that peak occurs in the pilot phase instead of the proto phase, although the effect for B2 is not as prominent. The proto cycle times for product family B on the other hand show steady improvement from B1 to B3.

So we can conclude from the data above that there are large differences in learning and cycle time, but what are the reasons one product is doing better than the other? The management framework in this book opens the black box. Because the mixing console slides make it possible to steer learning and cycle time, instead of letting them happen.

DESIGN ITERATION AND DEBUGGING DRIVES THE LEARNING CURVE 3.4

The previous paragraph showed actual company data. This paragraph takes the same data, but examines what happens if one particular parameter, learning, is 'switched off'. In other words, there is no learning: each DI is treated as if it were completely new.

While a product is developed, a continuous stream of Design Iterations and Design Debugs has to be processed. For high-tech products, hundreds of DI's may be open simultaneously. In addition, many DI's will in turn generate DD's, which have to be handled and solved as well. These DI's and DD's are the main reason why there are so many differences in learning, because their interaction determines the learning process.

Scenario: no experience

It is extremely important to learn from mistakes and to develop an advanced understanding. Illustration 10 uses a scenario analysis, based on actual data, to show what would happen to the cycle time

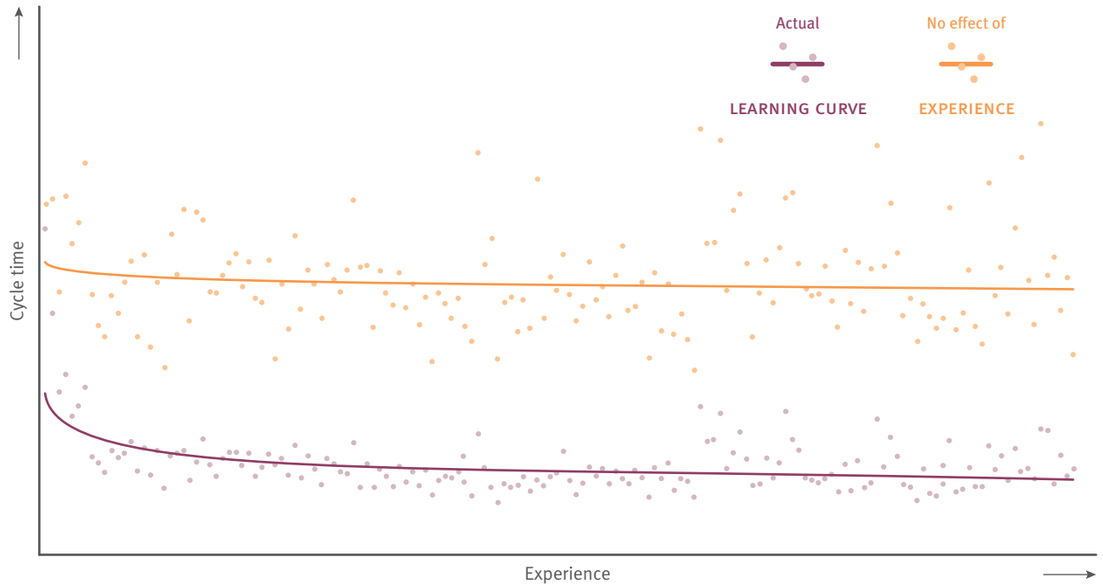


ILLUSTRATION 10
Cycle time with learning (–) or without learning (–).

(on the vertical axis) if there is no learning whatsoever (—), compared with the normal scenario, the actual data, including learning (—).

So, in this theoretical scenario when there is no learning at all, cycle time is two to three times higher than with learning. This clearly shows how important learning curve management is. The variations (the dots further away from the curve) can be explained by the drivers we discussed earlier.

IN CONTROL WITH THE MIXING CONSOLE

What this chapter shows is that learning is inevitable: without learning, cycle time goes through the roof. But managers do not need to be 'victims' of learning and cycle time: they can use the mixing console slides to steer the balance between learning and cycle time. Depending on the venue and the type of music, managers can choose for example to open up the Early Issue Finding slide (a management principle), to make the DD Lateness input channel property (a particular process measure) more prominent, which would limit learning.

The rest of this book uses the quadrant model to score all process measures of our framework: the input channel properties of the sound system. Chapter 4 explores DI related process measures, chapter 5 explores DD related process measures, and chapter 6 takes phase dependency into account.

4

Learning from design iteration

In the previous chapter we established that design iterations and design debugging drives the learning curve. This chapter zooms in on the former, with an emphasis on two DI management principles: Delta Management and DI Process Management. The next chapter will deal with design debugging.

4.1 TWEAKING TOWARDS STABILITY

In paragraph 3.3.2 we introduced the cycle time and learning quadrant model. This paragraph takes a closer look at the role of DI Stability and DI Impact.

Experimental strategies involve processes that consist of iterative steps to hunt for possible solutions. This process typically comprises iterative cycles of design, build, test, analyze, and learn. The process is repeated until the results are satisfactory. Design iterations may involve changes in product designs, test process changes, or improvements in the technical solutions. DI's are therefore mostly caused by uncertainty in technology. But they can also emerge from market uncertainty, serving as a variety generation mechanism that enables to develop bespoke products. Illustration 11 shows the effects of DI Stability and DI Impact.

The effects of these two process measures clearly illustrate the important balance that managers need to keep. Because many companies deliver products that are not strictly one-of-a-kind, but instead products sharing a certain degree of similarity, managers must balance stability and flexibility in their product design and processes[11]. On the one hand, they want to exploit existing technology, while at the same time they explore new innovation possibilities, implemented by high impacting DI's. Managing DI Stability and DI Impact plays an important role in this balancing act.

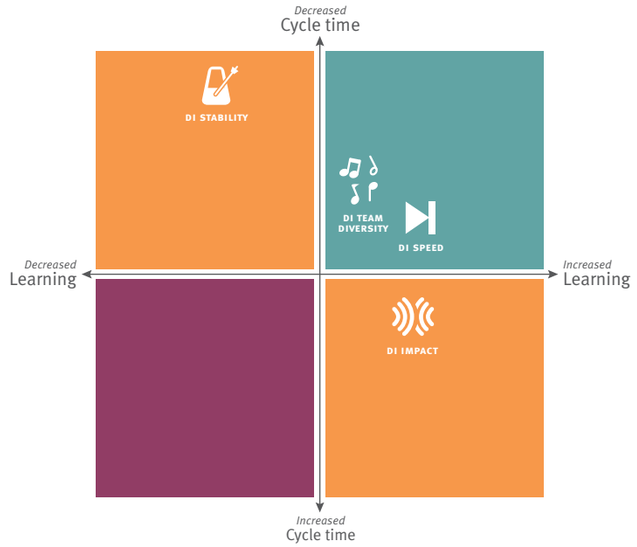


ILLUSTRATION 11
DI Stability and DI Impact in the quadrant model.

We can analyze them at product level, but also at module level.

DI Stability is measured at module level. Each module is a more or less independent entity, and adds its own functions to the whole product. A module is considered stable if it has a low number of DI's during production. This means that the module only requires a few changes to achieve acceptance. Illustration 11 shows that DI Stability does not improve learning (■), but it does improve cycle time (■).

DI Impact is defined as the number of DI's 'in progress' during production. Illustration 11 shows that DI's cause longer cycle times (■), but improve learning (■).

4.2 FROM LOW STABILITY TO SHORT CYCLE TIME

Illustration 12 uses a scenario analysis, based on actual data, to show what would happen to the cycle time (on the vertical axis) in case of no DI Stability (—), and in case of no DI Impact (—). The middle curve (—) shows the actual cycle time.

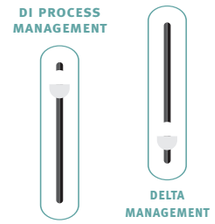
Scenario: no DI Stability, or no DI Impact

The top curve shows that a scenario without DI Stability per module will cause a learning effect, decreasing the CT for subsequent products.

The bottom curve would reflect the situation if there is no DI Impact at all during production. This will seldom occur in smart industries with high-tech products, but it is the norm in an assembly line for countless simple and similar products, for example a high volume factory. We can conclude that DI's drive learning, but require time.

To influence these two process measures (DI Stability and DI Impact), two management principles were developed. The rest of this chapter focuses on these two management principles (the slides of the mixing console), which we introduced in Illustration 3 of paragraph 2.6.

- Delta Management: handling DI's in the factory, and dealing with upgrades
- DI Process Management: setting up conditions for learning



[11] *Veldman & Alblas, 2012*

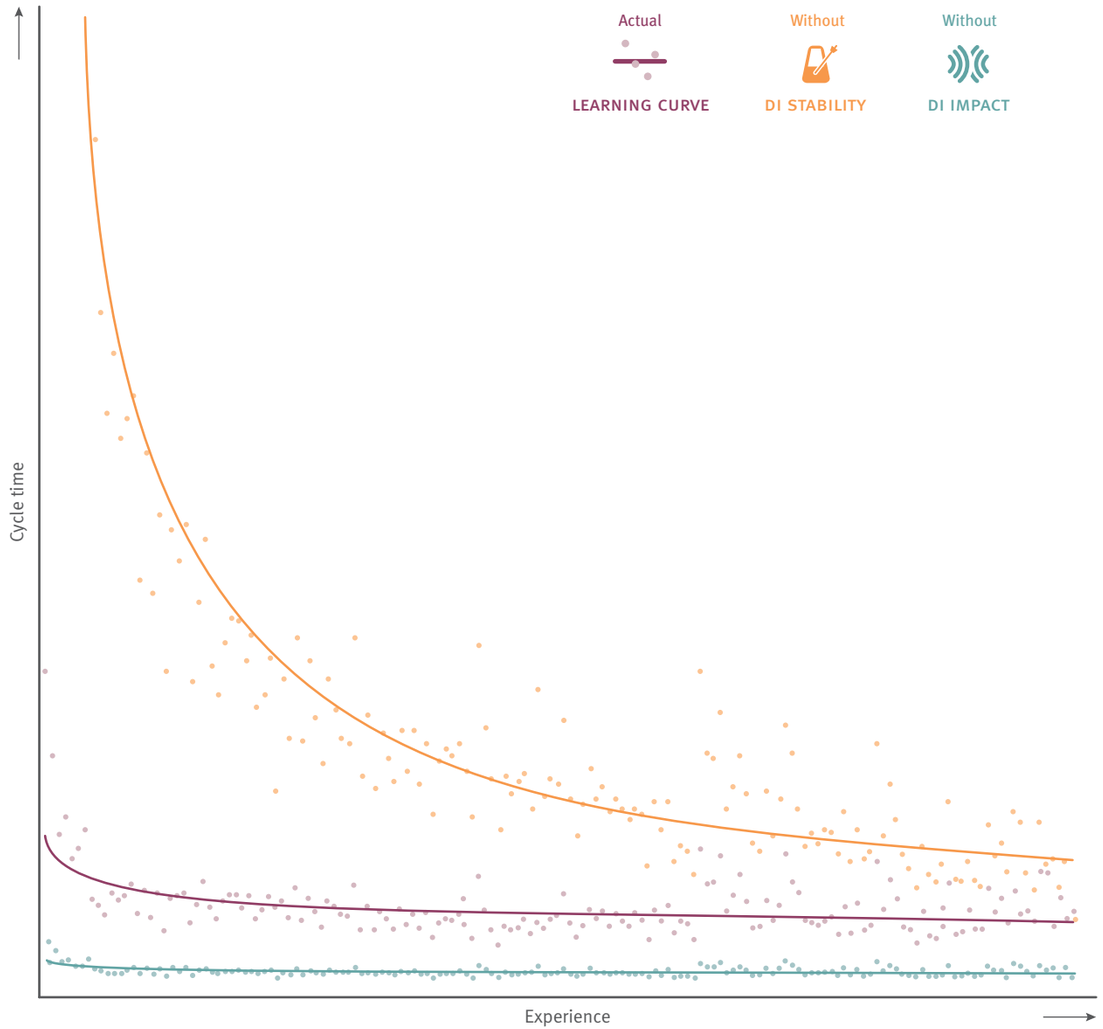


ILLUSTRATION 12

Cycle time without DI Impact (–) or without DI Stability (–).

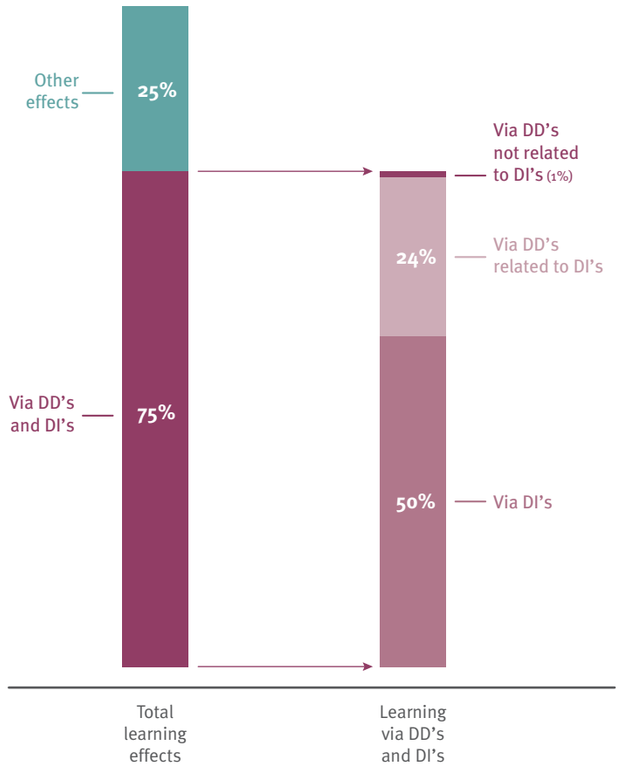
DESIGN ITERATION MANAGEMENT PRINCIPLES

4.3

Illustration 13 shows a breakdown of the various effects on CT within a single product family.

- The first column, the learning curve, accounts for 80% of the total learning effects on CT. The remaining 20% can not be attributed to any of the factors we discuss in this book.
- The LC can be split into two effects: DI's and DD's (75% of that 80%) and other effects, like experience (25%).
- In turn, that 75% can be split three-way:
  - 50% DI's
  - 24% DI's related to DD's
  - 1% just DD's

The above breakdown clearly shows the large effect that DI's have on DD's. But the main conclusion is the importance of managing DI's: 74% of the learning curve effects on CT can be attributed, directly or indirectly, to DI's.



EXPERT INSIGHT

DO NOT ADD WINDOWS AFTER THE WALL HAS BEEN BUILT

"Up until half a year ago we would start implementing all upgrades when the product was assembled completely. It kept the process orderly and uncluttered. But on the other hand, at that point the product was most expensive, and closest to customer delivery. We realized that it is possible to implement upgrades at an earlier stage. All it took to prevent upgrades during final assembly and testing, was to update the stock immediately, based on the design change. We now put the upgrade part in the product, eliminating the need for a swap later on. This is so logical and obvious, that it seems striking that it took us so long to do this. The reason was that it has to be balanced against what the customer wants, because there is always even later and greater technology which become available just after assembly.

There are many industries, some technologically less complex, where last minute requirements updates cause changes or upgrades. Buildings can have an alarming number of changes. It is OK to come up with them, but at some point you have to accept that the wall has been built, and that it is not allowed to add more windows."

ILLUSTRATION 13

Breakdown of the various effects on cycle time.

EXPERT INSIGHT

IF IT WORKS, DON'T FIX IT

"In our branche it is not uncommon to have 1000 development engineers, working on all competences, on every module of a product. These are smart people, they want to make improvements, and that means hundreds of redesigns in six months. The natural reflex of operations is to postpone these changes, because it may cause delays or issues. This means that when the next model of the product is introduced, a huge reservoir of redesigns is released. That swamps my entire pilot product with changes. Suddenly I have issues all over the product. It is much better for the company as a whole to have a steady undercurrent of changes, instead of opening the floodgate.

The key to Delta Management is 'if it works, don't fix it'. If a module is not working perfectly, but the customer is happy, then why change it? Design iterations should not automatically end up in the current product, or the next model. They should be implemented calmly, consciously and deliberately."

4.3.1 DELTA MANAGEMENT: SELECT WHAT TO CHANGE

Delta Management is the process of managing DI's and DD's in a production environment. It focuses solving power on the areas that need to be changed, based on customer requirements embodied in the product road map. Main goals of Delta Management are:

- Limit changes to a clearly defined part of the product, and to 'freeze' the rest
- Clear decision making and communication about design freeze
- Roadmap for new module releases and related upgrades coordination, together with product development
- Clear communication about changed modules, and to focus solving power on these areas
- Prevent 'engineering delight', as illustrated by the expert insight *If it works, don't fix it.*

Thus, Delta Management focuses on managing the DI Impact, by freezing as many parts of the product as possible, and changing only those parts that really require modification, for example due to a new generation, or particular customer demand. If changes to initially frozen parts turn out to be absolutely necessary, they should be extensively researched and tested before implementation. The product should remain OK or become better. If no parts are frozen, then changes will occur all over the product, including not completely finalized parts, and CT will certainly suffer.

In the accompanying background information we summarize research that distinguishes between low and high impact DI's, to examine the influence of experience, via DI's, on CT, as Illustration 14A shows.

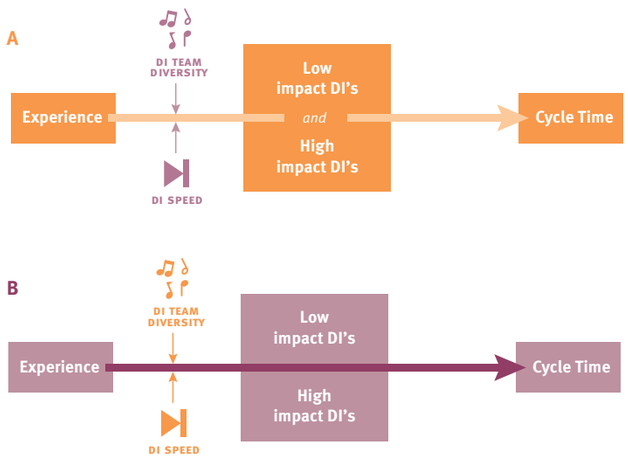


ILLUSTRATION 14

A. The influence of experience, via low/high impact DI's, on CT.

B. What drives the direct effect of experience on CT?

BACKGROUND INFORMATION

DESIGN ITERATIONS CAUSE LONGER CYCLE TIMES

Illustration 14A shows the factors influencing CT. The factors are partly interdependent. We analyze all factors separately, to better understand their individual influence on CT.

First, we focus on the center parts of Illustration 14A: low and high impact DI's. When a DI is approved, the structure of the product changes, and the DI must be implemented. Each DI is assigned a necessity level, depending on the perceived urgency:

- *DI's with a high necessity must be implemented earlier in the production process, on products in process. They therefore increase cycle time directly (■), because the B time increases.*
- *DI's with a low necessity can be postponed to the supply chain, so they do not impact CT directly. But they do have an influence on support processes like service parts, so they increase cycle time indirectly (■).*

The general rule of thumb resulting from our research is that high impact DI's influence CT more strongly than low impact DI's[12]. But keep in mind that this is not just the time required to implement the DI. In many cases the DI will also affect processes, tools or business functions, and it takes additional (learning) time (■) to get used to those changes.

Experience results in less design iterations

Next, we look at the influence of experience on DI's, regardless of the subsequent consequences for CT. Our research shows that the direct effect of experience on DI's is statistically significant. We measure experience in the number of cumulative product projects. As experience grows, more DI's are solved, and more and more knowledge on how to develop and produce the products is built up. As a result, less DI's are required to fix issues in product design.

Design iteration learning yields shorter cycle times

Finally, we look at the influence of experience on CT, which occurs via DI's. The influence of this factor is twofold, and may seem contradictory.

On the one hand, as we saw earlier, experience leads to a lower number of DI's. And less DI's to process means a better cycle time (■). But on the other hand, DI's are a prominent learning source, and they account for around 50% of the learning curve. So while the number of DI's decreases, learning opportunities will decrease as well. As a result, the learning curve will not improve as much (■).

The direct effect of experience on CT is statistically significant: as experience grows, more DI's are solved. High impact (urgent) DI's influence CT more strongly than low impact (postponed) DI's, which have an indirect effect.

4.3.2 DI PROCESS MANAGEMENT: USE SPEED AND DIVERSITY

The previous paragraphs focused on Delta Management, to manage the impact of DI's. In the next paragraphs we zoom in on DI Process Management, and particularly on the effects of the time required to finish DI's (DI Speed), and the number of departments involved (DI Team Diversity). Both factors have different effects on the Delta Management factors discussed earlier. The main goals of DI Process Management are:

- Distinguish between problem definition, solution development and solution implementation
- Strive for cross-functional involvement of product development, manufacturing and service
- Prevent an uncontrolled stream of DI's

[12] *de Kadt, 2015*

In Illustration 14B we identify two moderators of a DI:

- Design Iteration Speed: the time it takes to finish the DI (throughput time). We call this the moderator DI Speed. Higher DI Speed is associated with a higher learning rate and an improved CT.
- Design Iteration Team Diversity: the number of departments contributing to the DI: the moderator DI Team Diversity. Higher DI Team Diversity is beneficial to both learning and CT.

Both moderators are analyzed in more detail in the accompanying background information.

DESIGN ITERATIONS PROPEL PROGRESS

This chapter was all about DI's, and the next one will be about DD's. But Illustration 13 shows that the two factors are intimately related, and that design iterations have a huge impact. Progress in solving design iterations will result in a better learning curve, because in the end, without DI's also the number of DD's will drop to zero.

So, what is the ultimate conclusion of this chapter? What does a manager have to do in the early phases? Make sure DI's are solved quickly, and by multidisciplinary teams, using Delta Management.

BACKGROUND INFORMATION

DI SPEED AND TEAM DIVERSITY IMPROVE CYCLE TIME

Both characteristics are drivers of the learning curve, they lead to a steeper learning curve (■). This applies to the direct effect of experience on CT.

Long design iterations hamper learning

Next, the influence of DI Speed on DI's is analyzed, both low and high impact. We found that increased DI Speed is good for learning (■). This applies to both low and high impact DI's. A possible explanation for this is that the longer it takes to finish a DI, the more it clogs up the entire DI process, and the more likely the knowledge involved will become obsolete during that time.

The opposite may be inferred from this: managing DI congestion and processing capacity will improve the whole DI process, which in turn will benefit learning (■).

Team Diversity improves learning

DI Team Diversity, the other characteristic, also influences both low and high impact DI's. While a DI is processed, engineers work together to come up with a satisfying solution. Each engineer brings unique experience to the table, viewing the issue from a different perspective. And the earlier in the process they get involved, the more likely it becomes that the DI will fix the issue efficiently. This combined effort indirectly improves learning (■).

5

Learning from design debugging

In chapter 3 we established that design iterations and design debugging drives the learning curve. The previous chapter zoomed in on the former; this chapter will deal with the latter.

We started chapter 4 (Learning from design iteration) with Illustration 11: DI Stability and DI Impact on the quadrant model. In this chapter, we expand that illustration with process measures related to Design Debugging. After that, we introduce the management principles we developed to influence those process measures.

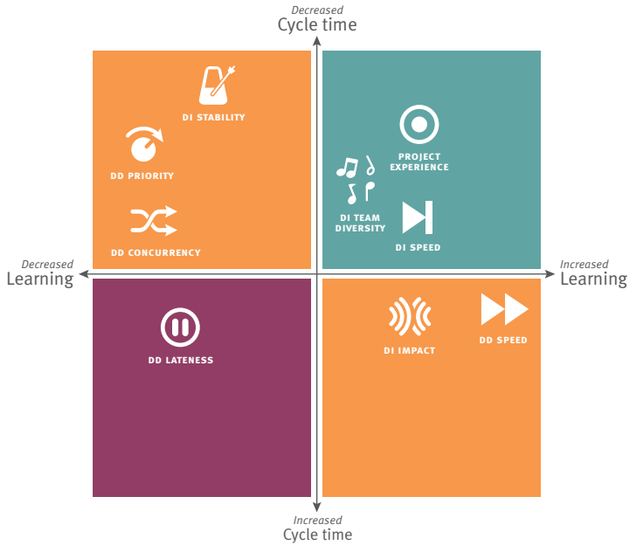


ILLUSTRATION 15
Design debugging in the quadrant model.

Illustration 15 introduces several new process measures. They match five of the process measures (in terms of the sound system framework: input channel properties), which we introduced in Illustration 3 of paragraph 2.6.

DD Priority: the average priority of Design Debugs on a product. The priority is assigned by management. This process measure is used to focus the limited capacity on the DD's that should be focused on first. Illustration 15 shows that DD Priority (working with priority setting during production on DD) does not improve learning (■), but does improve cycle time (■).

DD Concurrency: the total number of DD's, within the time window of an individual product or project, for all products in progress within the same product family. This does not improve learning (■), but has little effect on cycle time.

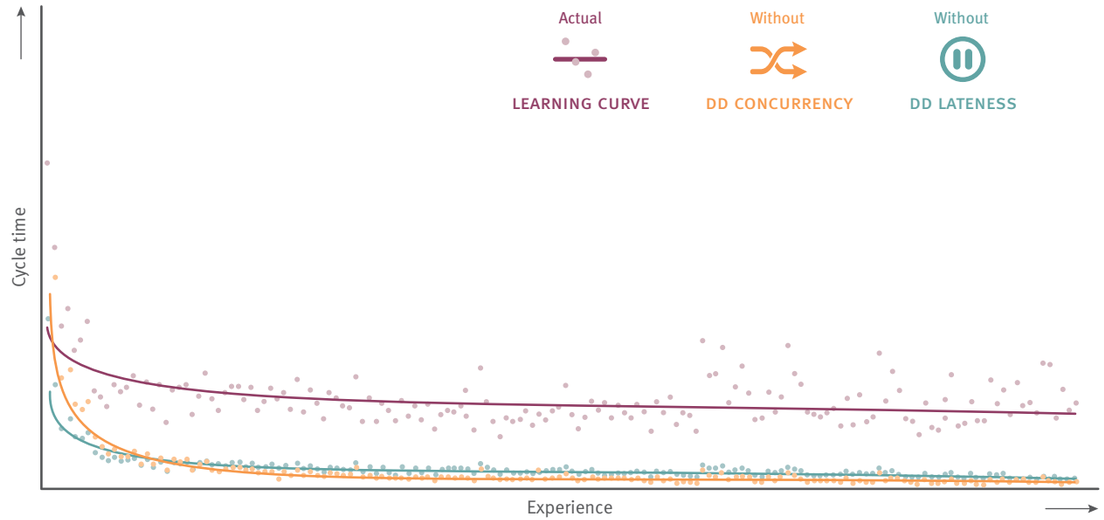


ILLUSTRATION 16
Cycle time without DD Concurrency (–) or without DD Lateness (–).

DD Lateness: a DD can occur early on in the production process, or near the end, when the product is almost ready. The later a DD occurs, the more difficult it is to solve or to find the root cause for each DD. To determine this process measure, the solution date is compared with the date of the very first DD on a particular product. The difference between those two dates (DD Solution date – First product DD date) gives a number of days for each DD. So the higher those numbers, the later DD's are solved. This process measure is the average number of days for all DD's. It improves neither learning (■), nor cycle time (■).

DD Speed: the average CT of the DD's on a product, calculating the time a DD takes from start to finish, from occurrence to containment. Illustration 15 shows that DD Speed causes longer cycle times (■), but improves learning (■).

Project Experience: the experience, approximated by the number of products created.

5.1 LATENESS AND CONCURRENCY HAMPER LEARNING

First, we look at DD Lateness and DD Concurrency. These are useful in simulating the conditions under which DD's are solved. A high DD Lateness means that there are many issues at the backend of the cycle time, close to shipment of the product. A high DD Concurrency means issues occur in many products within the same product family.

ILLUSTRATION 17
Cycle time without DD Speed (–) or without DD Priority (–).

DD Concurrency has two effects. On the one hand, it increases speed. The explanation for this is that DD Concurrency increases scale effects. On the other hand, it hampers learning, because the complexity hinders the possibilities to find a root cause, and to learn from the effects of a solution.

Learning is also hampered by DD Lateness, because the time to learn from DD's close to the deadline is limited. At that time, the focus is on finishing the product. Moreover, a DD might cause a lot of rework, which would negatively impact CT.

Illustration 16 uses a scenario analysis, based on actual data, to show what would happen to the cycle time (on the vertical axis) without the DD Concurrency effect (—), and without the DD Lateness effect (—). The top curve (—) shows the actual cycle time.

Scenario: no DD Concurrency, or no DD Lateness
The (—) and (—) curves show that learning is hampered by DD Concurrency and DD Lateness. Further statistical analysis shows that CT is reduced by DD Concurrency.

5.2 PRIORITY AND SPEED ENABLE LEARNING

Next, we examine the other two process measures, DD Priority and DD Speed. Illustration 17 uses a scenario analysis, based on actual data, to show what would happen to the cycle time (on the vertical

axis) without the DD Priority effect (—), and without the DD Speed effect (—) (different only at the start). The bottom curve (—) shows the actual cycle time.

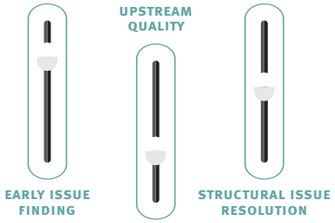
Scenario: no DD Priority, or no DD Speed
These curves show that DD Priority and DD Speed enable learning, but at the expense of some cycle time.

We postulate that operator self-reliance is an important factor: he or she should get the time to learn how to solve particular issues. Building on that, companies could even consider to offer discounts to customers to allow a longer solution time: customers get compensation for a longer downtime, and the ROI for the company is a structurally reduced cycle time.

In this section we analyzed the process measures related to design debugging. The next part of this chapter concentrates on the related management principles.

5.3 DESIGN DEBUGGING MANAGEMENT PRINCIPLES

The next paragraphs focus on three of the management principles (the slides of the mixing console), which we introduced in Illustration 3 of paragraph 2.6.



5.3.1 EARLY ISSUE FINDING MAKES FIXING EASIER

Our first DD management principle, Early Issue Finding, relates to 'frontloading' of DI's. The expert insight *The issue-finding champion* on page 48 shows that the sooner DD's are discovered, the easier it becomes to implement DI's, and also the fewer products that DD will occur on. Main goals are:

- Increase the number of issues found in the beginning of the new product introduction phase, by motivating and rewarding people to find issues, and to reduce the normal focus on output.
- Acknowledge cross-functional expertise. Prior work experience in other sectors can improve issue finding. For example, experience in the early stages of servicing products may inspire signaling an entirely different kind of DI's: those that are 'killing' in the field.
- Acknowledge issue finding expertise, and prevent the 'blame game'. Because to find issues, it is important to encourage an environment that focuses on solving them, instead of pointing fingers at those who made the mistake. To make a cable connection fool proof, you have to identify the problem, and that requires rewarding instead of blaming.

EXPERT INSIGHT

THE ISSUE-FINDING CHAMPION

"We were in the 'discovery phase' of our learning curve, trying to speed up design stability by solving design issues. While browsing through dozens of graphs we found one with the number of issues per employee. It showed that one manufacturing engineer reported 16% of the more than 10.000 issues! We interviewed him, and learned that he worked meticulously, and had factory, design and field experience. So he knew what was unacceptable for those worlds, which is different from a proto product where everything may go wrong. The interesting thing was that he was supposed to be output driven, so by reporting so many issues he went in fact against his manager. He told us that in the beginning, project leaders always found him a pain in the ass for reporting everything he found, but after a while they kept coming back. They discovered that if they processed his lists, it would be a smooth ride after that. He added true value. And he also passed on his knowledge: when we showed him the graph of top issue reporters he told us that numbers 2, 3 and 4 were his protegees!

These kind of engineers are extremely important. His case lead us to Early Issue Finding: ask a small group of people to find 80% of all design issues in, say, the first five products."

The best way to achieve Early Issue Finding is to select a few 'issue finding champions', and to give them budget, time and one or more products. Their only task is to identify as many potential design issues as possible, for example by using accelerated lifetime tests on those products. Product output is irrelevant, their only KPI is the number of issues.

5.3.2 UPSTREAM QUALITY: THOROUGH TESTING PAYS OFF

The second DD management principle relates to DD Lateness. As we saw in paragraph 5.3.1, it is vital to solve DD's as soon as possible in the production process.

Main goals are:

- Move issue solving responsibility upstream, to the module or sub-assembly level
- Increase solving speed, to catch mistakes early, when they are cheaper to fix
- Fast feedback from product test to module test
- Improve early testing with high test coverage at module level
- Set clear targets and priorities for structural improvements
- Reduce the 'redo factor', and to prevent swaps
- Continuously improve material quality

So Upstream Quality focuses on finding faulty parts and module defects as soon as possible, because it is much more expensive to swap a part or a module later in the production process. It takes more time to locate, reach and replace it, and the product also represents a higher value. That's why it pays off to take more time to test parts and modules more thoroughly, before they are released to production.

5.3.3 STRUCTURAL ISSUE RESOLUTION: CATEGORIZE AND ALLOCATE

The third principle, Structural Issue Resolution (SIR), is more about the issue process: from spottting the DD, via specifying the problem, to finding the correct solution provider.

Main goals are:

- Establish a clear connection between the problem finder and the solution provider (see the expert insight *Knowing who knows what* on page 50)
- Embed strong cross-sectoral issue ownership in both manufacturing and development
- Define coordination mechanisms to define, track and allocate responsibilities
- Acknowledge the importance of solving problems, compared with other responsibilities
- Create meeting structures, providing platforms for discussion

The art of SIR is to quickly categorize issues, for example according to factory execution, material and technical issues. That makes it easier to submit the issue to the correct solution provider. But the more perspectives, the better. Categorization according to production process is useful too, for example building and testing a particular module, because the same issue may occur in both processes. That means solving two issues at once, and improving CT for that module.

SIR is one of the most important drivers of the learning curve, because experience is accumulated in a feedback loop while solving issues (Illustration 18).

This wealth of experience that is built up, helps various departments to improve either production or design related activities. In turn, this inevitably leads to shorter B times, and that is our main goal, as stated in the summary of chapter 2.

In a perfect world, cycle time during new product development would only consist of A time, because production would occur without either planned downtime (C time) or unplanned interruptions (B time). DD's are responsible for a sizeable percentage of B time.

Research on several years of DD company data on a particular product family showed that 35% of DD's inhibited learning[13]. These DD's made it impossible to continue production, because the issue was too complex, and could not be solved quickly. In other words: if DD's are solved fast, B time will drop, and more time becomes available for production. The expert insight *Knowing who knows what* on page 50 shows how important networking skills are to speed up the solving process[14].

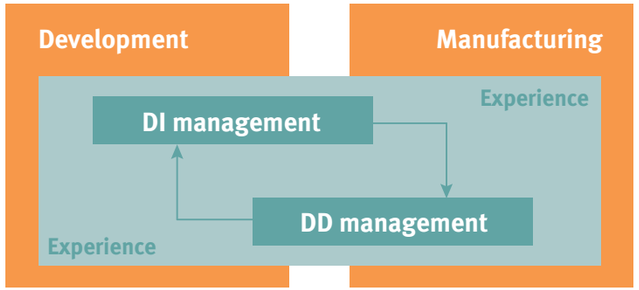


ILLUSTRATION 18
Experience is accumulated while resolving issues.

EXPERT INSIGHT

SAME PERSON TO DO MODULE AND FINAL QUALIFICATION

"Our supplier contracts are based on QLTC: Quality, Lead time, Technology and Cost price. But our buyers are driven mostly by lead time and price. That's how they reach their targets. Sure, they acknowledge the importance of quality, but their acceptance level is far lower than mine, in the factory. They find 50% dead parts completely acceptable, and it is extremely difficult to convince them that this is absolutely unacceptable for the factory. Even if a mere 70% of our parts is OK, we have to extract a hundred or more dead parts from a product to get it working.

It took me considerable time to realize that better end qualification of parts may be a larger investment than the additional cycle time to extract those hundred parts. That's why dead parts should be identified not at the end, but as soon as possible. It is cheaper, and most importantly, easier to find. Strict quality gates at module level will slightly increase production costs, but it will save a lot of time in the rest of the process. And reversely, if problems surface during end qualification, they must be transferred back to module level. And we think this should be done by the same person: people performing the module level qualification also do the final qualification. That way no knowledge transfer is required: learning is immediate."

[13] *van Olffen, 2015*
[14] *Argote, 2013*

EXPERT INSIGHT

KNOWING WHO KNOWS WHAT

"We have hundreds of people working in the factory, and hundreds of developers. So when one of those hundreds finds an issue, which one of those hundreds should he contact? Stacking your issue on the big pile doesn't help, developers constantly struggle with their workload. Linking the problem finder and the solution provider is a true profession. It speeds up the solution speed tremendously. Even if you pin down the issue to a particular module or part, there are still dozens of developers to contact, and each one you call will point to the next. The more time you lose talking about who should solve the issue, the less solutions you get."

Balancing concurrency with solving power

Without sufficient solving power (people) it is not possible to implement SIR and Early Issue Finding. Illustration 19 shows how crucial the first part of the pilot phase is.

- The top graph shows the number of DD's, with a clear spike at the start.
- In middle graph, the average DD Concurrency (at product level) rapidly increases during that same period.
- The bottom graph also shows a spike for average DD Speed at the start, and it takes considerable time to get that down.

In other words, a 'lake' of DD's is built up rapidly in the first part of the pilot phase. The best way to deal with this is by assigning solving power to DD Concurrency. Because if those issues are solved first, that will have a positive effect on multiple products at once.

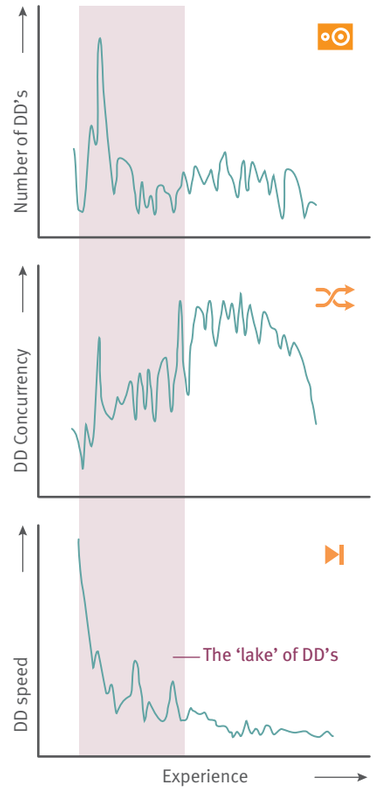


ILLUSTRATION 19

DD's (top), DD Concurrency (middle) and DD Speed (bottom) during pilot phase

PROVIDE TIME FOR DESIGN DEBUGGING

The process of design debugging is a double-edged sword: one the one hand, DD's emerge from product and process failures, while on the other hand they contribute to learning. Performance is enhanced when managers are able to harvest the experience from design debugging. This is managed in four ways.

First of all, our advice is to limit DD Lateness, because it hampers learning. The related enabling management principle is Upstream Quality: solve issues as soon as possible in the production process.

Secondly, we suggest to optimize DD Concurrency by using Early Issue Finding. This results in a big pile of early and concurrent DD's that must be solved as soon as possible. Best practice is to match the solving capacity with the number of issues on hand.

Thirdly, prioritize vital DD's. This will draw attention by solution providers and enable allocation of sufficient solving capacity. The learning that emerges from high priority issues is driven by the Structural Issue Resolution management principle, with guidelines to find structural solutions.

Finally, to enable the three aforementioned suggestions, managers should provide enough solving time and self-reliance for individuals to find solutions. It is therefore important to optimize issue finding and structural issue resolution, and to remove wasted time. This gives the required time for individuals that add value in design debugging.

Learning is phase dependent

The previous two chapters showed the learning aspects of design iterations and design debugging. Both these two processes strongly depend on the type of product development phase. This chapter shows that, indirectly, the way management has to cope with (and learn from) DI's and DD's depends on the product development phase.

6.1 FROM ORGANIZATIONAL LEARNING TO STEADY PRODUCTION

Let's first explain the fundamental difference. This was introduced in the first chapter (Shifting management principles), where we compared proto, pilot and volume phases with a tryout, a premiere and a tour. Another way of looking at this shift, is if improvising jazz musicians (proto) would be gradually required to play from sheet music (volume). These kinds of shifts are important to fully understand, because it impacts the management principles we laid out in the sound system framework.

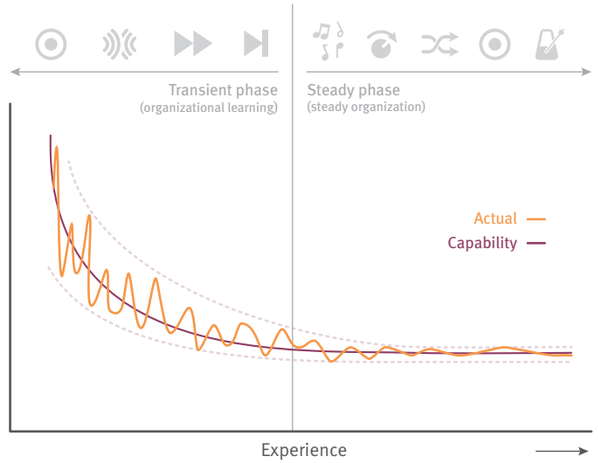


ILLUSTRATION 20
From organizational learning to steady phase.

Following the shift analogies, Illustration 20 shows the two phases we distinguish: the transient and the steady phase[15]. By dividing the environment in these two phases, with threshold levels, we can decrease the variability in the individual tasks over the number of products.

- The transient phase represents the time in which variability is high, for example because of sequence changes in the test process. Products and processes are not standardized yet, so the organization is learning. As a result, the actual cycle times deviate considerably from the target capability (for example, when B time is 3 to 20 times A time).

  This phase matches the proto and pilot phase of production, in which the first products are built.

- The steady phase represents the time in which variability is low. Eventually, products and processes become standardized, and the organization is done learning. As a result, the actual cycle times deviate little from the target capability (for example, when B time is 2 to 3 times A time).

  This phase matches the volume phase of production, in which multiple products are built.

The fundamental difference between phases thus directly influences the management of learning and cycle time in smart industries. The transient phase is enabled by management principles and mixing console settings, that drive improvisation and facilitate managing B time. The steady phase, on the other hand, requires stability settings that focus on managing A time. We explain the settings per phase in paragraph 6.5.

In an ideal situation these settings are sequential: first improvisation in the transient phase, then orchestration in the steady phase. However, in reality improvisational and repetitive activities are executed in parallel. In paragraphs 6.2, 6.3 and 6.4 we underline the challenges of concurrency of processes, and postponement of issue resolution. In paragraphs 6.6, 6.7 and 6.8 we highlight two additional principles that are required to manage concurrency and iterative design processes: competence management and interface management. Finally we will summarize the mixing console settings per phase.

6.2 REDUCE DEVELOPMENT AND MANUFACTURING CONCURRENCY

One aspect becomes more and more relevant for our phase distinction. Due to time-to-market demands, development and manufacturing processes become increasingly intertwined. Illustration 21 shows the number of development (—) versus manufacturing (—) engineers involved over time. Both the number of people and the time spent have been standardized to allow comparison.

The top three graphs are products from the same product family (B). The first two, B1 and its successor B2, show a clear handover from development to manufacturing. The next successor, B3, required

[15] *van der Kaaden, 2014*

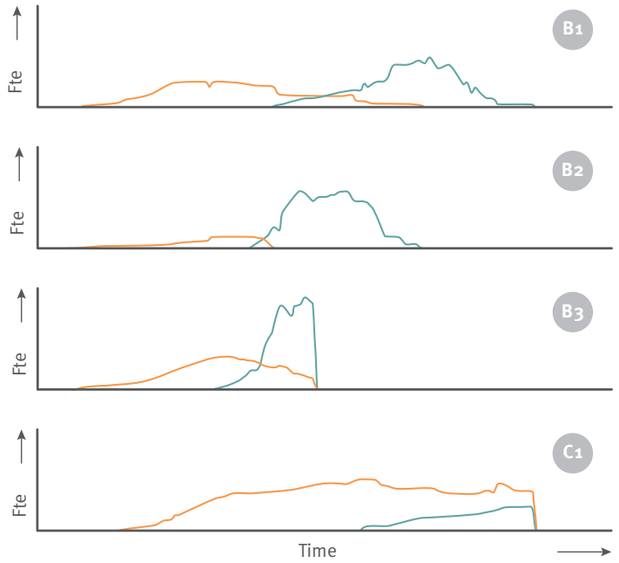


ILLUSTRATION 21
Development (–) versus manufacturing engineers (–) involved over time.

more development and considerably more manufacturing, but over a shorter period of time. Also, a larger overlap is visible: development continued during the manufacturing period.

The problem becomes clearly visible in the next product family (C). Development keeps on working, changing and updating the product, while manufacturing has already started. In other words: concurrency in development and manufacturing is increasing, but also the overall timeline is decreasing. This makes it much harder to make the phase distinction we introduced in paragraph 6.1. The steady phase is disappearing!

6.3 DO NOT POSTPONE ISSUE RESOLUTION TO VOLUME PHASE

The same problem occurs in issue resolution: design debugs which have been contained, but still require a structural solution, are postponed to the volume phase. Illustration 22 shows the number of open design iterations (—) versus design debugs (—) over time. Again, all values have been standardized to allow comparison.

The graphs are based on the same products as in Illustration 21. For the first one, B1, all issues are resolved during the transient phase. For its successor B2, all DD's are solved, but some DI's still remain open when the steady phase starts. For the next successor, B3, DD's

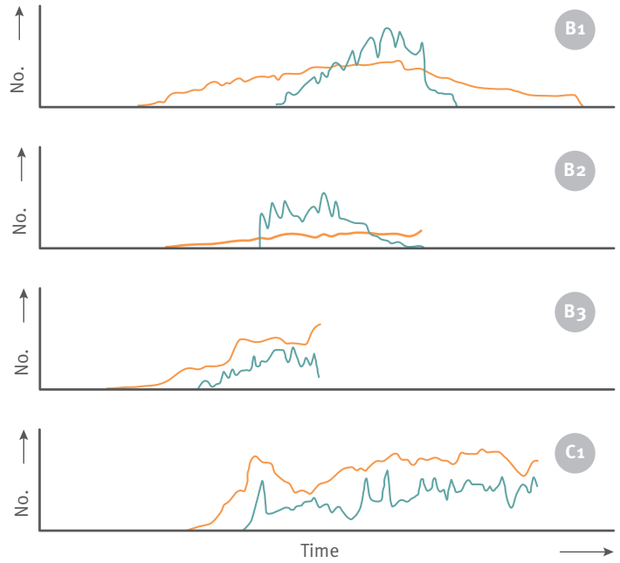


ILLUSTRATION 22
Open DI's (–) versus DD's (–) over time.

as well as DI's remain open. Also the numbers of both DD's and DI's are larger.

The problem becomes clearly visible in the next product family (C). Although an initial peak of DD's and DI's is partially resolved, after that the number of issues keeps rising, and a very large number is left open for the steady phase. This hampers many management principles we discussed earlier:

- Early Issue Finding is pointless: why bother finding issues if they are not resolved?
- Structural Issue Resolution is not possible.
- DI Process Management is useless: without solving DI's it is not possible to achieve a stable design.

So, also regarding issue resolution it is much harder to make the phase distinction we introduced in paragraph 6.1.

6.4 PARALLEL DESIGN ITERATIONS AND DEBUGGING AFFECTS LEARNING CURVE

The relation between the number of DI's and DD's on the one hand, and learning on the other hand, can be summarized in a table.

Table 1 shows that learning policies vary with the ratio between DI's and DD's. In the transient phase, with a high number of DI's, the cycle

Phase	DI's	DD's	Learning through	Effect
Transient	High	Low	Technology	Steep learning curve
		High	Design	Reduced learning curve due to complexity
Steady	Low	Low	Production process debugging	Learning approaches a maximum
		High	Continuous improvement Competence development	Incremental learning effects

TABLE 1
The relation between phase, DI's and DD's.

time is under pressure but learning occurs by solving DI's and DD's. As the number of DI's and DD's drop, focus moves to autonomous learning and self-reliance. And as production reaches stability, mainly DD's cause variance in CT. Learning effects are limited, and there is possibly even de-learning, as experienced operators move to the next product family.

6.5 PHASE DEPENDENT LEARNING: FROM TECHNOLOGY TO COMPETENCE

We can now expand the phase distinction of paragraph 6.1 with management principles that apply to the transient and the steady phase.

Table 2 provides an overview of the characteristics of the transient and steady phase in managing cycle time. It is apparent from this table that the speed tactics greatly differ between phases. Where managers in the steady phase can rely on a certain environment and planning, the transient phase needs to rely on iteration, testing and experimentation as a way to coordinate activity.

The phase distinction of Table 2 also applies to learning. Table 3 shows that first and second order learning are split over the phases, with their own unique characteristics.

Transforming experience into reduced cycle time is key. Learning is required to do that, and several studies have shown that the success of companies in doing so, depends on their optimization of first and second order learning.

This optimization is important, because each type can inhibit or enable the other. For example, too many iterations in the transient phase can inhibit first order learning, because there is little to no repetition. Reversely, an unstable situation in the steady phase reduces the possibility to learn from production.

Characteristic	Transient phase	Steady phase
Phase	Early phases: proto, pilot	Late phase: volume
Tactic	Experimental	Compression
Context	Uncertain	Certain
New product development activities	Unpredictable path with uncertain technologies and markets	Predictable, well defined steps, routines
New product development speed strategies	Quickly build intuition and maintaining options	Rationalize and squeeze the process steps
Speed tactics	Iterations Extensive testing Frequent milestones Powerful leaders Learning effect from issue resolution	Planning Concurrency Cross-functional teams Reward for meeting schedule Use CAD to reduce step time

TABLE 2
Management principles in transient and steady phase.

Please note that a hierarchy exists in issue resolution: before process issues can be resolved, first the number of technological and/or design issues should be reduced. These kinds of issues typically surface in the transient phase, which leads to second order learning.

6.6 MANY DIFFERENT COMPLEX TASKS ARE EASILY FORGOTTEN

Task complexity is the most important factor in learning or forgetting. If a task requires substantial cognitive capabilities from an operator, and requires training of specific skills, than it can be considered complex. But as more tasks are assigned to operators, it becomes harder to regularly repeat those skills, and to perform all of them efficiently.

That is because task frequency is another important factor. Operators also learn and forget based on how often they perform a task. If they do certain tasks more often, the task speed increases. But highly complex tasks are easily forgotten, because those must be done frequently to keep the skills at a high standard.

In general, smart industries have to cope with difficult and highly complex tasks, a high number of different tasks per product, and a

Characteristic	Transient phase	Steady phase
Learning	*Second order learning:* reflect on how the work is done, and learn from that	*First order learning:* do the work, and learn from that
Experience source	Evolving insights and heuristics	Repetition, association building
Triggers	Iterations in procedures, design and technologies	Re-applying what worked in the past
Employees involved	Higher (management) levels	Lower levels
Improves	Indirect behavior via structures and processes	Direct behavior and performance
Issue types	New	Recurring, similar
Management principles	Early issue finding Structural Issue Resolution: *enabling* Competence Management: *improvise* DD Process Management	Upstream Quality Structural Issue Resolution: *preventive* Competence Management: *record* Delta Management

TABLE 3
First and second order learning in transient and steady phase.

relatively low task frequency. Those three factors make tasks both difficult to learn and easy to forget. This leads many researches to conclude that it is better to keep operators focused on their dedicated tasks whenever possible.

6.7 COMPETENCE MANAGEMENT IMPROVES PERFORMANCE

COMPETENCE MANAGEMENT

If a company wants to reduce cycle time while increasing production, but also wants to keep the capacity and work pressure of the first line support department at a constant level, then other departments will have to take over the tasks of this support line. In other words, operators become more self-reliant. The best way to learn is from your own mistakes. While someone solves issues, he builds his or her own routines.

Many studies have shown that self-reliance has a positive performance effect: at individual level, at work unit level and even at company level. At each level, different preconditions determine if self-reliance does have the desired positive effect:

EXPERT INSIGHT

COMMITMENT MANAGEMENT

"Our factory works in shifts, operators work four of the twelve shifts per week. They work on the latest technology, a product worth millions of euros. Yet it was a challenge to engage them, to make them proud of what they are building, because they did not feel 'in control'. Our products are currently so complex that it takes around 15 experienced operators to cover all aspects. And I cannot assign them all to one product! There is not a single operator who can solve all problems. So when operators ran into a problem a year ago, they had to ask for help in 70% of the cases. Then it depends on who turns up and how long it takes before they can continue. That's why we started to focus more on self-reliance, translating knowledge into skills by documenting how known problems can be solved. These documents are also known as PCS documentation (Problem, Cause, Solution). Now, just one year later, operators ask for help 50% of the time! The main advantage is not the time we save, it is a psychological benefit. It means operators feel in control when they go home after a shift. They become proud of what they are building. And that means happy operators, less turnover, and better knowledge retention.

That is also why I don't believe in assigning operators to multiple production cells. It may seem efficient to distribute the available capacity over the products, but it does not pay off in the long run. They don't become familiar with the history of the product, they don't know if they will be working on that same product next week, so they don't get the chance to become proud, to ship 'their product' to the customer."

- At company level, the preconditions are job enrichment and skill enhancement
- At work unit and individual level, the right empowerment climate is required
- At individual level, psychological empowerment also improves job satisfaction

The above is often referred to as Competence Management.

Main goals are:

- Increase a sense of ownership and pride
- Encourage self-reliance and trust
- Transactional memory building
- Competence based training and cross-training
- Network building

6.8 INTERFACE MANAGEMENT IMPROVES ALIGNMENT

Design iterations and design debugs are performed by people scattered across the organization. As mentioned earlier, Structural Issue Resolution (see 5.3.3) requires them to connect to the problem owner and to the solution provider. Moreover, the previous chapters show the importance to align goals with respect to cycle time and learning curve management. For example, designers and engineers need feedback from operators and testers about the product design, while operators might want more capacity utilization to improve output. If managers are not able to align these objectives, the company could

INTERFACE MANAGEMENT

BACKGROUND INFORMATION

PARETO PRINCIPLE

The Pareto principle (also known as the 80–20 rule) states that, for many events, roughly 80% of the effects come from 20% of the causes. It was named after Italian economist Vilfredo Pareto, who showed that approximately 80% of the land in Italy was owned by 20% of the population. In our case, the principle applies to the observation that 80% of problems can be attributed to 20% of causes.

suffer from suboptimal decisions. Actually, an increased utilization might result in some savings, while it may cause dozens of designers and engineers to wait for essential feedback on the product design. Because goals and priorities are phase dependent, interface management is required for better alignment and improved agility.

Main goals are:

- Handshake at issue level between factory and product development stakeholders, about goals and priorities
- Consolidation and reporting on overall performance and learning to factory and product development stakeholders
- Clear definition of problem owners and solution providers
- Pareto and impact/effort analysis of issues to align priorities
- Commitment to solve issues until target per solution provider

6.9 HOW TO BECOME THE PERFORMER IN YOUR OWN PIECE OF MUSIC?

In this book we provide a framework that enables managers to optimize their production speed with learning. It responds to the challenges of a growing number of companies that operate as smart industries. Due to their distinguishing characteristics they need to balance learning and speed. The important underlying management principles, and process measures related to learning and speed, are now unraveled in a framework. The arguments in this chapter underline the importance of phase differentiation in managing the settings of the cycle time mixing console. The question now becomes: how should we use this framework? Here is some advice:

1. Managers are required to recognize the importance of learning curve management. So ask yourself the question: why should I focus on learning curve management over production output? The most simple answer is that learning in the transient phase of new product development is mainly driven by experience in design iteration and debugging, in contrast with experience in production output. If you are convinced of its importance, you can get in control with the mixing console, and manage progress in B-time.
2. The framework distinguishes between process and outcome measures, the mixing console, and the different cycle time types. The measures are quintessential for the settings mixing console. Managers that are responsible for learning curve and cycle time management are advised to operationalize and record these measures. Based on these performance indicators the mixing settings can be adjusted. In an ideal scenario this data is measured and updated in real-time, which propels the responsiveness of management and improves the quality of remedial actions.
3. With respect to the measures related to design iteration, managers need to apply the principles that enable quick and early implementation of design iterations. Design iterations propel progress, but cost time. The best practice is involving many disciplines, and use Delta Management for effective implementation.
4. Measures related to design debugging need to be established. A manager needs to provide time for design debugging and thus be aware of DD Priority, DD Speed, DD Lateness and DD Concurrency. These measures approximate the context where learning takes place. Managers for instance need to balance DD Concurrency and DD Speed with solving power. Among the most important best practices are Upstream Quality, Early Issue Finding, and Structural Issue Resolution.
5. Management of learning and cycle time is a delicate process where many people from different backgrounds and competencies are required to work together. In this book we touched upon several areas that require management attention. Aspects such as 'self-reliance', 'higher commitment', and 'problem solving capabilities', and settings that stimulate improvisation and prevent 'blame games' in making mistakes, are of utmost importance. Moreover, people's skills in 'knowing who knows what' and 'issue finding', and their ability in managing interfaces need to be acknowledged.
6. Managers need to be aware of the product development phase and the related settings: learning is phase dependent. The table on the next page summarizes the settings per phase.

MIXING CONSOLE SETTINGS PER PHASE

Because of the differences in phase properties listed in paragraph 6.1, also management principles (the slides of the mixing console) strongly depend on the phase of the product. The table below summarizes optimal settings for each phase.

Management principle	Transient phase	Steady phase	Remark
Early Issue Finding	Encourage	Too late	Select 'issue finding champions', to identify as many potential design issues as possible. See also paragraph 6.3
Upstream Quality	Hard	OK	Find faulty parts and module defects as soon as possible.
Structural Issue Resolution	Mainly as enabling process, take ample time	Mainly preventive, generic	In practice, steady phase is often reactive instead of preventive.
Competence Management	Build on trust, improvisation, knowing who to ask questions	Record skills and knowledge in instructions	At first, there is no process and there are limited procedures. You deploy creative, skilled operators who can write procedures themselves if necessary. In the steady phase you need the opposite: operators who follow procedures thoroughly.
Interface Management	Invest in networking skills	Networking skills less important	Find the correct resources: interface between 'functional clusters' of the product, and development engineers who can provide solutions.
DI Process Management	Desirable: encourage DI's	Undesirable: prevent DI's. Design freeze	Without solving DI's it is not possible to achieve a stable design. This requires solving thousands of DI's, and as soon as possible in the process.
Delta Management	Less relevant, maximum flexibility	Assign people to deltas. Design freeze	At the end of the process to focus on predictability. It is necessary to keep changes limited to specific parts of the product that require improvement, and to steer clear of other areas, functioning at an acceptable level.

References

Alblas, A., & Langerak, F. (2015). The impact of design debugging on new product development speed: the significance of improvisational and trial-and-error learning. In S. Narayanan & R. Bhageria (Eds.), *POMS 26th Annual Conference, May 8 - May 11, 2015, Washington D.C., USA*: Online Proceedings S.l.: Production and Operations Management Society (POMS).

Argote, L. (2013). *Organizational Learning - Creating, Retaining and Transferring Knowledge*. Boston, MA: Springer US.

Boehm, B. W. (1981). *Software engineering economics*. Prentice-hall Englewood Cliffs (NJ).

Cankurtaran, P., Langerak, F., & Griffin, A. (2013). Consequences of New Product Development Speed: A Meta-Analysis. *Journal of Product Innovation Management*, 30(3), 465–486.

de Kadt, N., Peeters, T. J. G., Langerak, F., & Alblas, A.A. (2015). *Accelerating the Learning Curve at ASML.* Master of Science in Innovation Management. Eindhoven University of Technology, Eindhoven.

Huizinga, G., Walison, P., Bouws, T., Kamer, F., van der Beek, H., Tops, P., and others (2014). S*mart Industry - Dutch industry fit for the future* (Project team Smart Industry).

Jarratt, T. A. W., Eckert, C. M., Caldwell, N. H. M., & Clarkson, P. J. (2010). Engineering change: an overview and perspective on the literature. *Research in Engineering Design*, 22(2), 103–124.

Loch, C. H., & Terwiesch, C. (2005). Rush and Be Wrong or Wait and Be Late? A Model of Information in Collaborative Processes. *Production and Operations Management*, 14(3), 331–343.

Terwiesch, C., Loch, C. H., & De Meyer, A. (2002). Exchanging preliminary information in concurrent engineering: Alternative coordination strategies. *Organization Science*, 13(4), 402–419.

van der Kaaden, T. G., Arts, J. J., & van Ooijen, H. P. G. (2014). *Allocation of a Cross-trained Work-force in an Environment Subject to Organizational Learning.* Master of Science in Innovation Management, Eindhoven University of Technology, Eindhoven.

van Olffen, J. B. J., Langerak, F., & Alblas, A. A. (2015). *Optimization of issue-resolution for cycle time : a case study at ASML.* Master of Science in Innovation Management, Eindhoven University of Technology, Eindhoven.

Veldman, J., & Alblas, A. A. (2012). Managing design variety, process variety and engineering change : a case study of two capital good firms. *Research in Engineering Design*, 23(4), 269–290.

Voogd, M. A. S., Borgh, W. van der, & Alblas, A. A. (2014). *De invloed van process mining op verstoringen.* BSc. Thesis Industrial Engineering. Eindhoven University of Technology.

Insight
the evolving insights of smart industry development and production

Insight Driven
evolving insights of design iteration and debugging drive the learning curve

Insight Driven Design
the development process based on the principle that not everything can be designed up-front: we have to learn, iterate and debug

If you are a development or production manager in a smart industry company, then this book is a must-read. Because a growing number of smart industry companies face the challenge of balancing learning and speed. Over the past decades cycle time has become a prominent driver for managing development and production. But managing cycle time becomes more and more complex due to evolving insights in design. We show you how to focus on learning curve management, and not just on production output, to keep your non-planned cycle time in check.

After two years of research on ten years of company data we can finally unveil the most important factors of non-planned cycle time in smart industries. This book presents the scientific results in a non-specialist way, and provides tangible advice for managers in smart industries. Some examples are:

- in the early phases, use Delta Management, and implement design iterations quickly, by multidisciplinary teams
- provide time for design debugging, but match the debugging solving power, and harvest the resulting experience
- do not postpone issue resolution to the volume phase

The findings have been condensed into one framework, containing:

- **nine 'process measures'**
  to understand and predict learning curve and cycle time performance, captured in a quadrant model
- **seven ‘management principles’**
  to balance learning and speed, by managing the process measures depending on the phase
- **four 'outcome measures'**
  against which the process measures are scored

To provide a better overview, the above framework is presented in the form of a sound system. Managers can use the mixing console (the management principles) to filter the properties of the input channels (process measures), to impact the output channels (outcome measures).

9 789038 642017 >