DIADROM

DIAGNOSTICS

Second, expanded edition

From vehicles to defence

Developing, producing and maintaining software-based capital products with high availability

Nothing works until everything works

#DiadromInside

Fredrik Ljungberg · Henrik Fagrell · Viktor Eliasson

FRONT MATTER

Title and author page

Diagnostics

Second, expanded edition

Developing, producing and maintaining flexible software-based capital products with high availability, from vehicles to defence.

Fredrik Ljungberg (Ph.D.), Henrik Fagrell (Ph.D.) and Viktor Eliasson (CEO)

diadrom.se · next.diadrom.se

Contact page

Contact

Diadrom Holding AB (publ), Gothenburg, Sweden

diadrom.se · next.diadrom.se · info@diadrom.se

Viktor Eliasson, CEO, viktor.eliasson@diadrom.se, +46 (0) 733 31 11 15

Switchboard +46 (0) 31 774 11 00

Diadrom is listed on NGM (Nordic Growth Market) under the ticker DIAH.

© Diadrom Holding AB (publ), 2026. Second, expanded edition.

(Contact details, copyright year and edition statement to be confirmed against the latest information before print.)

Foreword (original)

Our greatest asset when we started Diadrom at the end of 1999 was our knowledge of mobile applications. For several years we had worked on applied research on the theme of mobility. Our object of study, as one says in research, was the new nomads, that is, people who work and live in a mobile environment. In English we often used the term the newmads, because we found the new nomads awkward in both writing and speech. We therefore named the company we founded Newmad Technologies, technologies for the new nomads.

In the early 2000s we had the privilege of developing a mobile application for diagnostics. We simply applied our knowledge of mobile applications to the diagnostics domain. The project went well, and we soon received new enquiries within diagnostics. The ball was rolling. After a few years we had built up a body of knowledge and a component library within diagnostics. We had grown from researchers in mobile applications into experts in diagnostics, and choosing a name that alluded to diagnostics was an obvious step.

The choice fell on Diadrom, which means a complete pendulum motion. At the time we often explained what we did by showing a picture of how data flows from the vehicle's embedded systems, via the diagnostic application to the manufacturer's central system and back, that is, a complete pendulum motion.

In recent years, diagnostics-related problems and needs once experienced only in the car industry have begun to appear in other industries that work with software-controlled products. We have broadened our industry focus from vehicles to other sectors, for example security, and begun to develop diagnostic products that we have had the privilege of selling to customers in, among others, the defence industry. We have also received ever more requests to give training, and therefore begun to document our knowledge, of which the text you are holding is one example. Our ambition with this text is to introduce diagnostics in general terms. It is a first version, which we intend to develop further. We hope you enjoy it, and do get in touch with questions and comments.

We would like to thank our colleagues at Diadrom and our customers. A special thanks to Anders Dyhre, Oscar Lund, Carl Johan Andersson, Per Dahlberg and Patrick Jansson.

Fredrik Ljungberg and Henrik Fagrell

Foreword to the second, expanded edition

When Fredrik and Henrik wrote the first edition of this book in the early 2010s, their ambition was simple: to introduce diagnostics in plain terms, for anyone who wanted to grasp the big picture without drowning in detail. They succeeded, and the book has served ever since as a shared map for customers, colleagues and the simply curious.

They also saw something that was, at the time, only a clear trend. Diagnostics, they wrote, was spreading from passenger cars and commercial vehicles to almost every kind of capital product, from mining equipment to locking systems and space technology. That trend has not merely held. It has accelerated, and it has taken us to places we could then only guess at.

Since then, the car has become a computer on wheels. A modern product is governed less and less by mechanics and more and more by software, often connected and updated remotely. That opens enormous opportunities, but it also moves diagnostics out of the workshop and into a continuous stream of data. With connectivity come new demands: on cybersecurity, on traceability, on who owns the vehicle's data. What was once a cable between a mechanic and an engine is today a chain of

systems in which everything has to hold together. And that is exactly where it gets hard, because a chain is never stronger than its weakest link. Nothing works until everything works.

At the same time we have stepped into a domain where those demands are pushed to their absolute limit: defence and security. Here availability comes before cost. Here there is not always a workshop around the corner or a daily delivery of spare parts. Here maintenance has to be possible with a limited inventory, independent of terrain, on systems that are sometimes expected to fend for themselves. It is the same diagnostic discipline as in the automotive industry, only under the hardest pressure imaginable. For us at Diadrom this is not a new world, but a logical continuation. We have delivered diagnostics to the defence industry for many years, and we have now placed that effort at the centre.

That journey is what this edition is about: how the same diagnostic discipline reaches from vehicles to defence, by way of data, AI and cybersecurity. Our conviction is that whoever masters diagnostics masters availability, and whoever masters availability earns trust. We call it #DiadromInside, because our work is rarely visible, yet it is often what makes everything else work.

We have kept the core of the book, because it ages well. The maturity model, the difference between a fault code and a symptom, the path from a small defect to a system failure, and the idea that you get only one chance to meet the customer. We have modernised what had dated, and we have laid the new chapters on top. The rest is as it was always meant to be: written to be understood, not to impress.

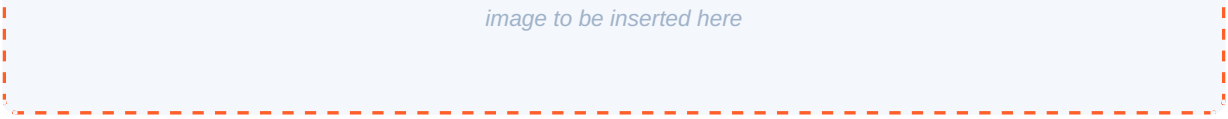
Enjoy, and do get in touch.

Viktor Eliasson CEO, Diadrom Holding AB (publ)

1 Introduction

FIGURE 1

The through-line: the same diagnostic discipline from vehicles to defence



The aim of this book is to introduce the field of diagnostics in general terms. We take a broad view of diagnostics, whose task we see as this: to develop, produce, maintain and upgrade software-based capital products in a way that enables high availability and flexibility. Our goal is to give the uninitiated an overview and an understanding of the broad strokes. That means we generalise and leave out details that a specialist in a particular sub-field might have wished to see. We write in a popular-science rather than an academic style, to make the text easy to read and accessible.

Many capital products are increasingly software-based. They are governed more and more by embedded hardware and software connected in networks, and are becoming ever less mechanical. The modern passenger car is a clear example. Where the first edition of this book spoke of 70 to 80 embedded computers, an advanced car today often has well over a hundred control units and software measured in hundreds of millions of lines of code. At the same time a counter-movement has begun. Instead of adding more and more control units, manufacturers now consolidate them into fewer and more powerful domain and zone controllers connected to central compute, something we return to. Products are moreover increasingly connected and software-defined, meaning they can be updated and improved remotely long after they have left the factory. Developing such a product is a highly complex technical and commercial undertaking. The difficulty of building large software systems is, incidentally, not new; it was described by Frederick Brooks as early as the 1970s (Brooks 1975).

Diagnostics plays a central role in developing systems and processes for software-controlled products, for example commercial vehicles such as trucks, buses and construction machinery, but also aircraft and other capital products with high technical content and a long service life. These products must be able to be loaded with software, updated with new versions and, when needed, upgraded with new functionality. Their characteristics must be adaptable to customers' changing needs, while high availability must be ensured.

A capital product plays a strategic role in the operation where it is used, and high availability is therefore a fundamental requirement. Maintenance must of course be carried out, but it should preferably be planned rather than unplanned. Because a modern capital product is a complex computer system, maintenance is no longer something a handy technician solves with a wrench. It

requires computer support, connectivity and specially trained personnel. High availability, and therefore little or no unplanned maintenance, lets the customer deliver efficiently to its own customers. Flexibility makes it possible to adapt functions as needs change. Diagnostics in the aftermarket is about maintenance and the upgrading of functionality, while diagnostics in product development is about building products that enable efficient maintenance and upgrading.

Because the automotive industry has long been at the forefront of the move toward software-based capital products, we draw many of our examples from there. But we see a clear trend that diagnostics is spreading from passenger cars and commercial vehicles to virtually every kind of capital product, from mining technology to locking systems and space technology. In this second edition we follow that thread all the way into the defence and security sector, where the same diagnostic discipline meets the hardest demands of all. The road there runs through data and AI, through cybersecurity, and through the question of who owns the data the products generate. We therefore treat diagnostics at a general level, and show how the same fundamental principles carry all the way, from vehicles to defence.

2 From mechanical to software-controlled capital products

2.1 Products and services

Companies produce and sell products and services. Examples of products are mobile phones, bandy sticks and cars, while a haircut, home cleaning and a restaurant meal are examples of services. A car you hire is, however, to be regarded as a service, while a self-service restaurant can in some respects be said to deliver products. In the latter case the customer orders the food, perhaps without interacting with anyone, and collects the meal, the product, to eat at home. The restaurant visit has then largely been turned into a product. Just as a classic product such as a car can be sold as a service, classic services can in principle be sold as products. It is the business model that decides.

Our focus here is on a kind of product called capital products. Such products can be sold as products, as services, or as a combination of both.

2.2 Capital products

FIGURE 2

Product types, from capital product to consumable

image to be inserted here

FIGURE 3

The value of a function over time

image to be inserted here

Capital product is a collective term for products developed to be used over a long time. That sets capital products apart from consumable products, which are used up quickly and perhaps only used once. An aircraft is an example of a capital product, a pen an example of a consumable. The aircraft is often bought to be used for many years, while the pen may be used for only a short period before it is

spent. Capital products are sometimes called capital-intensive products, and we treat the terms as synonyms. One can see capital-intensive and consumable as two ends of an axis, and the figure below shows where different products sit.

A capital product that is sold can have more or less service content. A used car no longer under warranty is a pure product, a car under warranty is largely a product but also partly a service, while a leased car hired from a firm is a service. Whatever business model applies, capital products must be maintained.

Capital products must be maintained because they are meant to be used over a long time. One can of course treat a new car as a consumable, but that is unusual. Without maintenance the product will deteriorate relatively quickly and eventually stop working.

Even when the product is maintained, the value of its functions falls over time. Eventually the functions become so poor that they break and need repair. With adequate maintenance an operation can considerably extend the time before a product breaks down. It can also better anticipate when problems will arise and plan maintenance accordingly. That reduces the risk of a product breaking down and requiring unplanned maintenance, for example a car that stops by the roadside and has to be recovered.

The figure below shows how the value of a function in a capital product declines over time. At first it is good, then it gradually deteriorates and finally breaks.

An advanced smartphone is not a pure capital product. The manufacturer is expected to release software updates, and the customer is willing to invest in some troubleshooting and repair if it breaks. A more capital-intensive product, such as a car, the owner is willing to repair for larger sums. It cost more to buy, its expected life is much longer, and the willingness to pay for maintenance is therefore much greater.

2.3 Maintenance

FIGURE 4

Maintenance restores the value of a function over time

image to be inserted here

The European Federation of National Maintenance Societies (EFNMS) defines maintenance as the full combination of technical, administrative and managerial actions taken across an item's life cycle to keep it in, or return it to, a condition in which it can perform the function required of it.

As the definition shows, maintenance starts from a product's functions. The aim is to retain and, when needed, restore the product's functions to their original condition. The assumption is that a product has a number of functions whose quality can be measured and which have a value. A helicopter has the functions lift, land, steer and so on. When the product is new the value of these functions is full, and one naturally wants to keep it there. That is not possible, however, because the functions wear and deteriorate as the product is used. One therefore wants to be able to restore the value of the functions.

The value of some functions is binary, either they work or they do not. Other functions can work more or less well. A brake disc gradually gets worse, and if it is not replaced it eventually becomes so poor that the function brake is considered broken. The value of the function is simply too low.

Even when a product is maintained, the quality of its functions often declines step by step. However effective the maintenance, the product will eventually run into problems and need repair. The time before a product breaks down can nonetheless be extended considerably with adequate maintenance. The figure below illustrates how the value of a function declines but is partly restored through maintenance, until the inevitable finally happens and the function breaks.

The most common type of maintenance has historically been corrective maintenance, that is, repairing a product after a fault has occurred. A problem with corrective maintenance is that it is unplanned. It is always negative when a product cannot be used, but the negative effects are worse when the stoppage is unplanned. If a lorry breaks down at the roadside the haulier cannot deliver the goods on time, and perhaps cannot collect the next load either. That hits the haulier's customer, who in turn may be unable to deliver to their own customer. The haulier may also have to bring in a replacement vehicle, which is often expensive.

Planned maintenance is negative in one respect, namely that the product cannot be used while it is being maintained. On the other hand it is a planned action and therefore need not disrupt commitments to customers. Planned maintenance is preventive, and the goal is to avoid unplanned stoppages and the need for corrective maintenance.

There are different types of maintenance developed in different industries. In general, preventive maintenance is planned, which distinguishes it from corrective maintenance, carried out as a consequence of the product breaking down. Preventive maintenance is done proactively, before a fault has occurred, to avoid unplanned stoppages. For example, one can carry out maintenance after a certain running time, such as changing the oil after 1,000 operating hours.

At a maintenance occasion, synonymous with a service occasion, a number of predefined operations are carried out. The time between maintenance occasions is usually called the interval, or service interval, and can be defined in, for example, operating hours or distance driven.

All products are used differently and therefore have unique maintenance needs. Two identical cars with different usage patterns have probably worn differently after, say, 20,000 kilometres. One may have been used by a taxi driver in dense city traffic, the other by a rural postal carrier on country roads. One way to handle these differences is simply to ask the product what maintenance it needs right now. The question is put by a software application that reads data out of the product and, based

on the values, suggests different maintenance actions. Maintenance is thus adapted to the product's current condition. That type of preventive maintenance is called condition-based maintenance, because it starts from the product's unique condition when planning. It is an idea we build on in the chapters on data, AI and predictive maintenance.

2.4 Functional and symbolic qualities

In the discussion of maintenance we have so far focused exclusively on a product's functions. But there are other kinds of quality than the purely functional, and although they are not central from a maintenance perspective, they are worth knowing about. Let us take an example.

Two cars with similar functions can be perceived very differently because their symbolic qualities differ. An Audi may signal a certain status and flair, while a Škoda signals thrift and rational thinking. One is a rational purchase, the other something you drive for the feel of it. Technically, however, cars developed within the same group have a great deal in common. So it is across large parts of the passenger-car industry, where several brands share a technical platform but where the individual brands target quite different audiences. The pattern has, if anything, grown stronger with electric cars, where a single platform can carry models from several brands in a group, for example Volkswagen, Škoda, Audi and Cupra, while each brand keeps its own profile.

As the reasoning shows, a product's functional quality is only one kind of quality among several in the customer's eyes. Functionality is of course very important, but you miss important dimensions of how customers relate to products if you look only at the functions. What a product radiates to the customer and those around them matters just as much. Let us call that the product's symbolic quality (Dahlbom and Mathiasen 1993).

The same reasoning is familiar from, for example, the clothing trade. Two pairs of jeans can have the same functionality and perhaps even last equally long. One pair might cost 2,000 kronor, the other 200. The expensive pair may signal that you are in the know, while the cheap one comes across as dull. The result can be that the expensive jeans sell very well and the cheap ones not at all, even though the products are functionally equivalent.

2.5 Competition and differentiation

Generally, a heavily contested product has a low degree of differentiation, that is, it is one of several similar products on the market. The customer therefore tends to focus on price, since that is essentially all that distinguishes the products. The product becomes a commodity, and you compete on low price. The more differentiated a product is, the more customers focus on the experience of owning it. They value the product being unique and genuine, while price matters less (Pine and Gilmore 1999).

In the automotive industry, each new vehicle generation contains ever more functionality, and the companies invest heavily in building brands. As a result, functions that were unique in one generation

often become commodity in the next. Companies must therefore constantly launch new functions, services and concepts in order to offer differentiated products focused on the experience, and avoid becoming a commodity that competes mainly on price.

As a consequence of ever tougher competition, companies must work on innovation in ever shorter development cycles. In the passenger-car industry, competition from Japanese manufacturers appeared in the 1980s. They were dismissed at first ("plastic cars", "poor quality"), but became serious challengers, above all to the American car industry, and in 2008 Toyota overtook General Motors as the world's largest carmaker. In the same way, many dismissed the Korean manufacturers when they appeared in the 1990s, and today Hyundai is one of the world's largest carmakers.

In the first edition of this book, written in the early 2010s, we noted that the Chinese manufacturers had begun to appear, and that nothing suggested they could not become serious challengers. That prediction has been borne out emphatically. Today the Chinese manufacturers are not knocking on the door, they are already inside it. BYD has grown into one of the world's largest carmakers, around sixth by volume in 2025, ahead of established names such as Honda and Ford. BYD has overtaken Tesla as the world's largest seller of electric cars, three Chinese groups (BYD, SAIC and Geely) are in the global top ten, and China has become the world's largest vehicle exporter. The lesson the original drew still holds, only sharper than then: dismiss a new challenger at your peril.

When a reporter once asked Volkswagen's then chairman Ferdinand Piëch how he viewed competition on the car market, he is said to have first been silent for 30 seconds, an eternity in an interview. When the journalist began to ask the next question, Piëch hissed: "Es ist Krieg. Es ist Krieg" (Åsberg et al. 2008). It says a great deal about competition in the car industry.

2.6 Innovation

While there are companies that have managed to break into heavily contested markets, it is often an advantage to choose markets with less competition, since there you have a greater chance of creating a unique offering. On heavily contested markets an offering tends to become one of several that resemble one another, and then price becomes the deciding factor. With differentiation you can create a more unique offering that customers are willing to pay more for. One point of so-called soft products, which we discuss later, is precisely the ability to increase the degree of differentiation.

The pace of innovation has been ramped up considerably in recent decades. A car model could once be sold for 15 years or more, whereas today it may have a life of five to seven years. Even though the life shortens, the number of technology shifts during the product's life increases. Take an example: the Volvo 240 was sold between 1974 and 1993 and rested throughout on a single product platform. It was a hardware product of mechanical components. A modern, software-based car, by contrast, changes components far more often, because it is so much easier to develop and distribute a new software component than to replace a physical one. To correct a fault in an old mechanical product, you had to develop, manufacture, stock and distribute physical parts to every corner of the world. To correct a software fault it is enough to fix the software and make it available. The difference is

enormous, and with connected and software-defined vehicles (section 3.9) it is taken a step further, as the software can be updated continuously over the air. The life shortens, but the number of technology shifts increases, and to succeed you must be able to manage the complexity of the existing product range while developing the new things the market demands.

In the context of innovation, people often wrongly believe that technical development is the great problem. More commonly, the problem is simply that you have not understood well enough what the customers want. It is the same kind of problem that sinks many systems-development projects. In the IT industry, the move has been from the waterfall model toward agile ways of working with short iterations and customer feedback, and the same movement is happening more broadly in product and service development. The insight is that you have to learn more about the customer and the customer's problem before you can produce a successful product. The original product idea is therefore almost always wrong, and the ability to learn quickly and rethink is decisive. The entrepreneur and investor Paul Ahlstrom describes it as "fail fast and learn to change" (Furr and Ahlstrom 2011), while Randy Komisar speaks of "getting to plan B" (Mullins and Komisar 2009). The need for iterative, customer-focused work was noted in the Harvard Business Review as early as 1986 (Takeuchi and Nonaka 1986), but has reached a wider audience only recently.

The point is not only the need for iterative development, but the importance of spending a great deal of time understanding the customer's problem before you start building. By talking to customers and observing them, you gradually learn their everyday reality and the problems they experience. The biggest problems are often the most interesting commercially. A need is moreover rarely obvious until you have begun using a product. Asking the customer what need she has is therefore often pointless, because she does not know. People rarely feel the need for a successful product until they begin using it, whereupon it suddenly becomes impossible to understand how they lived without it. The smartphone is a good example. Who had an explicit need for one before it existed, and who manages without it today?

A much-cited example from the early 2010s is Drew Houston, founder of Dropbox. His idea was that synchronising files between computers was a problem many people did not know they had. Investors dismissed it as small and already solved, but by believing in his vision, talking a great deal to potential users and building a product that was simple and worked from day one, Houston created a great success. The point is timeless: innovation is not about letting customers tell you what they want, but about understanding how they think and what problems they experience, and from that seeing what they do not yet know they want until they see it. That very ability, to understand a problem the customer cannot yet put into words, is incidentally at the heart of taking diagnostics into a new domain, something we return to throughout the book.

2.7 Soft products

For many manufacturers of capital products, services have become an ever more important part of the total business. One reason is that customers increasingly focus on their core business and the value they add. There is, for example, no value in itself for a bus company in owning its buses. What

matters most is being able to offer attractive travel services at a good margin. That of course requires access to buses, but not necessarily ownership. Demand for alternatives to the traditional product business has therefore grown.

Manufacturers too can have an interest in increasing the service content. In several industries competition has become so fierce that it is hard to sell pure products at a good margin. The focus therefore shifts toward offering the product in various service packages, for example leasing, which gives the customer greater freedom of choice and the manufacturer better scope to differentiate, since the packages are harder to compare and therefore easier to charge for. Services that complement the product, so-called add-on services, become interesting in the same way. Examples are extended warranties and insurance. Put a little pointedly, the product business partly takes on the role of a way to establish customer contact, in order then to sell higher-margin services.

A capital product controlled by software often contains functions the customer has not chosen to buy, and which are therefore not available. Many printers, for example, contain functions for copying and scanning. The customer can choose which functions to buy, but also expects the product to be adaptable when needs change. Initially it may be enough to print, but later one may also want to copy. The supplier can then switch on the functions that are requested.

Soft products is a term used to describe this kind of product, and in a wider sense to describe how service content takes on ever greater importance for companies that have historically sold products. AB Volvo is an example of a manufacturer that has used the term. Volvo Group today describes financing, leasing, insurance, service agreements, spare parts and maintenance as offerings that complement its core products, and its growing service business has become both a competitive advantage and a source of stability in a cyclical industry.

Why use the term soft product instead of service? There can be several reasons. The company may have gone from a pure product supplier to gradually acquiring ever more service content, and since its history and soul are spelled product, it becomes natural to call the services soft products. Customers are moreover used to buying products, and to avoid creating confusion in their world the services are described as soft products that complement the hard ones. With soft one can also mean that the products are flexible and can be adapted to the customer's needs of the moment, since they are software-based and functions can be switched on and off. A hard product, by contrast, is static and does not change over time.

Since the first edition, this has acquired an established name: servitization, the gradual shift of a manufacturer's centre of gravity from selling products to selling services and outcomes. In its purest form the business becomes outcome-based, where the customer pays not for the product but for the outcome it delivers. The classic example is jet engines sold on a power-by-the-hour basis, where the customer pays per flight hour and the manufacturer undertakes to keep the engines running. In Volvo's world there are counterparts in uptime and service contracts and in emerging equipment-as-a-service solutions. This is the soft product driven to its limit, and it is worth noting already here that an outcome-based business only works if you can guarantee availability. And availability, as we shall see, is fundamentally a question of diagnostics.

2.8 Software-controlled products

FIGURE 5

Configuration versus software download

Earlier in the chapter we described the move toward ever more capital products controlling their functions with embedded hardware and software. A major advantage of software-controlled functions is that you can easily change the product's properties. In the printer example, the supplier switched on the functions for copying and scanning. What actually happened was that the supplier either changed a setting in the software, which is called configuration, or downloaded new software, which is called software download.

Software-controlled products often contain several embedded computers that are loaded with software during manufacture. The software is then updated once the product is sold and in use, either to fix a fault or because the customer needs new functions. Software-controlled products must therefore be able to communicate with the outside world. Traditionally this has happened in a service workshop, often by cable or USB. When a modern passenger car is brought in, the software in the embedded computers is checked and updated as needed.

Since the first edition, however, a third path has become increasingly common, namely updating over the air (OTA). The workshop and the USB cable are no longer the only ways in. More and more products are updated remotely, sometimes overnight, without the customer having to do anything at all. This changes both diagnostics and the business fundamentally, and we devote section 3.9 to what it means.

In the printer example we assumed the user interface for the new functions was already in place. All the physical buttons and controls were already there, and by updating or configuring the software the functions were made available, for example by extending the menu with scanning and copying. But the physical interface a new function requires is not always in place. It may even be that the software for an as-yet-unavailable function already sits in the product, but that a physical control must be installed before the function can be used. Cruise control, for example, may already be in a car, but require a new stalk to be fitted before it can be used.

2.9 Processes: product development, manufacturing and aftermarket

A manufacturer of capital products often divides its work into three main processes: product development, manufacturing and aftermarket. In product development, sometimes called R&D, new products are developed. They are then produced in manufacturing. After sale, sometimes defined as a separate process, the products must be maintained and repaired, which happens in the aftermarket.

An ever larger share of the investment in product development of capital products goes into embedded hardware and software. Product development is increasingly about developing hardware and software, both within the product and in the support systems needed across its life cycle. Software has several large advantages over hardware, which we return to later. One advantage is that an updated version can be rolled out to the aftermarket in no time by being made available over the internet. Updated hardware must be series-produced, distributed and installed, which is more expensive and takes longer. That is one reason why investment in software is growing much faster, in relative terms, than investment in hardware, which is also growing.

The aftermarket has historically been seen mostly as a cost centre, for warranty cases for example, but today it plays an ever more important commercial role. One reason, as discussed above, is that competition has become so fierce in many industries that manufacturers struggle to make money on product sales alone. They therefore offer, alongside the product, various add-on services with higher margins. One can argue that the aftermarket has always played a very important role, because it is during that period that it is decided whether the customer will return. Even though the aftermarket is now seen as an important source of income, it remains a large cost item too.

The shift towards software-controlled products places greater demands on cooperation between, for example, product development and the aftermarket. A fault identified in the aftermarket may be due to a fault in the product's embedded software, which is corrected in product development. Previously the aftermarket handled all the spare parts needed for maintenance and could work relatively autonomously. Now the product can be updated with new software on an ongoing basis, and the changes reach both manufacturing and the aftermarket quickly. Because the aftermarket may want to sell upgrades, product development must produce a product that can be upgraded.

Companies that make capital products must handle ever more technically advanced and complex products whose development demands larger investments. At the same time competition is increasing, as are customers' demands for products adapted to their specific needs. This has led manufacturers increasingly to buy systems of hardware and software from suppliers, so-called third-party suppliers, for example seat belts from Autoliv. The third-party suppliers deliver systems that the manufacturers adapt to their unique application and integrate with other systems and components. That reduces the manufacturers' need to invest and lets them focus on their core business. Some manufacturers carry out essentially all development in-house, but the trend is clear: from in-house development towards adaptation and integration of purchased systems.

3 Software-controlled capital products

Capital products controlled by embedded hardware and software are more flexible and precise than their mechanical predecessors. The products are often based on a so-called product platform that contains functions common to several different products. Platforms give economies of scale, but also the ability to customise products. This development has made the handling of products, their configuration and their software, a complex task for every manufacturer.

3.1 Control unit: embedded hardware and software

FIGURE 6

A control unit (ECU): inputs, control unit and outputs, engine control example

image to be inserted here

The embedded hardware and software in capital products is often called control units, or ECUs (Electronic Control Units). A control unit may, for example, control gear shifting and the accelerator in a vehicle, or serve as the instrument panel in a boat.

The engine control unit was one of the first advanced control units. With an engine control unit, fuel delivery and combustion could be regulated based on, among other things, various sensor values, which improved both engine performance and reliability. Because software can also compensate for tolerances, engines no longer had to be built with the same mechanical precision, which reduced costs and removed a source of faults. The cost of engine development therefore fell, even though a new activity was added, namely the development of embedded software.

The engine control unit in a vehicle is almost always mounted on the engine itself. To cope with that demanding environment of high temperatures and dirt, the electronics must be carefully protected. The figure below shows, at a high level, how an engine control unit takes in inputs such as engine speed, throttle position, camshaft position, air temperature, oil pressure and boost pressure, and from them controls outputs such as the amount of fuel and the ignition.

Some other control units that often contain a great deal of software are the instrument panel in a car and the chassis control unit that governs the handbrake and accelerator in a lorry. The software in the control units is loaded during manufacturing, but updated as needed in connection with maintenance

or when the product is upgraded. Because the software controls the product's functions, security is very important. A car, for example, has software that controls the engine, brakes, steering and other safety-critical functions. As a general rule, capital products can therefore only be loaded with software through the manufacturers' own systems.

3.2 From cables to networks

FIGURE 7

From separate sensors with their own cables to a shared sensor on a network

image to be inserted here

Until the early 1990s, sensors and control units were connected by cables. A control unit therefore had to be directly wired to the sensors it used, and the same kind of sensor could exist in several places in the vehicle. One might, for example, need two sensors to measure the outside temperature, one for the engine and one for the climate control. The result was many cables and redundancy among the sensors.

During the 1990s, sensors began to be connected in networks. That allowed several control units to access one and the same sensor, for example the outside-temperature sensor. It also reduced the number of cables and the physical complexity of the products. The figure below illustrates the difference, from separate sensors with their own cables to a shared sensor on a network.

This development made new kinds of function possible. The instrument panel could, for example, warn the driver when the car is switched off, the key removed and the door opened, but the parking brake is not engaged. Such a function uses values from several different sensors and control units and was not possible before. Note, too, that it is a function based purely on software.

3.3 Product platforms

FIGURE 8

Product platform and products: common and optional functions

image to be inserted here

To achieve economies of scale in production, a manufacturer wants to create products that are as standardised as possible. That is one reason platforms are developed, consisting of functionality that can be included in several different products. Instead of developing every product separately, one tries to develop a basic product, the platform, containing functions that can be used across several products.

Suppose a car maker offers a number of large models, for example an estate, a saloon, a slightly sportier crossover and an SUV for the American market. The models differ, but a number of functions will be needed in all of them. By developing a platform that contains these common functions, the maker can reach several advantages, for example more efficient product development and manufacturing, because the platform's base functionality can be reused across many models, and purchasing of higher quantities of fewer components, which therefore cost less per unit.

Let us take a fictional and somewhat simplified example to illustrate the principles. A company develops a platform that contains six functions, to be used in three different products. In product A, functions 1, 2 and 3 are used, while functions 4 and 5 can be added later through an upgrade, and function 6 is not available for the product. Products B and C likewise contain a set of functions, some upgradable and some not. The figure below shows the relationship between platform and products.

Suppose all functions are software-controlled. That means the products can be upgraded with new functions in the aftermarket. Product A can, for example, be upgraded with functions 4 and 5 if the customer wishes. By developing a platform, the company has made its product development more efficient, while the software-controlled functions make it possible to adapt the products to customers' changing needs. On the same principle, many similar products can be developed relatively easily. Two products may, for example, have identical hardware and differ only in software, with one having a more powerful engine than the other. Through shared platforms, the car industry has been able to reach synergies even between different marques within the same group.

3.4 Advantages and disadvantages of product platforms

Product platforms bring several advantages. The two we touched on earlier are lower product development cost, because functions are reused across products, and more efficient manufacturing and purchasing thanks to standardisation. Another major advantage is that several products can be developed faster and, because they are software-based, upgraded to suit customers' changing needs. A large group working with platforms also has fewer components and spare parts to handle, since they are included in several different products.

This can also be sensitive, for example if products aimed at very different segments are based on the same platform. It is therefore very important to build a brand around one's products, spare parts and components. An Audi owner may not want a spare part to say Skoda, even though it is widely known that these marques share platforms developed within the Volkswagen group.

Another reason branding of spare parts matters is that manufacturers often have very good margins on their original spare parts. These parts are often developed by suppliers. Manufacturers used to forbid their suppliers from selling the parts under their own brand, which is no longer legal. That means competition increases and the same spare part can appear under different brands in the aftermarket, for example as an original part with the manufacturer's brand and as a part from a third party. The term original has been criticised for implying that the alternatives are plagiarism or pirate parts. Some manufacturers have therefore switched to the term genuine parts.

Taking the platform idea a step further, one can develop identical products that are branded differently for different markets. A well-known example is that GM sold Daewoo cars as Chevrolet in some markets. The problem was that not everyone perceived these cars as Chevrolets, but rather as Daewoos with Chevrolet badges, which was of course negative. The phenomenon is sometimes called badge engineering. Another problem with platforms is that a fault in one component can be costly to fix, because the component appears in many products on the market.

3.5 Upgrade and update

FIGURE 9

Update versus upgrade of a control unit

image to be inserted here

Based on a platform, manufacturers develop different products with different functions. The embedded software in the product's control units is loaded during manufacturing and updated over the product's life cycle, often in the aftermarket. The software may be updated because it contains a fault or has been improved. Another reason may be that a change of hardware requires the software to be updated. By update we mean loading a control unit with a new version of an existing piece of software.

To give the product new functions it must instead be upgraded. That can happen by changing a parameter in existing software, or by loading the control unit with different software that gives the product new functionality. The distinction between update and upgrade is central to the rest of the book, and the figure below illustrates it.

3.6 Variants

FIGURE 10

The variant explosion: versions of hardware and software over the life cycle

When a capital product is released it has a certain set of hardware and software, but both will be updated over the product's life cycle. Software can be updated to fix faults or introduce improvements. Hardware may need to be replaced because the original hardware has gone out of production and is no longer available.

The consequence is that there will be products with a wide range of variants of embedded hardware and software. These variants must work together, because they may be present in products out in the aftermarket. A calculation illustrates the problem. Assume a product consists of ten control units, that is, ten pieces of hardware and software, that the product is manufactured for four years, and that each piece of software is updated with two new versions per year. That gives eight versions of each piece of software, which must be able to combine with all versions of the others. The number of theoretical combinations is then eight to the power of ten. If, in addition, the hardware in the control units is replaced once during the life cycle, for example to add a sensor or because memory has run out, two to the power of ten combinations of hardware are added. The figure below illustrates this variant explosion.

One naturally strives for software compatibility when hardware is changed, so in absolute terms the example probably ends up closer to 80 than 160 unique pieces of software. On the other hand, replacing one control unit can trigger chain reactions, forcing the replacement of another. The point is that producers of software-based products must be able to handle a large number of variants, because both hardware and software are changed over the product's life cycle.

3.7 Managing product platforms

Manufacturers of vehicles and other modern capital products must be able to offer maintenance long after the product has left the factory. A lorry manufacturer is required by law to offer maintenance for up to fifteen years after the lorries are made. Lorries are high-technology products, and the hardware used in their development, such as memory and processors, will not be available throughout the product's life. A particular processor may be available for up to twelve months, and after about three years the hardware changes have become so great that the embedded software must be modified to work with the new hardware. There will be a number of such generational shifts over the product's life, which creates challenges in managing the products and their platforms.

To manage the software efficiently, one wants as few managed objects as possible. A managed object can be a piece of software unique to a product, or a component used across several products. When replacing one piece of software, version A, with another, version B, one wants to ensure that all instances of the old version can be eliminated. One also wants to avoid the new version introducing new functionality without the customer knowing, because the customer should be aware of, and pay for, an upgrade of the product's properties. One option is therefore to control the new functionality with parameters, so that the new software version can be installed without automatically activating new functionality.

There are many reasons capital-product makers nonetheless have a heterogeneous set of managed objects. It may stem from a strategic deal where, as a counter-requirement, one must buy and integrate a component from a local supplier. In management terms it might have been better to use an existing component, but then one might not win the deal. The commercial perspective naturally always comes first, while the management perspective must be kept in mind when planning technical development, so as not to incur large future costs.

3.8 Software and traditional spare parts

Software differs from traditional spare parts and upgrades in some important respects. Let us compare two cases.

A customer wants a tow bar fitted to their car and books an appointment with the workshop. The workshop has time immediately, but the actual fitting can only happen five days later, because the tow bar must be ordered and shipped there. At the visit the mechanic installs the tow bar and checks that it sits correctly and works. Because a screw was missing from the accessory kit for some reason, the mechanic used another screw of the same dimension that he happened to have at hand.

Another customer has a car of 120 horsepower, but it is starting to feel a little weak, so the customer wants to upgrade. For the car there is an accessory package that raises the output from 120 to 180 horsepower. The upgrade is done by loading new software onto the car's control units for the engine, gearbox and so on. The customer calls the workshop, which has time the same day. The mechanic connects the car to a computer that fetches the relevant software from the manufacturer's central system. The download takes a few minutes, and the mechanic can see on the computer that it went well and that the car is verified. The car looks exactly as it did before. What the mechanic does not know is that the software was released in six new versions over the past year, and that a great deal of logic was needed in the manufacturer's central system to find the right software for that particular vehicle. The computer also noticed that the car had old versions of three other pieces of software, which were updated in the background without the mechanic having to do anything.

In both cases an upgrade takes place. In the tow-bar case one had to wait five days for the accessory to reach the workshop. The software needed to raise the engine's output was sent over the internet and was available immediately. Downloading software is fast, but distributing physical products takes time.

When fitting the tow bar a screw was missing. The mechanic could see that with the naked eye, and could also find another screw to use instead. When the mechanic had downloaded the software, he could not possibly tell with the naked eye whether the upgrade had gone well, but had to rely on a program on the computer connected to the car. One cannot tell by eye whether a software upgrade has succeeded, which one can when installing a mechanical accessory. When the screw was missing, the mechanic could improvise a solution, which would never have been possible with a software upgrade. In one respect, the mechanical world's flexibility is lost in the digital one.

When the computer realised there were old versions of software in the car, an update was made. That meant the car was up to date after the visit, and that one could, once all cars had been updated, stop maintaining the old version. That kind of update can never be done with mechanical parts. There were surely components in the car that had been identified as problematic and were therefore replaced with updated spare parts, but those were of course not updated during the workshop visit.

One reason the software was updated is that one continuously reads out how cars are doing during workshop visits. Data from these read-outs is sent to the manufacturer, who analyses it to identify faults. If the faults can be corrected with software, new software is released that can be downloaded at a workshop visit. In this way the aftermarket and product development cooperate far more closely when upgrades are made with software. One consequence of this development is that, for example, large lorry fleets and armed forces can no longer work as autonomously as before. Manufacturers want to protect their profitable aftermarket business, while access to civilian workshops can sometimes be hard to obtain. It is a thread we pick up again in the chapter on the defence and security sector.

3.9 Software Defined Vehicles and OTA

We have now worked through control units, networks, platforms and variants, and in section 3.5 we made a point of the difference between upgrading and updating. In section 3.8 we saw how software has become a product in its own right, sometimes more important than the physical spare parts. Put those ideas together and you have, in essence, what the industry now calls the Software Defined Vehicle, the SDV. That is where the whole development is heading, and it changes diagnostics at its core.

The idea is simple to state. A vehicle's characteristics are determined less and less by the hardware and more and more by the software. Two cars can roll off the same line with identical hardware and still be different products, because they run different software, and one and the same car can become a better product after delivery without a single bolt being changed. What was once fixed at manufacture can now change throughout the product's life.

From a hundred control units to central compute

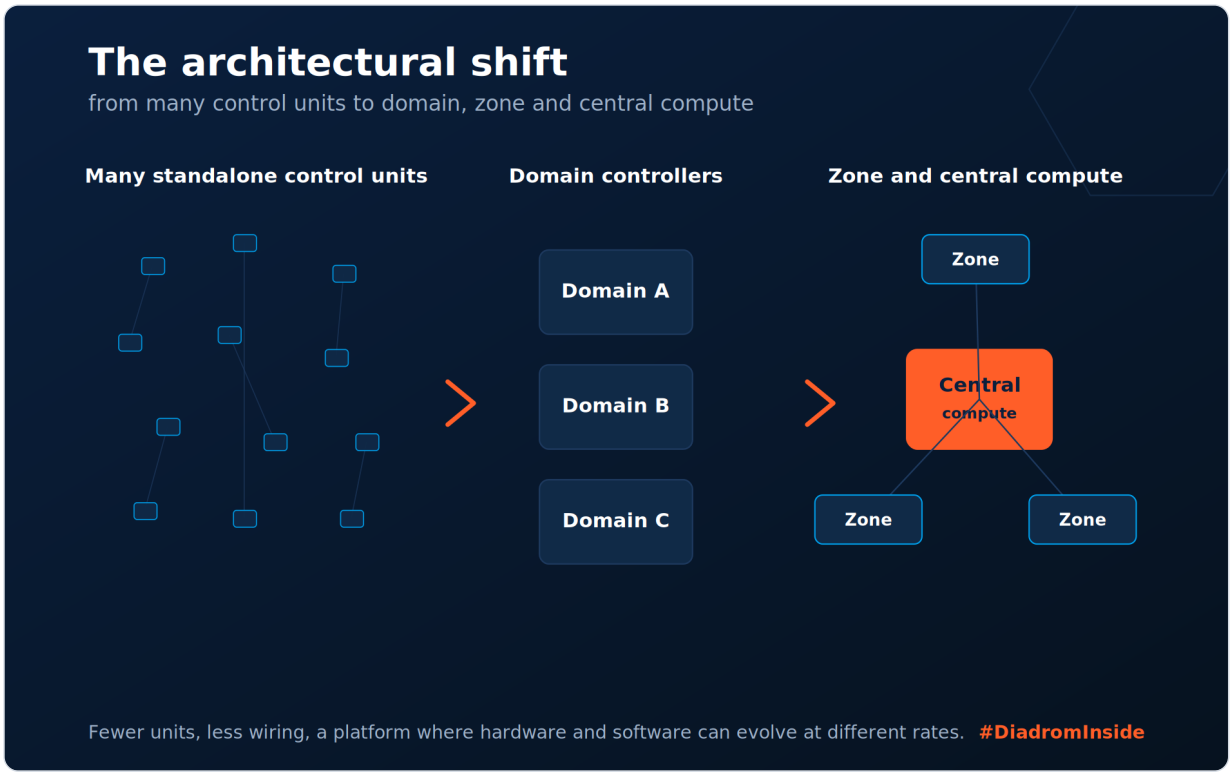


Figure 11. The architectural shift: from many control units to domain, zone and central compute.

For this to work, it is not enough to add more software. The architecture itself has to be rebuilt. In earlier sections we described how the number of control units grew until a modern car could have well over a hundred, interconnected in networks. That model becomes untenable when ever more is to be governed by simultaneous, updatable software.

The industry is therefore moving in three steps. First, standalone control units were merged into domain controllers, powerful units that run several functions at once. That was a good intermediate step. Then come zonal controllers, which group the electronics by where they sit in the vehicle rather than by what they do, and connect them to a central compute unit. The result is less wiring, lower weight, simpler manufacturing and, above all, a platform on which hardware and software can evolve out of step with each other. To this belong faster networks such as Automotive Ethernet and a service-oriented architecture, in which functions talk to one another as services rather than over hard-wired cables. It is the same movement we saw in section 3.2, taken all the way.

OTA: the product stops being finished

The other half of the equation is over-the-air, OTA, the ability to download and install software remotely, without a visit to the workshop. OTA is what makes the difference concrete. A product that can be updated over the air is never quite finished. It can be corrected, improved and given new features long after it has left the factory.

That sounds convenient, and it is. But for the aftermarket it is a quiet revolution. The distinction between upgrading and updating, which once played out at occasional workshop visits, now happens continuously and remotely. Diagnostics moves in the same motion, from a single moment when the vehicle is in for service to a constant stream of data from vehicles in operation. The workshop does not disappear, but it gains the company of a connection that never closes.

For the manufacturer this also opens a new line of business. Features can be sold after delivery, as options or by subscription, and a vehicle can earn money long after it has been sold. There is a lesson worth carrying here, though: customers appreciate new features, but react badly when something they feel they have already paid for is locked behind a fee. Where that line falls is a question of business and trust, not of technology, and we return to it in the chapter on the manufacturer's aftermarket business.

	The car as a product (before)	The car as a platform (SDV)
Features	Fixed at manufacture	Can be added long after delivery
Architecture	Many standalone control units	Domain and zone toward central compute
Updating	In the workshop, by cable or USB	Over the air (OTA), continuously
Value	Delivered once	Grows over time
Diagnostics	At a workshop visit	Continuous, connected, in operation
Business	One-off sale and spare parts	Also features and services over time
Security	Requires physical access	Must be protected remotely

Table: The shift from the car as a finished product to the car as an updatable platform. The same logic applies to all software-defined capital products, not only passenger cars.

Why this leads onward

The shift is no longer a vision of the future. During 2025 and 2026 the software-defined vehicle went from experiment to industrialisation, and close to half of all manufacturers now rank the transition as their single most important strategic question. Tech-native players such as Tesla and several Chinese manufacturers are ahead, while many established manufacturers are still closing the gap.

But with the opportunities come three questions, each of which the book devotes a later chapter to. When the product constantly sends data, how do you turn that data into better decisions and predictions rather than just more noise (chapter 6, data, AI and predictive maintenance). When software is loaded remotely, how do you know the update is genuine, unaltered and secure (chapter 7, cybersecurity in diagnostics). And when the same shift reaches systems where availability comes before everything else, what happens then (chapter 8, defence and security).

Before we go there, though, we will tie chapter 3 together with the book's perhaps most important model. For a car that is constantly connected and updatable is also a car whose state the manufacturer can actually know, in real time, across its entire population. And that, as it turns out, is precisely what is needed to climb all the way up the maturity model.

3.10 The maturity model

At the end of the previous section we made a point of the fact that a connected, updatable product is a product whose state the manufacturer can actually know, in real time and across its entire population. That is the very key to this section, the book's perhaps most important model.

How effectively a manufacturer can work with its capital products over their lifecycle depends largely on how good its control over the products is. There are of course other important factors too, from how well the product meets customer needs to brand and customer care, but control is the foundation. If you do not know how the products are actually used, for example, you cannot know which technical problems are most common in the market.

We have developed a maturity model to assess how good a manufacturer's control is over the embedded hardware and software in its capital products, and what is needed to improve it. Greater control creates the conditions for a better product and service, and therefore for differentiating the offering. The model runs from the most basic, knowing which products you can build, to the most advanced, optimising the customer experience through adaptations and improvements based on how the products are actually used.

The five levels

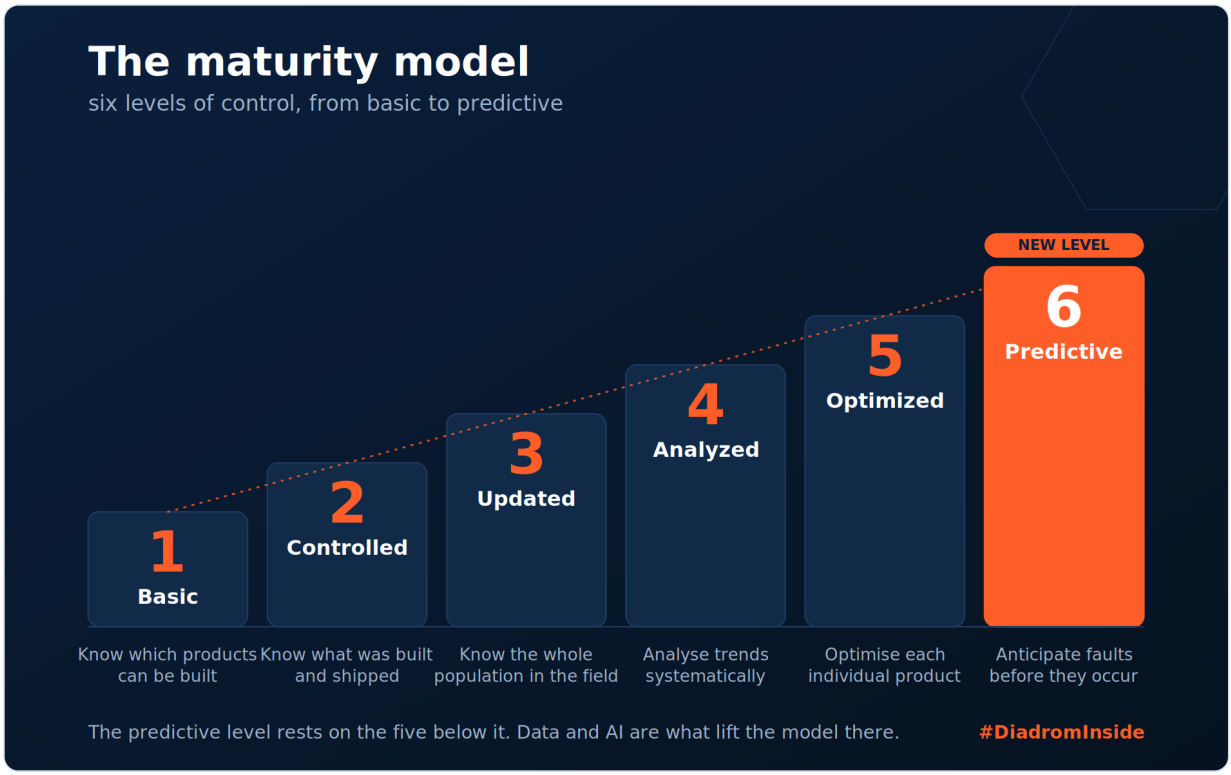


Figure 12. The maturity model, six levels of control, with the new predictive level.

A concrete example makes the ladder clear. At level 4 you can see that a particular car model tends to have cold-start problems in certain climates, and on that basis roll out a software update aimed precisely at that part of the population. You can also discover that a customer uses their product in a way that would make an engine upgrade worthwhile, and offer it. At level 5 such adaptations happen not just for a group but for each individual, based on its own use.

The new layer: a predictive level

The five levels have stood the test of time, but since the model was first formulated two things have happened that we devoted earlier sections to. Products have become connected and updatable (section 3.9), and the volume of data they produce has exploded. Together they make a sixth level reachable.

The difference from level 5 is the direction in time. At the optimised level you react and adapt based on how the product has been used. At the predictive level you act based on how it is going to behave. Instead of fixing a fault when it occurs, or even at the next service, you anticipate it and plan the maintenance in good time. That is the step from reactive and proactive to predictive, and ultimately toward products that diagnose themselves.

It is important to say that the predictive level does not replace the others, it rests on them. You cannot anticipate anything unless you first know which products exist (levels 1 and 2), have control of the population (level 3) and analyse it systematically (levels 4 and 5). AI is no shortcut past that control,

but a layer that becomes powerful only once the foundation is in place. How that layer is built, and why discipline matters more than the choice of model, we devote chapter 6 to.

Why it matters

The maturity model is not an academic exercise, but a map. It shows where a manufacturer stands today and what the next step requires. And the through-line is the same all the way up the ladder: the better the control over the embedded software, the better the product, the more satisfied the customer, the stronger the business. That is also why the rest of the book is about exactly this. In the next chapter we move from the model down into practice, to how diagnostics actually works in the aftermarket, where control has to be turned into concrete fault-finding and maintenance.

4 Diagnostics of capital products in the aftermarket

Diagnostics often brings to mind a situation where a service technician troubleshoots a product. It could be a car mechanic repairing a car in a workshop, or a service technician looking for a fault on a product in the field. Both situations play out in the aftermarket, that is, after the product has been developed, manufactured and sold, as a service or a product, and is being used by a customer in their operation. The problem is that the product has broken down and requires corrective maintenance to work again. Another very important part of diagnostics is preventive maintenance, that is, maintenance aimed at reducing the risk of unplanned stoppages.

4.1 Diagnostics in the aftermarket

Because a capital product is often complex, maintenance is carried out by a trained service technician. To help, the technician uses a computer with a diagnostic application developed for the aftermarket, which is connected to the product, for example with a cable.

In corrective maintenance the diagnostic application is used, among other things, for troubleshooting. There it is important to read out the product's fault codes, that is, codes meant to inform the technician of faults in the product. Fault codes can be more or less precise. Some pinpoint the area in which there is a technical problem, while others are very general and essentially only indicate that something is wrong. Many computer users recognise a fault code of the kind an error of unknown type has occurred, an example of an imprecise code that says very little about the actual fault. There are of course more precise fault codes, which may concern problems with the graphics card, memory and so on. More complex capital products such as lorries and aircraft display fault codes in the same way. Some are shown to the user in the instrument cluster, for example on a display by the speedometer, while others are intended for a trained technician and shown only in the diagnostic application.

Unplanned maintenance can be initiated by the user or by the product itself. The user may experience faults serious enough that the product must be repaired urgently, that is, one cannot wait until the next scheduled service, perhaps because the product cannot be used safely or because there is an imminent risk of unplanned stoppages.

When troubleshooting a capital product it is often valuable to discuss the problem with the user. In what situations does the fault arise? When did it first occur? In some systems one works systematically with the user-experienced problems in the form of so-called symptoms. By classifying the symptoms in tree structures one tries to narrow down the problem. Some diagnostic applications can, based on the fault codes read out and the symptoms the user has given, guide the technician through troubleshooting. For a given fault code or symptom there is then a set of tests that help the

technician narrow down the cause, and the result of one test determines which test should be done next.

In planned maintenance a number of service actions are often carried out as part of a service programme. At one service one might change the brake discs, at another the oil, and so on. During maintenance one often checks which software is present in the product's control units and, if needed, updates it, which can resolve a problem the customer has experienced. The poor cold start the customer complained of may, for example, have been fixed with a new version of the software that handles fuel injection at start-up.

Upgrading functions in a capital product requires a software download, or changing parameters in existing software. An upgrade may also require a physical installation. In an earlier example we described fitting a cruise control to a car, where the software was already in the vehicle but the control the driver uses was missing and had to be installed. When the hardware in a control unit has broken down it must be replaced. In some cases the software is already in the new hardware, but it is becoming ever more common for the hardware to be delivered empty and to have to be loaded with software.

4.2 Systems

FIGURE 13

The structure of the diagnostic system: user, capital product, interface box, computer and service technician

image to be inserted here

FIGURE 14

VODIA, an example of a mobile diagnostic application

image to be inserted here

FIGURE 15

OBELISK for the RBS 70 NG, built with Diadrom's Diag Studio

image to be inserted here

A diagnostic application runs on a computer connected to the capital product via a so-called interface box, which lets the application and the product communicate. In the application you can, among other things, troubleshoot, read and clear fault codes, view operating data, configure and calibrate the product, and download software. Some applications also contain information about the product, such as drawings and spare parts. The embedded software relevant to a particular product is often not in the application but in the manufacturer's central system, which means an internet connection is needed to perform certain diagnostics.

As an example one can mention VODIA, Volvo Penta's diagnostic application. It illustrates a point that runs through this book: a diagnostic tool is itself a software product that is developed and updated over time. Where an early version was a handheld application, the tools have since gone through several generations, and today's diagnostic applications are increasingly connected and cloud-based rather than tied to a single computer.

The internal software in the control units is updated continuously, and it is very important that the right software ends up in the right product. During maintenance the application therefore connects to the manufacturer's central system to identify whether there are new software versions for the product in question, and if so which. The central systems often contain replacement chains that define which software replaces which, and whether a replacement requires other updates. Another reason for storing the software centrally is the wish to keep control of it and ensure it is not modified, since the software governs many of the product's vital functions. That control becomes all the more important when products are connected, which we return to in the chapter on cybersecurity.

Manufacturers often want to store information about the product "as is" in their central systems. They want to know which software is in use across the product population and which fault codes are registered. The application therefore often reads out data about the product for storage and analysis centrally. By systematically collecting and analysing that data, faults and shortcomings can be identified and trends seen, and by correcting the faults and releasing new software versions, problems are solved efficiently without disturbing customers. More and more manufacturers also collect operating data wirelessly via telematics, which provides real field data in a way that was not possible with yesterday's mechanical products. Here lies the seed of what later becomes predictive maintenance and data-driven diagnostics (chapter 6).

Diadrom's own product Diag Studio is a platform for building exactly this kind of aftermarket diagnostic application for manufacturers. One example is Saab's application OBELISK for the RBS 70 NG air-defence system, built with Diag Studio. The test usually highlighted predicts when the

cooling system needs to be refilled with helium, by tracking how the cool-down time changes over time. It is an early and concrete example of predictive maintenance, and at the same time a bridge to the book's defence chapter.

4.3 Fault codes

A fault code (DTC, Diagnostic Trouble Code) is meant to inform a service technician about technical problems the product itself has identified. When the product is developed, one defines which conditions must be met for the product to register a fault code: when certain conditions are met, a particular problem should have arisen, and a specific fault code is then set. The goal is for the fault code to signal a specific fault, but what it actually signals is that a number of conditions are met, which ought to mean that a certain fault has occurred.

There is no one-to-one relationship between a single fault code and an underlying cause. The conditions for setting the code can be met for other reasons. A concrete example was documented in a study of how service technicians work in the marine industry (Kuschel and Ljungberg 2004). A boat had registered a fault code indicating that the fuel filter should be replaced, which was done at the workshop. The same code was registered again shortly afterwards, whereupon the filter was replaced once more. When the code was set a third time, a more thorough investigation was made, which showed that the real cause was dirt in the tank. The conditions for the code were met, but the code guided the mechanic in the wrong direction.

Some fault codes are shown to the user, for example in the instrument cluster, while others are shown only to the technician in the diagnostic application. Codes shown to the user may aim to inform of a problem that should be addressed soon. For serious problems the product can switch off functions itself to reduce the risk of further damage. A lorry can, for example, be put into a limp home mode, in which it can only be driven at very low speed, so the driver is not left stranded at the roadside but can make their way slowly to the nearest workshop. By reducing power, one wants to avoid the vehicle being used in a way thought to risk serious, perhaps safety-related, problems. Serious problems can also be encountered during planned maintenance, but they are often shortcomings the user did not experience as serious, otherwise the repair could not have waited for the planned service.

A fault code can be intermittent, meaning it has been registered but the conditions for registering it are not met right now. Intermittent codes are often hard to analyse, because one does not know how to recreate them. The technician can often find out when the code was registered and ask the user whether the product was used in any particular way then, whether the user noticed any problems, and so on. One way to catch intermittent faults is to use the diagnostic application to put the product into extreme states to recreate the problem. One can also use the application to record data from the product while it is in use, which in vehicle contexts is sometimes called a flight recorder.

4.4 Technical problems at different levels

FIGURE 16

The defect-to-system-failure model: how a fault can spread through the system

Technical problems can occur at different levels. If we view a software-based capital product as a system, we can categorise and analyse technical problems as a chain (Parhami 1997): a defect can put the product into a fault state, the fault state can give rise to erroneous logic, the erroneous logic can create erroneous data, erroneous data and signals can make parts of the system stop working, when parts stop working the value of one or more of the product's functions falls, and when the value falls far enough the system as a whole fails. The figure below illustrates the chain.

A fictional example makes the model concrete. Suppose an engine control system, in a certain state, makes a calculation error so that the amount of fuel to be injected is computed incorrectly. The system is then defective. The fault manifests only when an instance of the calculation error occurs, and erroneous data is created. The erroneous data causes an incorrect amount of fuel to be injected, which makes the engine run worse than it should. The engine running worse reduces the customer value the driver gets from the function engine. And if other subsystems base their calculations on the engine, the fault can spread, so that several functions deteriorate and the system as a whole eventually fails. There are many techniques for ensuring that faults do not spread and affect other subsystems, a theme that recurs throughout the book, not least in the chapter on the defence and security sector.

4.5 What is a problem?

FIGURE 17

Technical problems and customer-experienced problems

FIGURE 18

Services in relation to technical and customer-experienced problems

image to be inserted here

A fault code states a product's technical problems. Which problems a user experiences is another matter. A product can have a host of fault codes but a satisfied customer, and conversely a product can be technically faultless while the customer experiences problems. The question that arises is: should one regard a fault code as a problem if the customer experiences no problem with the product?

Troubleshooting and maintenance cost money, which either the manufacturer, for example under warranty, or the customer must pay. If the customer experiences no problem, the incentive to pay is low. Another aspect is that an update that fixes a fault code can change the product's properties, because the properties are controlled by software. If the customer was not dissatisfied, or does not consider the changed properties an improvement, an update that removes a fault code may instead lead to a dissatisfied customer. At the same time there are safety-related updates one naturally wants to install on all products in the aftermarket, regardless of whether the customer experiences problems.

Turning the perspective around, one can ask: is something a problem just because the customer experiences it as one? Perhaps it is a matter of a property not quite matching the customer's expectations. In software contexts one sometimes jokes that it is not a bug, it is a feature, meaning that something the user experiences as a fault in fact works as intended. One way to handle this is to distinguish between technical and customer-experienced problems. The technical problems are the product's fault codes, the customer-experienced ones are based on the customer's experience. The figure below illustrates the two.

A technical expert in one of our marine-industry projects put it that a fault code's relevance depends on what the customer is actually complaining about. If the customer experiences engine problems, an indication of a fault in the audio system is not relevant, but if the customer complains about sound quality it is highly relevant (Kuschel and Ljungberg 2004). The technician must therefore make a competent interpretation of what the customer says in order to judge what is relevant.

Alongside technical and customer-experienced problems, corrective maintenance can also be initiated by the manufacturer. Bulletins may have gone out stating that certain faults occur on certain models and are to be fixed at no cost to the customer, for example safety-related problems. Once one knows what is wrong, the repair is carried out by replacing one or more components with spare parts. It may also mean that the embedded software is updated, or that the product is calibrated with the diagnostic application after a component change. Finally the repair must be verified, either with built-in tests in the product or with the diagnostic application, to ensure the product really is fixed and the customer does not experience the fault persisting, which otherwise creates bad will. Any leftover spare parts must be returned, and the technician makes any warranty claims against the manufacturer.

Because capital products are increasingly becoming soft products whose properties can be upgraded, a third dimension should be added to the model: the services one offers customers, for example product upgrades. It is often said that selling is about defining a problem together with a prospective customer, and then showing that the product or service one offers is the solution. In the same way, a customer-experienced problem can be seen as a business opportunity, because an upgrade can give the product exactly the properties the customer feels are missing. The figure below shows how services relate to technical and customer-experienced problems.

4.6 Fault-code-based and symptom-based diagnostics

One way to work systematically with customer-experienced problems is to build a hierarchical structure of the symptoms one believes customers may experience. By talking to customers, one can document the customer-experienced problems in terms of symptoms. The symptoms can be linked to guided troubleshooting, and the customer-experienced problems can then be handled in the same way as technical problems, that is, fault codes. One strategy is to address only customer-experienced problems and serious technical problems, for example safety-related ones.

The symptom trees are hierarchical, with an experienced problem refined the further down the structure one goes. At the top level there might be the area brakes, at the next anti-spin problems and braking problems, and so on. The further down one goes, the more specific a symptom one can state. To identify which symptoms a customer experiences, the technician or service receptionist must talk to the user, who is not necessarily the customer, that is, the one who pays. One asks questions and tests various statements to select the right symptom. Sometimes it goes quickly, sometimes it takes longer. The technician's knowledge of the product's technology is important, but so is knowledge of the customer and the specific product.

There are manufacturers in the car industry that have removed the ability to read out fault codes and work only with customer symptoms. A problem with that is that the technicians lose the degrees of freedom to identify problems they have had historically. It can cause problems both with performance, that is, whether one can identify the faults from symptoms alone, and with changing an ingrained way of working at its root. One way to support the approach is to develop guided troubleshooting, that is, application support when identifying problems.

An alternative to symptom-based diagnostics is fault-code-based diagnostics, sometimes called up-front diagnostics, focused on the fault codes the product generates. Diagnostic applications have traditionally been focused on fault codes. The starting point has been a technician about to repair a product with technical problems, and the first thing one does is therefore to read out the codes. As we have noted, there is often, but not always, an overlap between a product's technical problems and the customer-experienced ones. A third variant that occurs is model-based diagnostics.

4.7 Customer-experienced quality

Customer-experienced quality depends greatly on whether one manages to repair the fault on the first attempt. As a rule of thumb, a customer whose problem is solved on the first attempt tends to be more satisfied than a customer who never experienced any problem, while a customer whose problem is not solved on the first attempt tends to be very dissatisfied, and the experienced quality drops dramatically. The bottom line is that you only get one attempt.

Experienced quality does not, however, depend only on the product itself, but also on the treatment received (Grönroos 1996). If the customer is met positively and senses a genuine interest in identifying and resolving the problem, the experienced quality tends to be higher. A further dimension is the relationship between the customer's expectations and experiences. If the customer expects a faultless product and fast maintenance, the reaction is negative at the slightest problem, and vice versa. That means a product and a service that are relatively good can still be perceived as poor, and the other way around. Another important source when identifying problems can be bulletins from the manufacturer, for example information on what is important to bear in mind right now.

4.8 Diagnostics in product development and manufacturing

Diagnostics is most often associated with the aftermarket, but the discipline is central already in product development and manufacturing. In product development, diagnostics is about developing products that can be maintained and upgraded efficiently. That means one must think through, already when the product is designed, how faults are to be detected and corrected, which fault codes and symptoms there should be, and how the embedded software is to be updated and upgraded over time. A product that is hard to diagnose becomes expensive to maintain across its whole life.

Connectivity and collected field data also let product development benefit from the same diagnostic information as the aftermarket. By analysing fault codes and operating data from products in the field, one can identify recurring problems and fix them with new software versions. The same technical solution can be used to test functions, for example by loading an updated piece of software remotely and evaluating it under real conditions.

In manufacturing, diagnostics is used to load the control units with the right software, configure and calibrate the product, and verify that each manufactured unit works before it leaves the factory. Many control units are delivered empty and programmed at the end of the line, and the same diagnostic tools and communication stack used in the aftermarket are often reused in production. Diagnostics thus ties together the book's three processes, product development, manufacturing and aftermarket, and investing in good diagnostics early gives more efficient maintenance and more business opportunities later.

4.9 Functions in an aftermarket diagnostic application

An aftermarket diagnostic application gathers a number of recurring functions. Here are the most important.

Read and clear fault codes. Fault codes (DTCs) are often the starting point for troubleshooting, and can both be read out and, once a fault is fixed, cleared.

Read operating data and sensor values. Examples of operating data are operating hours and fuel consumption, of interest both in maintenance and to a fleet owner. In troubleshooting, more detailed sensor values can be valuable. Some products store historical values, others provide only real-time data.

Set and change customer parameters. It is often possible to adapt the product's properties to the customer's wishes, for example the maximum setting of the seat heating in a car.

Calibrate the product after replacing a component. After a component replacement, calibration may be required. If a new bucket has been fitted to a construction machine, the machine may need to learn the bucket's extreme positions.

Software download and configuration. One way to maintain a product is to provide it with the latest software. Download is also often needed after replacing a control unit, since many control units are delivered empty today and must be loaded after installation. These cases concern updating. Upgrading, by contrast, means changing the product's functions by loading a new type of software.

Troubleshooting. One of the most important functions. The main variants are fault-code-based and symptom-based troubleshooting. Guided troubleshooting means you get step-by-step help in pinpointing problems defined by fault codes or symptoms.

Information. For maintenance and upgrading, information about the product is often needed: spare parts and accessories, repair instructions, details of design and function, and news and fault fixes from the manufacturer. The information is sometimes integrated in the application, which makes it quick to navigate from a problem to the right instruction or part, and sometimes in a separate system.

Integrated applications. Some diagnostic applications have interfaces to other systems, for example CRM systems with customer information, applications for ordering spare parts directly from the diagnostic application, and support flows for escalating problems from service technicians to market companies and manufacturers.

To these classic functions, two have been added as products have become connected.

Connected and remote diagnostics. Diagnostics no longer requires that product and technician be in the same place. With connectivity you can read fault codes and operating data, and in some cases perform operations, remotely. It is used to catch problems early and avoid unplanned maintenance, and in product development to, for example, load and evaluate updated software remotely, quickly and efficiently.

Secure software download. When software is loaded remotely, it is not enough that it ends up in the right place. It must also be genuine, unaltered and authorised, which requires authentication, signing and verification. Secure software download has therefore gone from a technical detail to a precondition, and we devote the whole of chapter 7 to what it means.

5 Diagnostics in the automotive industry

In the car industry, manufacturers have historically held a great deal of power. Among other things, they have been able to control which workshops may carry out maintenance. That is changing through legislation, not least within the EU, which we cover in the section on legislation further on.

5.1 Diagnostics and telematics

Diagnostics from a manufacturer's perspective is about developing, producing and maintaining software-based capital products in a way that enables high availability and flexibility. That in turn aims to let the customer, or user, create value and do business with their own customer, the end customer. High availability, and therefore little or no unplanned maintenance, means the customer's production of goods and services can be carried out efficiently and productively, which creates opportunities to do business with the end customer. Diagnostics in the aftermarket is about maintenance and the upgrading of functionality. Diagnostics in product development is about developing products that enable efficient maintenance and upgrading.

Diagnostics often happens with the capital product and the service technician in the same place, for example in a workshop or in the field. It can also happen at a distance, that is, with the product and the technician in different places, which is often called remote diagnostics. In the aftermarket, remote diagnostics can be used to read fault codes and identify problems early, in order to avoid unplanned maintenance. One can also collect operating data to try to predict how long a component will last, or to raise an alarm about changed maintenance needs. In product development, the same technical solution can be used to test software-based functions. If one wants to try an updated piece of software under very hot conditions, for example, one can load it remotely and evaluate the result quickly and efficiently.

Diagnostics focuses on the capital product itself, that is, maintenance and upgrading, while telematics often refers to wireless add-on services that concern the customer's operation but not the product itself. A haulier can, for example, use telematics for order handling and remote diagnostics to read fault codes at a distance. Reading out vehicle data remotely, such as fuel consumption, can, depending on use, be regarded as both remote diagnostics and telematics. There is a wealth of definitions and descriptions of remote diagnostics and telematics in the research literature, and rigorous definitions are often lacking, which is one reason for our attempt at conceptualisation above.

Since the first edition was written, this development has accelerated. Products are now increasingly connected straight from the factory, and whole fleets of vehicles, machines and systems report data continuously. That makes remote diagnostics the norm rather than the exception, and shifts the

emphasis from fixing faults that have already occurred to detecting them in time. The connected fleet is also the precondition for the predictive maintenance and data-driven diagnostics we devote chapter 6 to, and it places new demands on cybersecurity, which we return to in chapter 7.

5.2 Aftermarket diagnostic applications

FIGURE 19

OEM versus independent: a designed comparison of functions (Table 2)

image to be inserted here

In the automotive industry there are, broadly, two kinds of aftermarket diagnostic application: those developed by the vehicle manufacturers and those developed by independent players, often simply called independent. To these is added a third path, the custom application, which turns out to be the bridge to the book's chapter on defence.

The manufacturers

The manufacturers, often called OEMs (Original Equipment Manufacturers), are required by law to offer maintenance and spare parts for a certain number of years after a vehicle has been sold. Their diagnostic applications, sometimes called OEM tools, generally provide the best functionality on the market.

The OEM tools are built on one underlying assumption: that you should replace the smallest part in the shortest possible time, because that is the most cost-effective. This has two consequences. One is that a great deal of training and experience is needed to master the tools. The other is that a major cost for the manufacturer is warranty obligations, which makes it central to be able to judge whether a fault is a warranty matter or not. That is an important starting point when the manufacturers' applications are designed.

In the passenger-car industry, the focus is increasingly on the customer's perceived problems rather than the purely technical faults. Put simply: as long as the customer is satisfied, the fault is not fixed. Diagnostics thus becomes symptom-based, taking its cue from the symptoms the customer actually experiences, and a technical fault becomes a fault to the degree the customer experiences it as a problem.

But if you instead prioritise availability over perceived problems and maintenance cost, you make different choices. Then it can be more important to swap a larger component quickly than to isolate the fault at a lower, more cost-effective level. Hold that thought, because it is the key further on.

Two things have, moreover, strengthened the manufacturers' position since the model was first described. One is the connected channel: through telematics and OTA (section 3.9) the manufacturer reaches the vehicle continuously, something an independent generally cannot. The other is cybersecurity: the most sensitive operations, such as software download and configuration, now require secure and authorised channels, which we devote chapter 7 to.

Independent

Historically, only the manufacturers could offer comprehensive diagnostics, because the technology was proprietary and outsiders lacked the specifications needed to understand it. That has changed through laws and regulations that have forced the manufacturers to open up and support a number of open vehicle protocols. As a result, independent players can increasingly develop equipment for diagnosing the manufacturers' products.

The independent tools cannot provide the same broad and deep support as the OEM tools. Instead they compete on lower price and on covering several makes and models in a single application. Some focus on a part of the vehicle, such as Bosch's tools for common-rail engines. Legislation continues to widen independents' access to the basics, which we go through in section 5.3, but the most far-reaching and security-sensitive operations remain, in practice, the manufacturers' domain.

Functionality	Manufacturer (OEM)	Independent
Read and clear fault codes	Yes	Yes, now broadly, driven by legislation
Read operating data and sensor values	Yes	Partly, increasingly
Edit and set customer parameters	Yes	Partly
Calibration after component replacement	Yes	Limited
Software download and configuration	Yes	Not in practice
Bug-fixing of embedded software	Yes	No
Loading software after replacing an empty control unit	Yes	No
Configuring the vehicle platform (increased performance)	Yes	No
Advanced settings	Yes	No
Connected and OTA-based diagnostics	Yes, via its own channel	No or limited

Table 2: A rough comparison between diagnostic applications from manufacturers and independents. The security-sensitive operations remain OEM-dominated, partly because of the cybersecurity requirements (chapter 7), while legislation (section 5.3) continues to widen independents' access to the basics.

The custom application

There is a third path. An organisation that uses capital products may choose to develop its own, custom diagnostic application. One motive is custom telematics, for example to follow up availability and status. A fleet owner wants to know how available the vehicles are and how much fuel they consume.

Another and more interesting motive is that the maintenance itself has characteristics that set it apart, in important respects, from traditional workshop diagnostics. And here the thought we asked you to hold returns. Maintenance in military contexts can be very different from maintenance in a typical workshop. The focus may be on being able to carry out maintenance quickly with a limited spare-parts inventory, rather than on prioritising low cost and daily deliveries.

The purpose of diagnostics in such a domain therefore differs from both the manufacturer's and the independent's. The manufacturer's starting point is to replace as small a part as possible, which is why the application likes to pinpoint the fault in a single sensor that may take a long time to replace. A quite different starting point is to isolate the problem at a higher level and quickly swap a larger component, precisely to maximise the vehicle's availability.

It is the same diagnostic discipline as throughout the book, but governed by a third logic in which availability comes before cost. That logic, and the domain where it is driven to its limit, we devote the whole of chapter 8 to.

5.3 Legislation

An important driver behind the move toward ever more software-based capital products is ever stricter legislation. It regulates the products' environmental impact and safety, and in recent years also cybersecurity and who has the right to data. This section is deliberately dense with facts, because this is an area that moves quickly and where exact dates matter.

Emissions, OBD and a standard that spread

Ever since the 1970s, legislators have tried to regulate the maximum limits for emissions from motor vehicles. The American Clean Air Act and, in the mid-1990s, the requirement from California's Air Resources Board (CARB) that vehicles be able to diagnose for themselves that they stay within permitted emission limits, laid the foundation. That standard is today applied, in practice, throughout the Western world, and covers trucks and buses as well. Violations of OBD (On-Board Diagnostics) are punishable, and manufacturers are obliged to make tampering with the system difficult.

By OBD we mean that a vehicle has built-in diagnostics and can give off fault codes and other vehicle data. The dominant standard is OBD-II, which specifies both the connector and the protocols for communicating with the vehicle, and which appears in variants such as EOBD in Europe and JOBD in Japan. Two purposes have driven the work: to reduce emissions through measurable limits, and to create an open interface for third-party suppliers. The latter made it possible to develop one's own diagnostic programs and fleet-management systems, services that previously only the manufacturers' own tools could provide.

From Euro 6 to Euro 7

Euro 6, and its counterpart Euro VI for heavy vehicles, has been the European emissions legislation. It is now succeeded by Euro 7 (Regulation (EU) 2024/1257), adopted in 2024, which brings all vehicle categories together in a single regulation. Application is phased: new types of passenger cars and vans must meet Euro 7 from 29 November 2026 and all new such vehicles from 29 November 2027, while heavy vehicles follow in 2028 and 2029. New with Euro 7 is that it also regulates wear particles from brakes and tyres, sets requirements on the durability of batteries, and requires vehicles to stay within the limits for a longer part of their service life.

For the aftermarket, one part of the framework is particularly important: RMI (Repair and Maintenance Information). In short, the manufacturer must make available the information needed to maintain the vehicle, but not such information as would upgrade the vehicle's function. You should, in other words, be able to maintain (update) but not modify (upgrade), and the software download itself should, for security reasons, be done with the manufacturer's authorised tools. That very security aspect is now governed by a regulation of its own, to which we shortly return.

The block exemption and the right to repair

The EU generally wants to encourage competition and is therefore wary of agreements that work against it. Within the automotive sector there is, however, an exception, the so-called block exemption (Motor Vehicle Block Exemption Regulation, Regulation (EU) No 461/2010). The reasoning has been that service and maintenance, for safety reasons, must be carried out by trained personnel with the right equipment.

The block exemption allows manufacturers to build networks of authorised workshops, but not at the expense of competition. They must therefore also give independent workshops access to technical information, spare parts and training, and the workshops may use parts that are not original. A partial right to repair has been built in here ever since 1995. The block exemption has been extended to 31 May 2028, and the updated guidelines from 2023 state explicitly that aftermarket operators are to have continued access to the vehicle-generated data needed for repair and maintenance. In parallel, the EU has adopted a broader directive on the right to repair (Directive (EU) 2024/1799), which applies to goods in general and which, among other things, establishes a European repair platform.

Cybersecurity and software updates

When the vehicle became connected and updatable, an entirely new category of legislation was added. The UN body UNECE adopted R155 and R156, which require manufacturers to have a management system for cybersecurity (CSMS) and for software updates (SUMS) respectively, with ISO/SAE 21434 as the supporting engineering standard. Without these in place, no type approval is granted. We devote the whole of chapter 7 to what this means in practice, but it belongs here too: secure software download is now not merely good practice, but a legal requirement.

The EU Data Act: vehicle data is opened up

Perhaps the most far-reaching novelty is the EU Data Act (Regulation (EU) 2023/2854). It entered into force on 11 January 2024 and has applied since 12 September 2025, when the Commission also published specific guidance on vehicle data. Its meaning is that users, that is, owners and fleet owners, gain the right to access the data their connected vehicle generates, and to nominate a third party to receive the data on their behalf, on fair and non-discriminatory terms. Connected products must, moreover, be designed so that the data is available from the start.

Two boundaries are worth noting, because they say something important. The manufacturer's trade secrets, such as proprietary calibration parameters, fall outside. And so does derived or inferred information, for example predictions about future condition. In other words: the raw data is opened up, but the conclusions and the refinement are not. It is a piece of legislation that almost happens to confirm the book's point, that data is not value in itself, but becomes value only when someone turns it into a service. We develop that in chapter 6.

Regulation	What it governs	Status and dates
Euro 7, Reg (EU) 2024/1257	Emissions, including wear particles and battery durability	New types of cars and vans 29 Nov 2026, all new 29 Nov 2027, heavy vehicles 2028 and 2029
Block exemption (MVBER), Reg (EU) 461/2010	Aftermarket competition, independents' access to data, parts and information	Extended to 31 May 2028, updated guidelines 2023
Right to Repair, Dir (EU) 2024/1799	Consumers' right to have goods repaired	Adopted 2024, EU platform operational by 1 Jan 2028
UNECE R155 and R156	Cybersecurity (CSMS) and software updates (SUMS)	In force 2021, all newly produced vehicles from July 2024. See chapter 7
EU Data Act, Reg (EU) 2023/2854	Access to data from connected products	In force 11 Jan 2024, applies from 12 Sep 2025

A comment on the legislation

The overall movement is clear. Vehicle technology and vehicle data are being opened up, and the manufacturers' historical lock-in is weakening. But it would be premature to think the manufacturers are losing their strong position. That data is made available often means it is handed over in raw, untagged form, and a great deal of work is needed to make it usable. Data structures differ, much remains proprietary on grounds of upgrade or security, and the most sensitive operations require authorised and secure channels.

This leads to an uncomfortable but useful conclusion. Legislation gives more players access to the data, but at the same time it raises the value of being able to do something meaningful with it. Whoever masters diagnostics, and can turn an opened data stream into dependable decisions in a secure way, stands stronger than ever. That is where the following chapters take over.

6 Data, AI and predictive maintenance

In section 3.9 we opened the data tap. When a product becomes connected and updatable, it also begins to send out a constant stream of information about how it is faring and how it is used. In the book's maturity model (section 3.10) the highest level, the predictive one, was long mostly an ambition to strive toward. Data and AI are what finally make that level reachable. But only if it is done with discipline, and it is that discipline this chapter is about.

6.1 The data deluge and an uncomfortable truth

The amount of data a modern vehicle can produce is staggering. Where the classic CAN bus gave a modest stream, on the order of a few tens of kilobytes per second and a couple of hundred measured values, modern interfaces can move megabytes per second and tens of thousands of variables, and control units for autonomous driving are heading toward gigabytes per second. The tap is no longer dripping. It is wide open.

Here, though, lurks the first trap. Data is not value. Services are value. Collecting data is easy, almost too easy. Turning it into something a customer will pay for is hard. It is fashionable to say that an AI is worthless without customer data, and that is true, but the flip side matters just as much: data without context and quality is merely noise, and an AI will happily turn noise into confident nonsense. Garbage in, garbage out.

6.2 Why most organisations fail

Most organisations celebrate too early. They buy in AI tools, produce an impressive demo, and believe the value is in the bag. But between a demo and a figure on the bottom line lies the hard work: changing processes, ways of working and, ultimately, the operating model itself. Whoever stops at the tool gets no effect, only a cost.

To this comes the fact that the generic models will not save anyone. When everyone has access to the same base models, it is not the model that separates the wheat from the chaff, but the data. And not just any data, but your own, context-rich data, tied to real diagnostic knowledge. A fault code, a log or a signal says rather little on its own. We touched on this already in chapter 4: a fault code is not the fault, only a clue, and the path from clue to diagnosis requires a model of how the system actually fits together. An AI inherits exactly that limitation. It is never better than the diagnostic understanding behind the data it is fed. This, incidentally, is not a new thought. As far back as the 1970s, Frederick

Brooks pointed out that there is no silver bullet in software engineering, that the real difficulty lies in the inherent complexity rather than in the tools (Brooks 1975), and that holds just as much for AI.

And then, finally, the question of who owns the data and on what terms it may be used. We dealt with that in section 5.3, but it stays with us as a precondition throughout this chapter. Without the right to use the data, it does not matter how good it is.

6.3 The discipline that turns data into value

FIGURE 20

From raw data to value: data, information, decisions and services

image to be inserted here

The good news is that the diagnostic discipline is precisely what turns data into value. Three moves do the job.

The first is the path from data to prediction. Instead of waiting until something breaks, we want to foresee it. With predictive and condition-based maintenance, a model learns how a unit behaves when it is healthy, often as a kind of virtual sensor, and raises a flag when the behaviour begins to drift before a critical threshold is crossed. You can then estimate the remaining service life and plan the maintenance in good time, that is, move from reactive to proactive. None of this is new in essence. OBELISK, which we meet in chapter 8, was already predicting years ago when a cooling system needed refilling, by tracking how the cool-down time changed. What AI adds is scale: doing the same thing for an entire fleet, automatically.

The second move is to choose where the computation should happen. Not everything needs to go to the cloud.

	Edge (in the vehicle)	Cloud (central)
Latency	Very low, decisions in real time	Higher, depends on connectivity
Bandwidth	Processes locally, saves transmission	Requires data to be sent
Availability	Works even when disconnected	Requires connectivity
Privacy and security	Data stays in the vehicle	Data leaves the vehicle
Strength	Fast decisions close to the source	Heavy learning across the whole fleet

In practice the two are usually combined. Edge for what has to be fast, local and independent, the cloud for the heavy learning across many vehicles. Where the line falls is a trade-off between latency, bandwidth, cost and security.

The third move is to put AI on top of what we can already do. On top of the maturity model, AI becomes a layer above the optimised level, the thing that helps us not only to react and maintain but to anticipate and optimise. On top of guided, symptom-based troubleshooting (section 4.6), AI becomes an amplifier that can draw on the experience of an entire population at once. And when you use today's generative models, of the LLM and RAG kind, the art is to ground them in real diagnostic knowledge: service history, manuals, the manufacturer's diagnostic databases, ECU signals and fault codes. Then the model answers from actual context instead of guessing. Data quality and context are, in other words, first a diagnostics problem and only then an AI problem.

6.4 #DiadromInside: from noise to decisions

This is where Diadrom's 25 years of diagnostic discipline make a real difference. The difference between a fault code and a symptom, the path from defect to system failure, the maturity model, everything the book has built up, is precisely what makes an AI dependable rather than a confident parrot. Whoever masters diagnostics masters the data, and whoever masters the data can turn it into decisions.

Diadrom is actively moving in that direction, among other things through the Dia & Drom project, which is about deep vehicle integration where AI is fed with contextual signals and diagnostic knowledge for manufacturers and their customers alike. The business value is concrete. Predictive maintenance lowers warranty costs and raises availability, and the data becomes the basis for new services, something we return to in the chapter on the manufacturer's aftermarket business. For a defence capability with a limited spare-parts inventory, as we are about to see, the ability to foresee faults is not a convenience but a decisive advantage.

But all of this rests on one precondition. The moment data streams out of the product, and software and commands stream back in, every such channel has to be one you can trust. Otherwise the fine data and the clever model are built on loose ground. How that trust is built is the subject of the next chapter.

7 Cybersecurity in diagnostics

In section 3.9 we watched the car become a platform that is updated continuously, and in section 4.9 we saw how diagnostics increasingly happens connected and at a distance. It is convenient, powerful and inevitable. But it comes at a price. The very channels that let a workshop or a cloud load new software into a control unit are exactly the doors that someone with bad intentions most wants to get in through. When diagnostics becomes connected, security stops being a detail and becomes a precondition.

That ties back to the book's central idea. A chain is never stronger than its weakest link, and a single unverified update can be that link. Nothing works until everything works, and these days that also means: nothing can be trusted until everything can be trusted.

7.1 When a convenience becomes a way in

FIGURE 21

The communication stack (OSI) and secure software download with signing and verification

image to be inserted here

The diagnostic interface and the update path are privileged. Through them you can write software, unlock functions and read out data. That makes them invaluable to a workshop and just as invaluable to an attacker. A connected vehicle is a digital device on wheels, and its attack surface grows with every new interface.

To this comes an uncomfortable truth: today's security has a best-before date. A vehicle that was secure when it left the factory is not necessarily secure two years later, once attackers have learnt new tricks. The question is therefore not only whether an update is good, but whether it is genuine, unaltered and authorised, and whether you can prove it across the vehicle's entire life.

7.2 From nice to have to a licence to sell

The legislator has answered that question. The UN body UNECE, through its WP.29 forum, adopted two complementary regulations that have changed the playing field. They entered into force in 2021, became mandatory for new vehicle types in July 2022, and apply to all newly produced vehicles from

July 2024 in the countries that apply the framework, including the EU, the United Kingdom, Japan and South Korea.

Regulation	What it governs	In brief
UNECE R155	Cyber Security Management System (CSMS)	Manage cyber risk across the whole lifecycle, from design through operation to decommissioning. Built on recurring threat analysis and risk assessment (TARA).
UNECE R156	Software Update Management System (SUMS)	Secure, traceable and tested updates, including OTA. The certificate is valid for three years.
ISO/SAE 21434	Engineering standard for cybersecurity	Shows how the requirements are met in practice. R155 says what is to be achieved, 21434 says how.

The point is not the alphabet soup, but the teeth in it. Without valid CSMS and SUMS certificates, a vehicle type receives no type approval, and without type approval it may not be sold. Cybersecurity has thus gone from a technical virtue to a licence to operate in the market. Some manufacturers have even chosen to retire models rather than meet the requirements.

This does not only press the vehicle manufacturers. Because the manufacturer is responsible for security across the entire supply chain, suppliers too must be able to show that they measure up, often through the industry's information security assessment, TISAX, and by being able to provide evidence that holds up to an audit. For a specialised software partner this is not a burden, but a reason to exist.

7.3 How trust is built in practice

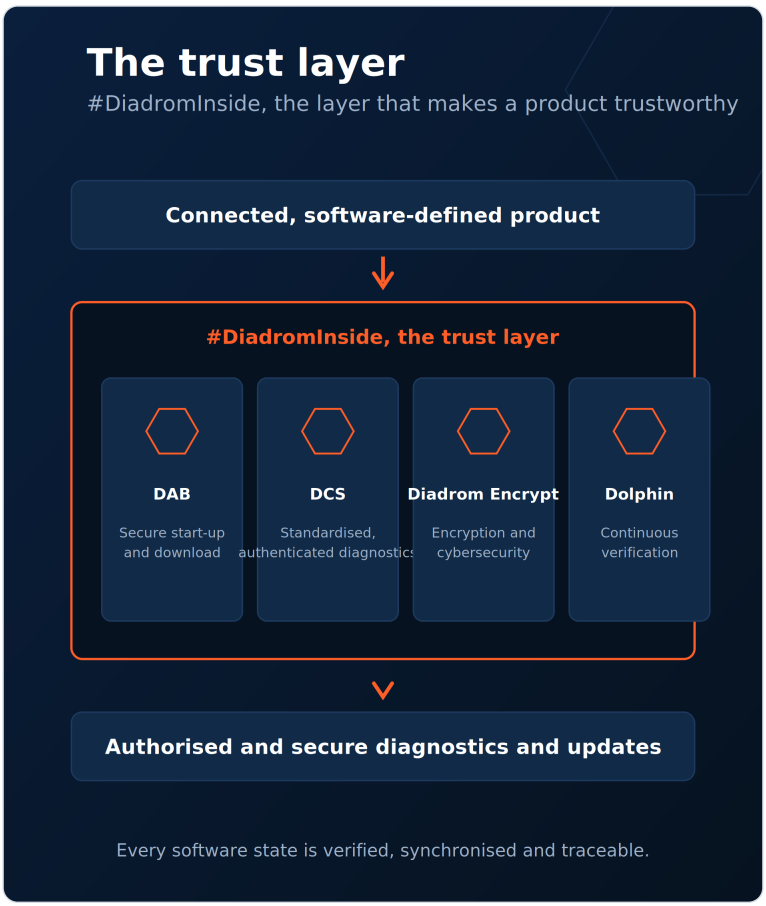


Figure 22. The trust layer: the #DiadromInside stack (DAB, DCS, Diadrom Encrypt, Dolphin).

Trust is built in from the start, not bolted on afterwards. Behind the fine words lies a set of concrete mechanisms, and you will recognise most of them from earlier chapters, only now with a security lens on.

The first is secure software download. Every update has to be genuine, that is, demonstrably from whoever it claims to be from, and unaltered, that is, not tampered with along the way. This is solved with digital signatures, so that only signed and verified software is installed, and with a built-in self-test that checks the firmware's integrity before it is put into use. Just as important, an interrupted or rejected update must not be allowed to knock out the unit entirely and render it inoperable.

The second is the bootloader as a security construction. Diadrom's Autotech Bootloader is built in two parts. The primary one starts the unit and is deliberately without flashing functions, precisely so that an attacker who gets into the channel cannot use it to write unauthorised software. The primary part cannot be overwritten either, nor can the keys on which security rests. Security thus lies in the structure itself, not in an extra layer.

The third is access and authentication. The standardised diagnostic protocol (UDS) has services for exactly this, Security Access and Authentication, so that only authorised test tools and back ends gain privileged access to a control unit.

The fourth is the cryptographic foundation in hardware. Verifying authenticity and integrity requires heavy cryptography, and if it runs on a single processor core it blocks everything else for a long time. It is therefore placed in isolated hardware, an SHE module or a Hardware Security Module, called via a standardised interface (the AUTOSAR Crypto Service Manager). On top of that, Diadrom adds an abstraction layer toward the security hardware, so that the same secure code works regardless of which semiconductor vendor has been chosen. That sounds like a technical nicety, but it is a strategic strength. When component supply wavers, as it did during the semiconductor shortage, you can switch vendors without rebuilding security, and the question of independence leads straight into the next chapter.

The fifth is continuous verification. Security is not a one-off check, because the threat picture is always moving. Testing is therefore automated and run on every change, in a continuous integration chain. In one of Diadrom's assignments, the build time for a release was cut from three days of manual work to one hour of automated execution, with every code change tested. Trust that is maintained, not trust that is assumed.

7.4 #DiadromInside in the trust layer

Each of the mechanisms is useful in itself. Together they form a coherent layer of trust: the Autotech Bootloader for secure start-up and download, the communication stack DCS for standardised and authenticated diagnostics, the cybersecurity stack Diadrom Encrypt, and the test tool Diadrom Dolphin that verifies it all continues to hold.

This is not theory. Diadrom's software is in serial production, including in the Volvo EX90, and meets TISAX, UN R155 and R156. Together with Nippon Seiki, Diadrom implemented the first cybersecurity concept in Volvo's SPA2 platform. This is #DiadromInside in its most literal form, a layer that rarely shows but that decides whether a connected, updatable product can actually be trusted.

And here a question arises that we have so far only touched on. For a passenger car all of this is a requirement in order to be sold. But what happens when the same connected, software-defined and updatable system has to work where availability comes before everything else, where an attacker is after not data but the disabling of a capability, and where control over your own software, independent of any single vendor, becomes a question of sovereignty? Then we have left the rules of the market and arrived at those of defence. That is where we go next.

8 Diagnostics in the defence and security sector

Earlier in this book we noted in passing that maintenance in military contexts can be very different from maintenance in an ordinary workshop (section 5.2). This chapter is about exactly that difference, and about why defence and security have become the domain where everything we have described so far is pushed to its limit. It is the same diagnostic discipline as in the automotive industry. The difference is that the demands here are the hardest imaginable, and the price of failure is of an entirely different order.

A modern defence capability is not a product, but a system of systems. Combat vehicles, sensors, command and control, missile systems and logistics all have to work together, in operation, under conditions no one gets to choose. Value emerges only when every part works in concert. Thinking in systems, where the whole behaves differently from the sum of its parts and where a small disturbance can propagate through the entire chain, is the very key in this domain (Meadows 2008). To put the book's central thesis plainly: nothing works until everything works.

8.1 A domain with its own rules

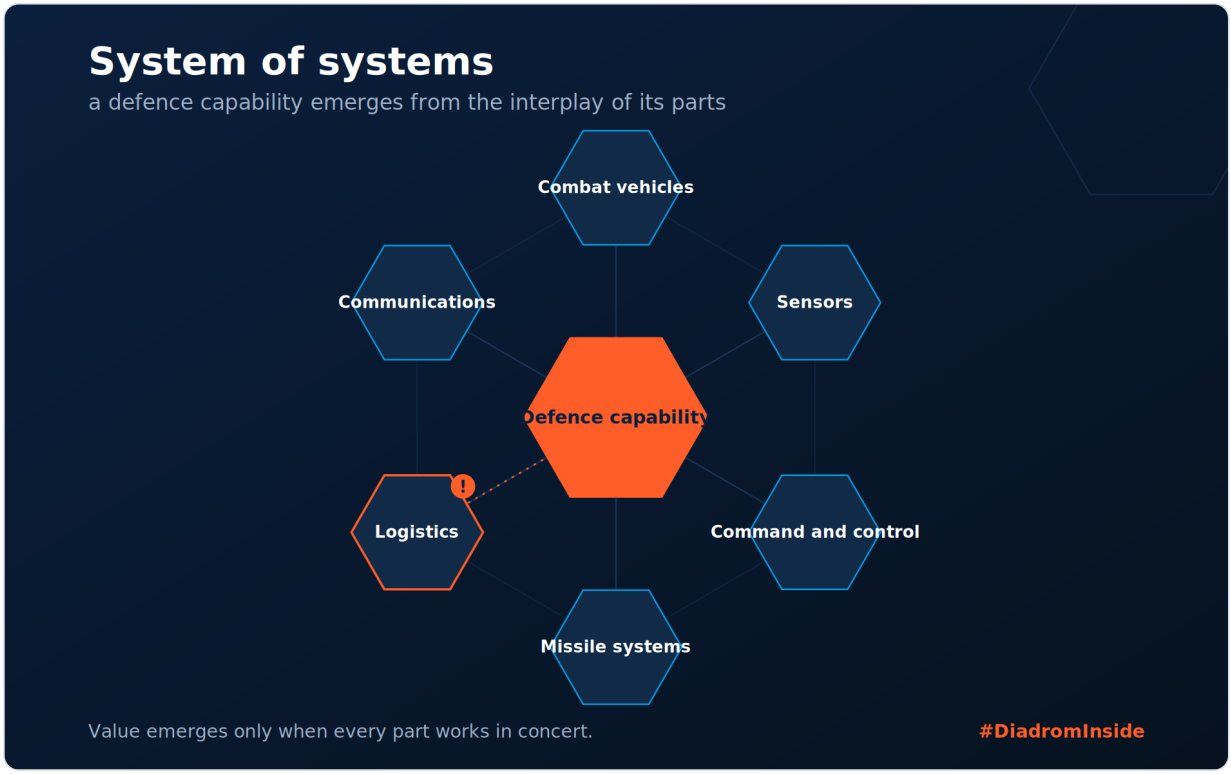


Figure 23. The defence capability as a system of systems.

The civilian aftermarket rests on a quiet assumption: that there is a workshop nearby, a spare-parts inventory replenished daily, and a commercial logic that says you should replace the smallest possible part at the lowest possible cost. That logic is rational when availability can be bought back quickly and cheaply.

The defence and security domain inverts it. Here availability comes before cost. A system that does not work produces no effect, however advanced it is. There is not always a workshop around the corner, not always a daily delivery, and sometimes not even a connection you can rely on. Maintenance has to be possible with a limited inventory, independent of terrain, and in an environment where an adversary is actively trying to disrupt both logistics and communications.

To this add autonomy. More and more systems are expected to operate, monitor themselves and, to some degree, help themselves in the field, periodically disconnected from their central systems. A product that has to fend for itself must also be able to diagnose itself. Diagnostics then moves closer to the sharp end, from the workshop out to the platform.

8.2 The consequences

The consequence is easy to state and hard to live up to: without control of the state of the software, you do not, in practice, have control of the system. However advanced the platform, it is only as dependable as your control over what is actually loaded, configured and verified on board.

The variant problem we described in section 3.6 becomes worse here, not better. Defence materiel often has a service life measured in decades. Over that time, hardware and software are replaced many times, generations are mixed, and the combinations still have to work in the field with whatever happens to be at hand. Keeping track of that population is not an administrative detail. It is the difference between a capability that exists and one that exists only on paper.

It also shapes how you choose to troubleshoot. The manufacturer's classic goal, to isolate the fault at the lowest possible level and replace the smallest possible part, is often the wrong goal here. When availability comes before cost, it can be wiser to swap a larger, well-defined module quickly and get the system back into operation, saving the fine-grained fault isolation for later and for a safer place. It is the same reasoning we set out in section 5.2, with the stakes raised.

And then cybersecurity, to which we devoted chapter 7. In a connected, software-defined capability, secure software download, authentication and signing are not an extra layer on top of diagnostics. They are a precondition for daring to trust an update in the field at all. To this is added the question of sovereignty: who controls the software, where the data lives, and who can guarantee the lifecycle across the system's entire life. For European defence systems, this has become a strategic question in its own right.

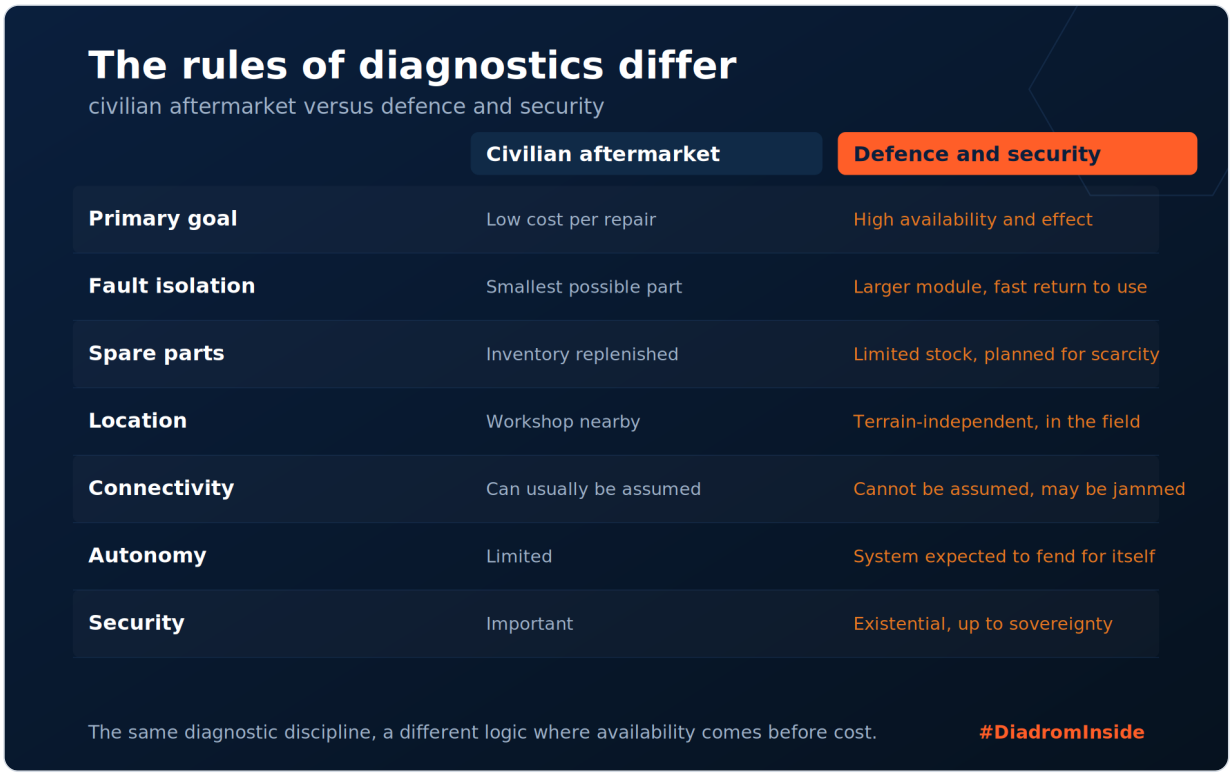


Figure 24. The rules of diagnostics, civilian aftermarket versus defence and security.

Table: The rules of diagnostics differ between the civilian aftermarket and the defence domain. This is one of the book's strongest standalone modules and works well as a separate LinkedIn post.

8.3 The diagnostic discipline applied

The good news is that the toolkit already exists. It is the same one used throughout this book, but it gains new weight in this domain.

The maturity model (section 3.10) is about control over your own population. In defence, that very control, knowing which systems exist, how they are equipped and how they are faring, is often the difference between readiness and uncertainty. The defect-to-system-failure model (section 4.4) describes exactly how a small fault in a single component can travel upward and knock out an entire capability, an uncomfortable but useful way of thinking for anyone building a system of systems. And the idea of customer-perceived quality, that you get only one attempt (section 4.7), translates here into mission-perceived availability: in a real situation there is no second attempt.

Diadrom's own history in the domain shows how long these principles have been applied in practice. Together with Saab, the diagnostic application OBELISK was developed for the RBS 70 NG air-defence system, built with Diadrom's product Diag Studio. One of its tests predicts when the cooling system needs to be refilled with helium, by tracking how the cool-down time changes over time. That is predictive maintenance in its purest form, long before the term became a buzzword. The same logic, reading the state of a system and acting before it fails, is precisely what a defence capability with a limited spare-parts inventory needs.

In Sweden, Diadrom today delivers, among other things, a solution for software management and a so-called flight recorder to the Swedish Defence Materiel Administration (FMV), installed in several Swedish defence capabilities. It makes it possible to trace, analyse and continuously improve the systems' performance. That is off-board diagnostics (section 5.1) set in a defence context: data from the field that becomes better decisions and better systems over time.

On top of this sits the secure foundation. Diadrom's portfolio includes a bootloader (DAB) and a communication stack (DCS) for loading and talking to control units in a controlled way, together with the cybersecurity stack Diadrom Encrypt. Together they form a layer of control and trust for safety-critical and mission-critical platforms, in which every software state is verified, synchronised and traceable.

8.4 Differentiation: #DiadromInside when it matters most

Why Diadrom, and why now? The answer is that the engineering discipline this demands cannot be improvised. It is built over time. Since 1999 Diadrom has solved diagnostics for serial production in the automotive industry, with Volvo as its first customer, and it is the same rigour that now carries over to defence and security. The diagnostic discipline is domain-independent. That is why the journey from vehicles to defence is logical, and not a leap in the dark.

Nor is the timing accidental. Sweden has been a member of NATO since 2024, Europe is rearming, and defence is becoming ever more software-defined. The way commercial technology and fast-moving methods are now reshaping defence at its core is captured well by Shah and Kirchhoff in Unit X (2024), and the same logic applies to diagnostics: the discipline that has grown up in the civilian industry is exactly what defence now needs. In such a setting, control over the software's configuration, traceability and secure updates becomes decisive for operational availability, interoperability and cybersecurity. Diadrom has positioned itself in the middle of this, as a member of the Swedish Security and Defence Industry Association (SOFF), as an exhibitor at Defence Expo Sweden, and with the ambition of establishing itself as an enabler of sovereign, resilient and lifecycle-controlled software for European defence systems.

This is where the ingredient brand #DiadromInside takes on its sharpest meaning. Diadrom's work rarely shows on the surface. It sits inside, as the layer that lets the visible capability actually be trusted when it counts. A platform impresses in a brochure. A platform that works in operation, every time, is something else. The difference between the two is often diagnostics.

And with that we are back where the book began, at the simple but unrelenting truth that runs through every chapter, from the first engine control unit to the most advanced defence system: nothing works until everything works.

9 The manufacturer's aftermarket business

Manufacturers in many industries face ever tougher competition, and in many cases the margin on the product business itself has shrunk year by year. Some even make a loss on the product, with the ambition of recouping it and turning a profit in the aftermarket. Many also sell globally and are exposed to currency movements. To handle the shrinking product margin, virtually all manufacturers work on two fronts at once: developing new aftermarket revenue, and cutting their costs there, not least for warranties. This chapter ties the book's threads to the bottom line, because this is where control and availability finally become business.

9.1 Ways to increase aftermarket revenue

Anyone who has worked in sales knows that upselling is easier than winning new business. Making another deal with an existing customer is considerably easier than landing a new one. The aftermarket is therefore about capitalising on the relationship you already have. Classically there are three routes: selling spare parts, selling accessories and selling services.

Spare parts and accessories. The sale of spare parts and accessories has long been a very important source of revenue. On the Swedish passenger-car market, manufacturers could historically choose their authorised dealers and workshops, and with no real competition from independent workshops, margins on parts could be kept high. That picture has changed, partly through new legislation (section 5.3) and partly because new independent players have appeared. Suppliers may now sell parts of matching quality directly to independent workshops without affecting the warranty (Albrecht 2011), and manufacturers can no longer freely choose which workshops become authorised, but must instead specify requirements, so-called dealer standards. Since an original part and a non-original part are sometimes the very same component from the very same supplier, manufacturers work hard on business development and branding to defend the margin.

Services. More and more manufacturers offer services in addition to the product, and sometimes instead of it. Add-on services have long been used as a sales argument, for example free service for a number of years or operating hours. As we saw in section 2.7, the movement is toward servitization, where the centre of gravity shifts gradually from product to service and outcome. If the customer leases a software-based product and pays for availability and function, the deal moves toward a service platform. If the customer buys the product and upgrades it later, it remains a product. Which it becomes is decided by the business model, not by the technology.

To these three classic routes a fourth has been added, made possible by connected and software-defined products. **Data-driven revenue, feature-on-demand and subscriptions.** When a product

can be updated over the air (section 3.9), features can be sold long after delivery, as one-off purchases, options or subscriptions, and the product can generate revenue across its whole life. The data it produces can moreover become the basis for entirely new services, as we foreshadowed in chapter 6.

There is, however, a lesson here worth taking seriously, and it is about trust. Customers appreciate paying for genuinely new value, but react badly when something they feel they have already paid for is locked behind a fee. Where that line falls is a question of business and trust, not of technology. To this comes the fact that the EU Data Act (section 5.3) opens up the raw data itself, while derived insight and predictions fall outside it. The durable competitive advantage therefore lies not in keeping the data locked away, but in the ability to turn it into services and decisions the customer trusts. That is the same point that runs through chapters 6 and 7, and it is where the diagnostic discipline becomes business.

9.2 Aftermarket costs

FIGURE 25

Revenue and cost in the aftermarket

image to be inserted here

Alongside finding new revenue, manufacturers work to reduce their warranty costs. Warranty costs arise when faults are discovered during the warranty period, and it is not always easy to judge whether a fault is covered. Among the most common problems are that the wrong or an oversized component is replaced, that the customer claims warranty even though the product has been misused (for example heavily tuned), that the technician fails to find the root cause which leads to return visits and lower customer-perceived quality, that complex cases create a long administrative aftermath, and that it is hard to obtain many replacement components quickly during serious quality problems, since supplier lead times can be long.

To cut costs, manufacturers have long worked with the cheapest repair method, efficient case handling, preventive maintenance, symptom-based diagnostics (not spending resources on faults the customer does not experience) and feeding operating data back into product development to improve quality. For acute problems that cannot wait until the next regular service, the manufacturer contacts customers directly, something that normally happens only when safety is judged to be at risk, or, in the automotive industry, for problems linked to emissions legislation. Such recalls often bring negative publicity.

This is where connectivity, data and predictive maintenance make a real difference, because several of the classic warranty problems can be addressed directly.

Warranty problem	How connected data and predictive maintenance help
Wrong or oversized component replaced	Better fault isolation and guided troubleshooting point to the right part
Customer claims warranty despite misuse	Operating data shows how the product has actually been used
Root cause missed, return visits required	Remote diagnostics and history catch the root cause earlier
Long administrative aftermath	Traceable data speeds up assessment of the case
Hard to balance spares during quality problems	Early trends give warning, and recalls can be targeted rather than blanket

The greatest lever, even so, is to move maintenance from unplanned to planned. With predictive maintenance (chapter 6) you can foresee faults before they occur and address them in good time, which reduces both expensive emergency repairs and blanket recalls. Good control of the population, that is, climbing the maturity model (section 3.10), also means a recall can be aimed precisely at the individuals affected rather than at the entire fleet. It should be said that connectivity also brings a new structural cost, namely meeting the cybersecurity requirements (chapter 7), but that is now a precondition for being allowed to sell, not a choice.

From control to business

And with that the circle closes. The book's whole arc has been about control, from knowing which products you can build, through knowing how they are faring in the field, to being able to anticipate and optimise. That control gives availability, availability gives satisfied customers, and satisfied customers give a profitable aftermarket. Diagnostics is the engine of it all. It holds for the passenger car in the workshop, and it holds, with the stakes raised to their utmost, for the defence capability in the field. The same discipline, from vehicles to defence. Nothing works until everything works.

10 References

The literature references from the first edition are retained where they are still cited in the text. To these are added the regulations and standards on which the second edition's new sections rest. Identifiers and links were verified during compilation and should be checked once more before print.

Literature

Albrecht, H-W. (2011). *Innovate Expectations project: Unexpected insights from crafting strategy.* PENTAX Life Care.

Brooks, F. P. (1975). *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley, Reading, MA. (Anniversary edition 1995, with the essay "No Silver Bullet".)

Dahlbom, B. and Mathiassen, L. (1993). *Computers in Context: The Philosophy and Practice of Systems Design.* Blackwell, Oxford.

Furr, N. and Ahlstrom, P. (2011). *Nail It Then Scale It: The Entrepreneur's Guide to Creating and Managing Breakthrough Innovation.* NISI Institute.

Grönroos, C. (1996). *Marknadsföring i tjänsteföretag.* Liber-Hermods.

Kuschel, J. and Ljungberg, F. (2004). Decentralized Remote Diagnostics: A Study of Diagnostics in the Marine Industry. In S. Fincher, P. Markopoulos, D. Moore and R. Ruddle (eds.), *People and Computers XVIII: Design for Life* (pp. 211-226). Springer, London.

Meadows, D. H. (2008). *Thinking in Systems: A Primer.* Ed. D. Wright. Chelsea Green Publishing, White River Junction, VT.

Mullins, J. and Komisar, R. (2009). *Getting to Plan B: Breaking Through to a Better Business Model.* Harvard Business Press, Boston.

Parhami, B. (1997). Defect, fault, error, ... or failure? *IEEE Transactions on Reliability,* 46(4), pp. 450-451.

Pine, J. and Gilmore, J. (1999). *The Experience Economy.* Harvard Business School Press, Boston.

Shah, R. M. and Kirchhoff, C. (2024). *Unit X: How the Pentagon and Silicon Valley Are Transforming the Future of War.* Scribner, New York.

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. *Harvard Business Review,* January-February, pp. 137-146.

Åsberg, J., Billing, A. and Ekelund, A. (2008). Spelet om Scania: en tysk triumf. *Affärsvärlden*, 18 April 2008.

Regulations and standards

California Code of Regulations, Title 13, Section 1968.2. *Malfunction and Diagnostic System Requirements (OBD II).*

UK Department for Transport. *Final Regulatory Impact Assessment: Implementation of emission standards for new heavy-duty vehicles.*

UNECE. *UN Regulation No. 155: Uniform provisions concerning the approval of vehicles with regard to cyber security and cyber security management system.* unece.org

UNECE. *UN Regulation No. 156: Uniform provisions concerning the approval of vehicles with regard to software update and software update management system.* unece.org

ISO/SAE 21434:2021. *Road vehicles: Cybersecurity engineering.*

ISO 24089:2023. *Road vehicles: Software update engineering.*

Regulation (EU) 2024/1257 of the European Parliament and of the Council (Euro 7), on type-approval of motor vehicles with respect to emissions. data.europa.eu/eli/reg/2024/1257/oj

Commission Regulation (EU) No 461/2010 (Motor Vehicle Block Exemption Regulation, MVBER), prolonged to 31 May 2028 by Regulation (EU) 2023/822. data.europa.eu/eli/reg/2010/461/oj

Directive (EU) 2024/1799 of the European Parliament and of the Council, on common rules promoting the repair of goods (right to repair). data.europa.eu/eli/dir/2024/1799/oj

Regulation (EU) 2023/2854 of the European Parliament and of the Council, on harmonised rules on fair access to and use of data (Data Act). Entered into force 11 January 2024, applicable from 12 September 2025. data.europa.eu/eli/reg/2023/2854/oj

European Commission. Guidance on in-vehicle data under the Data Act, published 12 September 2025.

Data and market sources

Figures on carmakers' volumes and rankings for 2024 and 2025 are drawn from manufacturers' published results and independent industry compilations. Specific sources are given in production.

Information on Diadrom's products, customers, projects and reference cases is taken from Diadrom's own material (product sheets, whitepapers and presentations).

Editorial note: all links and identifiers are checked one final time in the production phase.