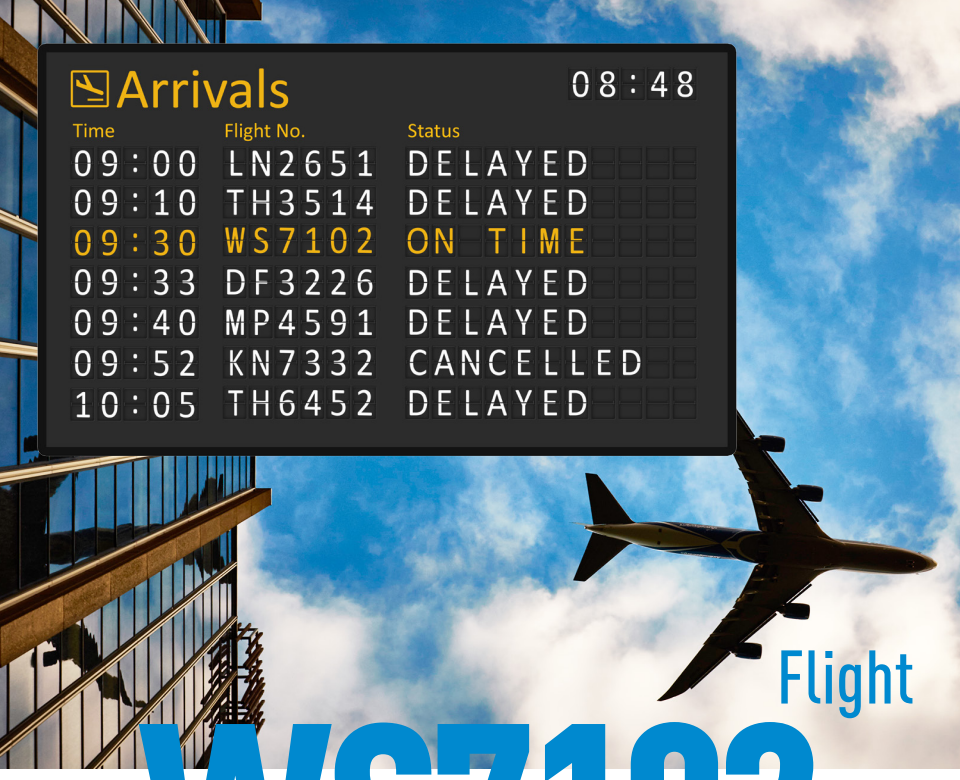


Flight

WS7102

fast forward to web-scale architecture

infoSupport
Solid Innovator


Flight
WS7102
fast forward to web-scale architecture

Publication details

This book was written by members of Info Support's Architecture Competence Center.
They are: Raimond Brookman, Ronald Bosma, Wiljag Denekamp, Wim van Gool, Sander Molenkamp, Rogier Schrama and Edwin van Wijk, all of whom are software architects at Info Support. They received help in writing the text from Freek Jansen at LEWIS Communication Agency and freelance copywriter Martijn Vet.

The illustrations are by Wiljag Denekamp and Edwin van Wijk, and the layout was done by Erika Watt-van de Bovekamp.

All the characters and events in this story are fictional. Any resemblance to real persons, living or dead, or to real events, is purely coincidental.

First edition: October 2016

Preface

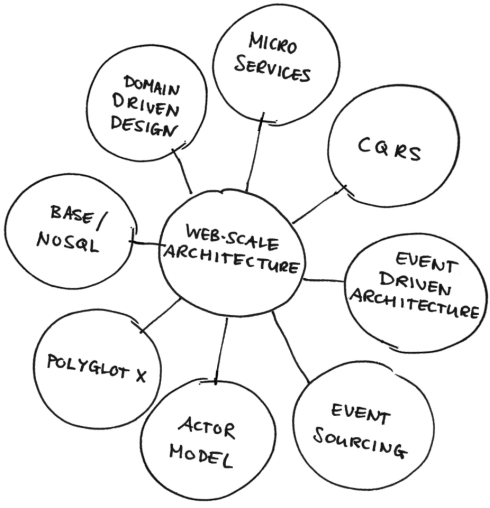
This novel takes us on the journey made by two CIOs, who meet on Flight WS7102.
Peter has just had the toughest week of his career, and is happy to be able to leave his troubles behind him. Isn't it time for him to start looking for a new job? He had wanted to think this over during the flight, but things turn out other than planned when he meets fellow professional Harold.

During the journey, Harold and Peter talk about their experiences in IT architecture. Harold tells Peter what web-scale architecture entails, why he chose to apply it and how Peter's organization could profit from it.

Web-scale IT was introduced a few years ago by Gartner, the world's leading information technology research and advisory company. It is still viewed primarily as a vision of IT, inspired by the practices of Silicon Valley giants such as Google and Facebook. 'Web-scale' in this case means extremely flexible and scalable. This process is facilitated by a web-scale software architecture.

Gartner was less forthcoming about precisely how such an architecture should be realized. At Info Support, we have developed our own vision on this subject: a very specific interpretation of the term 'web-scale architecture'. It's an architecture that we feel is supported by eight subjects that are closely

related to each other: microservices, CQRS, event-driven architecture, event sourcing, actor model, polyglot 'X', BASE / NoSQL and domain-driven design. Most of these subjects and their underlying relationship are discussed in this book.



The idea of this fictional IT novel is loosely based on the 2013 book The Phoenix Project, in which the concept of DevOps is explained. This is not a complete coincidence, since both DevOps (an agile set of practices for team collaboration) and continuous delivery (updating software in short cycles) are perfectly suited to web-scale architecture.

But we will take a closer look at this elsewhere in this book. Flight WS7102 can be seen as a combination of a travel story and a fictional case study. Peter's hospital and Harold's airline are both imaginary, but they could just as well be real.
That's something that became apparent while writing this book; soon after the first brainstorming session, the writers realized that the crisis in the hospital at the beginning of this story had already occurred, almost identically, in the real world.

Whether that hospital also found its solution in web-scale architecture, we don't know, but it could perhaps form the basis for a sequel....

We hope you enjoy reading this book!

1. A 'minor' update

The CIO we're looking for meets challenges head-on, and can respond flexibly to unforeseen situations.

"That's me," Peter de Graaf hears himself saying out loud. Which is strange, since there's no-one there to hear him. He's finally alone for a while. Finally relieved of having to answer to anyone, in contrast to the past week.

Peter's employer, Albert Havik Hospital, had been suffering increasing damage on a daily basis. At a certain point, it was no longer even possible to consult the surgery schedule, never mind schedule new surgeries. The local newspapers had been following the debacle closely for days, and the hospital saw its reputation become more compromised every day.
No, it certainly had not been a pleasant week. It was the final nudge he needed to get him to start looking around for a new challenge.

As he saved the job vacancy to his notes app, he found himself wondering once again: could he have seen it all coming? It had actually been only a minor update; a standard release. Everything had been planned meticulously, and there had been no problems when they went live over the weekend.

The mood in the department had been exuberant,

since a lot of people had been waiting a long time for this release.

First thing on Monday morning too, nothing but happy faces. But by the end of the morning, the first reports that the scheduling module was down were coming in at the service desk, and in no time, the system collapsed like a house of cards. By lunchtime, Peter and the software architect were called to the director's office to explain themselves.

After that, everything happened at lightning speed: while everyone behind the scenes was working hard on a patch, Peter was preparing a press conference with his director and the PR department. The media were bound to get wind of it, so it was better to beat them to it. Soon the team was working at all hours in a frenzied attempt to limit the damage.

In retrospect, the consultants he had hired to get the system back up and running had of course seen it all coming. A new search function in the electronic health record, which is also encrypted, is just asking for overheated servers. I'm quite sure the consultants would have known that.

But then, of course, hindsight is 20/20. There simply hadn't been any time to test that minor update as extensively as they should have. And if they hadn't implemented that handy search function now, they would have had to wait another six months for the

next release. Then Peter would have had to face the wrath of the entire hospital.

Oh well, it's all behind him at this point. One blessing in disguise is that the system was back up and running just before Peter would head off to the Gartner Summit, definitely one of the year's highlights. Time for a bit of light relief. Time to try and forget and in the meantime, start thinking about the future.

2. A big monolith

"Welcome on board, would you like a newspaper?" Peter shakes his head. After the misery of the past days, he can't bear to look at another newspaper. Today's paper will probably say the problems at the hospital have been solved, but the news will no doubt be just a small paragraph somewhere on page seven. The headline *HOSPITAL NEEDS TRANSFUSION*, plastered across the entire width of the front page last Friday still makes him shiver. So it's best to pass on the newspaper for now.

Perhaps the biggest advantage of flying business class is that you get to your seat straightaway. It's a great feeling knowing you can just decompress and switch off for the next couple of hours. With a deep sigh, Peter lowers himself into the comfortable seat.

"Everything all right?" says a voice from the adjoining seat.

"Oh yes, just great!" Did that sound a bit over the top? Peter is just so happy to be sitting down.

"Good. I only asked because of the deep sigh…."

Oh, I'm just relieved to be sitting down, that's all. Oh well… Let me introduce myself. My name is Peter de Graaf."

"Not the Peter de Graaf from..." Peter's neighbor points to the newspaper he's holding in his left hand. Almost simultaneously, they read aloud from a section of a column on page 9:

"According to IT manager Peter de Graaf, the problems with the information system at Albert Havik Hospital have finally been solved."

"Why, yes, the very same. Why do you ask? Are you also in IT?"

"I sure am. Harold ten Kate, CIO at MaxAir. Nice to meet you."

"Ah, a fellow professional. And one who works for an airline. Pleasure to meet you. Well, I'm really glad I'm able to attend the Gartner Summit.

"I can imagine. I'm looking forward to it too. But... I have to say, well done! That's some impressive damage control you did there."

"You don't want to know how many liters of sweat I lost this last week."

"Spare me the details," Harold says with a broad smile. "If you ask me, you were just really unlucky with that update. Do you feel like talking about it or would you rather have a drink first?"

Well, I wouldn't say no to a Johnnie Walker...

Peter launches into his story. "Honestly, we never saw this coming. Everyone was so happy to see that update finally implemented. Then it turns out that a new fuzzy search function has introduced a bug in the query functionality of the electronic health record, which brought all processes to a standstill."

"I imagine all your other systems are dependent on that EHR?"

"That's right. Every department has to be able to consult the patient card and enter changes relating to the treatment. If that functionality fails, their hands are tied."

"I see. So actually, you introduced a central dependency, a single point of failure, into your landscape with that EHR."

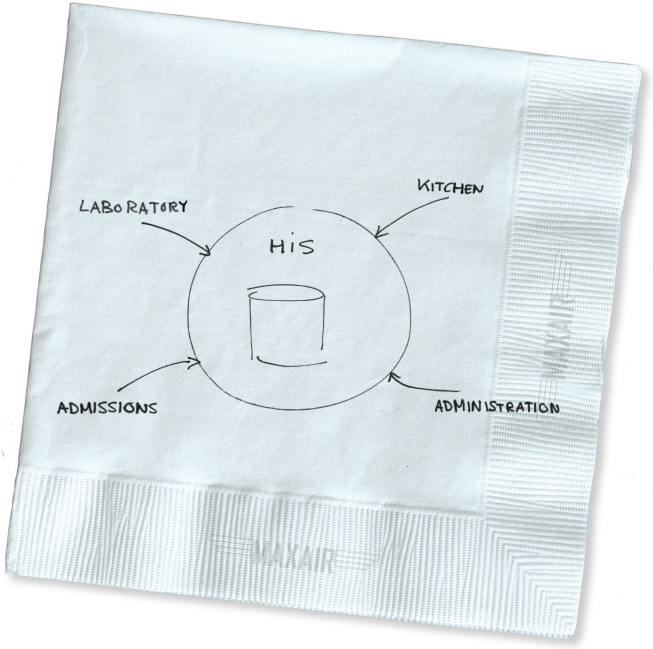
"A single system for all departments in your company is a corporate risk"

"Maybe. But it's not as though we didn't carefully consider before choosing this EHR. There was an extensive selection process beforehand. This system allows us to cover the functionality for many departments in one go. And we're pleased with that, as such, because

it's a huge improvement on the old situation."

"I bet you used to have a lot of customized software?"

Peter nods. "Exactly. And all sorts of legal requirements and standard expense processes are covered, and actually, we'd rather not spend our IT capacity on that. In that sense, as a hospital, our back-office processes are not unique. But there's a downside to it as well: we're now stuck with the twice a year release cycle of that supplier.
And if the supplier finds our requirements too specific, they're not always willing to incorporate them.
See, this is roughly what our Hospital Information System looks like."



Even though he's reminded of all the problems of the past week, Peter is feeling more relaxed than he has in quite a while. Obviously, the whisky helps, but meeting a fellow professional who actually wants to hear his story makes him feel almost happy.
He can't think of a single colleague at the hospital who is as empathetic as Harold.

"I totally get it," replies Harold. "Our airline also makes use of large ERP applications that we're very satisfied with. But we have to ensure that bugs in those applications don't undermine our entire landscape. And we've now found a really great solution for that."

"Oh?"

Harold settles himself for his story. "Two years ago, we were in exactly the same situation as you are now. When the ERP application was down, we couldn't even manage to let people check in. All the passenger information was recorded in that one ERP application based on the well-known architecture principle that you record the information once, and use it multiple times. We had to do everything manually, even things that didn't seem to have anything to do with that application. Then it turned out that a lot of specific applications from departments were linked through services, and therefore stopped working when that application was briefly unavailable. Our architects searched for a way to make the rest of the landscape independent at runtime from that ERP application; to

keep things running smoothly when something else is unavailable for some period of time. We've found a good solution: microservices architecture. It allows autonomous, relatively small services to communicate with each other by means of messages."

3. Patience is a virtue

"Cabin crew, boarding completed."

This is going to be a long flight, but Peter's not worried about being bored. They haven't even taken off yet, and he's already heard so many interesting things that the journey is already worthwhile.
He quickly checks his messages.
Dear Peter, Groningen University Hospital read how you solved things and would really like to meet you. Informal interview next week, is that okay?

Groningen, that would mean relocating. But an informal interview can't hurt. *Absolutely! Back on Monday!* Peter sends the message before switching his phone to flight mode.

So, basically, Harold used to work precisely the same way at MaxAir as the Albert Havik Hospital currently works.

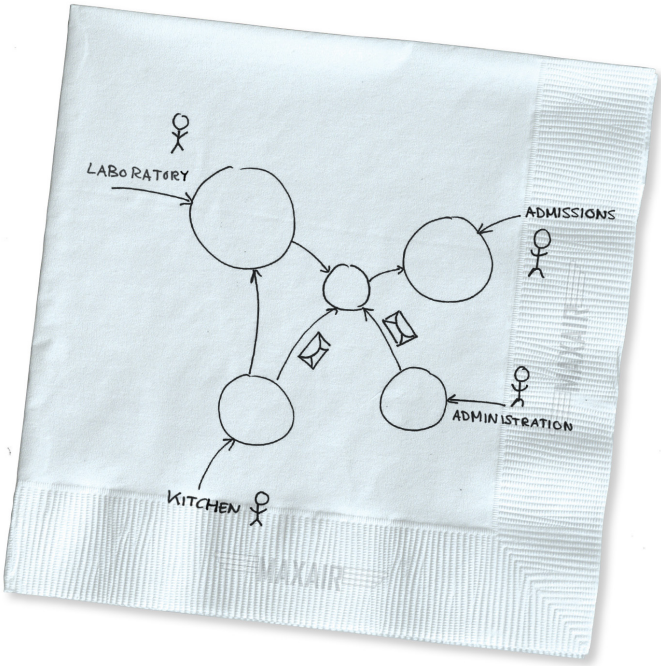
"But hang on," Peter says. "Autonomous functioning services..., doesn't that mean you have to deal with data duplication? If there's one thing our software architects despise... Well, let's just say we've had some less than pleasant experiences with a lot of stand-alone databases that came into existence in the client-server era. We made the switch to an ERP system precisely to solve all the hassle of 'differing truths'

in various systems."

"Yes, been there," Harold replies. "You don't want all sorts of different applications administering the same information with differing truths about a passenger or a patient floating around in your system. But there are other, possibly better solutions than reducing everything to a single monolith, such as, for instance, determining ownership of data. It's where you agree that everyone can store a copy of the information, but that only one system is responsible for changing that information. If that one system fails, for whatever reason, such as our ERP or your EHR, the other applications can keep running on their own copy of the data."

"Right, so that way you can define multiple systems, with their own logic and data collection, and even design them to perfectly suit the department using them."

"Exactly! Now, let me see... For your situation, it would look something like this." Harold takes a pen and starts sketching enthusiastically.



Peter is thinking out loud. "That would certainly help us to tailor those specialized systems to the specific needs of that department. Departments often complain about the lead time for changes, because we are obliged to prioritize new functions in the system hospital-wide.
But can't that cause problems, if not every department is looking at exactly the same data? What if changes to the data in the central system are not directly applied to the data in the department's system? That department is then looking at outdated data."
Apparently, Harold was expecting that question.

"That's true – you do indeed have to take into account a certain degree of temporary inconsistency. The question is whether that's such a bad thing. I mean, does it really make such a big difference if a change in a patient's information arrives a second or two later in another system?"

"Availability of systems is often more important than being able to access the latest version of the data everywhere at any time"

"Hmm, not always. A patient's change of address is not directly important to the treatment. In fact, the department systems don't really need that kind of information at all. But there is one really important component for us: the patient card. This card contains information such as which medication a person has been given and any allergies they may have. For example, if a patient comes into the emergency department, is treated and then goes to the OR, it's important that the OR has access to up-to-date information."

"I can well imagine," Harold says with a laugh. "That's pretty crucial information. But how do you currently deal with that? What do you do if that information isn't available because of a failing EHR?"

"In the case of this kind of crucial data, we adhere to the principle that there should always be a paper copy

with the patient, containing the most recent information. We don't always have time to update the system anyway. Often we don't get around to that until after the operation."

"That's what I mean. In cases like that, you need procedures you can always fall back on. It's the same with disconnected systems. Ultimately, everything can fail at some point. The advantage of separate systems is that often you can still extract information from one system while the other one is down, such as consulting patient information in the OR if the EHR is temporarily unavailable. The most recent updates are on paper, but the rest of the information is accessible. And remember, as soon as all the systems are up and running again, any messages still get processed. They call that *eventual consistency*. You choose to make the availability of information more important than the guaranteed consistency at every point in time."

Peter is starting to understand. "Okay, so if we take a good look at the company processes and the transfer moments between departments, it seems to me we could work perfectly from a model based on eventual consistency, and so we could surely apply that kind of microservices architecture. The EHR remains in existence, but thanks to the services and the exchange of messages, it is fully disconnected from the department systems.
Mind you, I do see another challenge: how do we get that many small, different, connected services te-

sted and into production? It was enough of a drama just getting the EHR operational and that's only one thing."

Harold leans forward. "Ah, and this is where it gets really interesting!"

4. Don't be afraid to think small

Just thinking about the amount of time and effort it took to get that EHR up and running made Peter feel anxious again. Big releases, slow updates; they're nerve-wracking. How does Harold manage with all those microservices?

Harold resolutely empties his glass and starts to explain. "With a monolith, all the work is concentrated around the big release moments. And the supplier often determines more or less just when such a release is deployed. The test effort in particular is usually huge. And systems often need an entire weekend of maintenance, people need to be stand-by... You know the drill.
The idea behind microservices is that not only you can operate each service separately, but you can also develop them and take them into production separately. You have tiny bits of functionality, in which testing activities, risk and impact are all limited to that small part of your system. That also means that those little bits of functionality can go live much more often than was the case with big releases up to now, which means that you can offer added value to your organization much faster."

"Ah yes, continuous delivery, right? I read something about that on the Summit site."

"That's the one. We'll be hearing that term more than once in the next few days, I imagine."

Peter remembers something else he saw on the site, and has a question about it. "But I thought that was only for the big tech companies, the Googles and the Amazons. I mean, how do you organize something like that?
It all sounds wonderful, but how are you going to take those services into production separately, while, in terms of functionality, they are all interdependent?"

"I'm not saying it's easy. For example, services have to be programmed in such a way that the links to other services keep working when new versions are taken in production. But that trick is certainly not reserved for the big tech companies. You will, however, have to invest in the knowledge and skills of your IT staff to implement it well."

"I get that. It actually all sounds so logical. When I get an update on my mobile phone or laptop, I expect that all the other apps I have installed will keep working. But, as you know, money is always an issue, and the IT organization is already costing the hospital a pretty penny. The introduction of the EHR was very hard and expensive. I wouldn't know how to sell that to the MT and the Board. That would be like me saying: that last investment didn't achieve the results we'd hoped for. I need more time and more money."

"That's always a justified concern," Harold agrees. "But let me put it another way: how important is the IT department to your hospital these days?"

Peter knows what he means. The IT department is, of course, essential. He has personally experienced what happens if the systems no longer function properly. Before you know it, the hospital gets a reputation for being unreliable, and in a hospital, people's lives are at stake.

"Of course, IT is essential," Peter replies. "That's why we invest so heavily in it. But it's still an expense on the budget, and the budget has its limits."

"It's interesting you call it an expense. Tell me, how does your hospital distinguish itself from others in the region?"

"Well, now you mention it... Besides the fact that we provide good care, obviously, we also try to improve our services. For example, by restricting waiting times, limiting errors in administration, providing a good customer experience and taking a personal approach. Unfortunately, with this catastrophe surrounding the EHR, we haven't been all that successful recently..."

"So, as I understand it, IT isn't only important in supporting the functioning of the hospital, it also plays an important role in the edge it gives you with your staff, the health insurance companies and the patients.

Doesn't that make the IT department more of a revenue-generating department than an expense?"

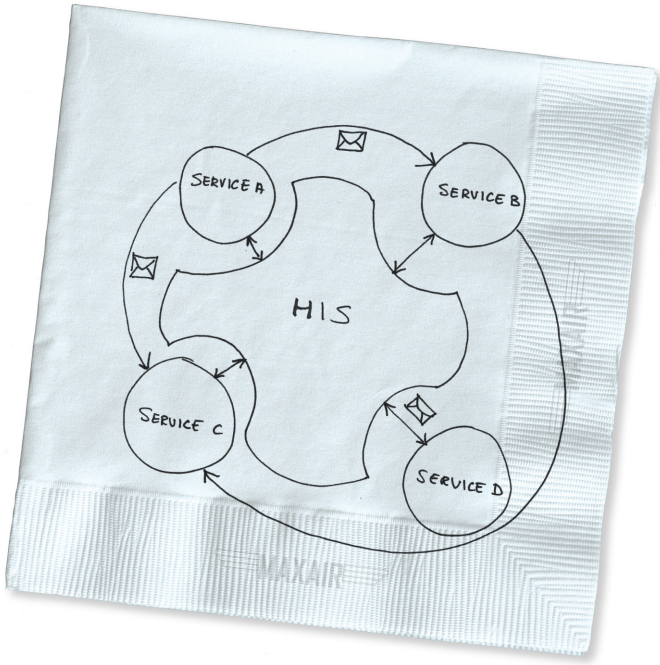
Peter nods thoughtfully. "Hmm, as a CIO myself, I hadn't thought of it that way before... I've always had to elicit investments through the Board by coming up with all kinds of doomsday scenarios, or describing acute problems that needed solving, but they would naturally be more inclined to invest if they saw the opportunities for us in that area!"

Harold laughs. "It's all a matter of perception. It's not only about solving problems; your organization will be a fair bit more agile if your IT systems are more flexible.
Once you realize that, investing in the knowledge and skills of your staff is a no-brainer. And remember that the new way of working in your IT department will ultimately be just as normal as what you're used to now. Obviously, you have to go through a pretty steep learning curve first, but eventually, you will reap the benefits."

Peter has to admit that the idea of those microservices sounds like a welcome solution. And yet something's still bugging him. He can't stop thinking about the huge electronic health record that was delivered only five months ago. He'd even organized a dinner party for the whole IT department to celebrate the milestone. So that can stay, but how does that all work together then? And won't it drive the users mad?

He can already see complaining nurses, doctors and colleagues from the information counter standing in front of his desk again.

Not only does the CIO of the airline know everything he needs to, he also appears to be able to read Peter's mind. "We had one large system too. A year and a half ago, we started constructing a shell around that system, with services that unlock the application. Our ERP system is still in use, but we have added more and more services. You can also just keep working with the EHR's user interface."



"Using microservices provides the organization with flexibility and agility"

"Okay, so with microservices you have less chance of other functions and other departments running into difficulties, but the problem of the search functionality still hasn't been solved."

"We'll get around to that later."

5. Reading and writing

As often as he's seen the inside of an airplane, Peter still gets a somewhat queasy feeling at the moment the aircraft leaves the ground. It's that split second when the wheels leave the tarmac and the plane always seems to sway a little.

"Are you all right there, neighbor?"

"Oh yes, sorry about that." Peter has no idea why he's apologizing. "I'm okay with flying, except for the take-off..."

"You know what helps?" Harold says, and gestures to one of the cabin crew to see if he can order something. Not until the seat-belt sign is turned off.

"The same actually goes for your planning system too, that above all, it's important to be able to view the appointments, right? So that appointments can be kept and nobody needs to be sent away?"

Peter nods. "That's more important than the additions. Although we obviously also have to be able to schedule appointments."

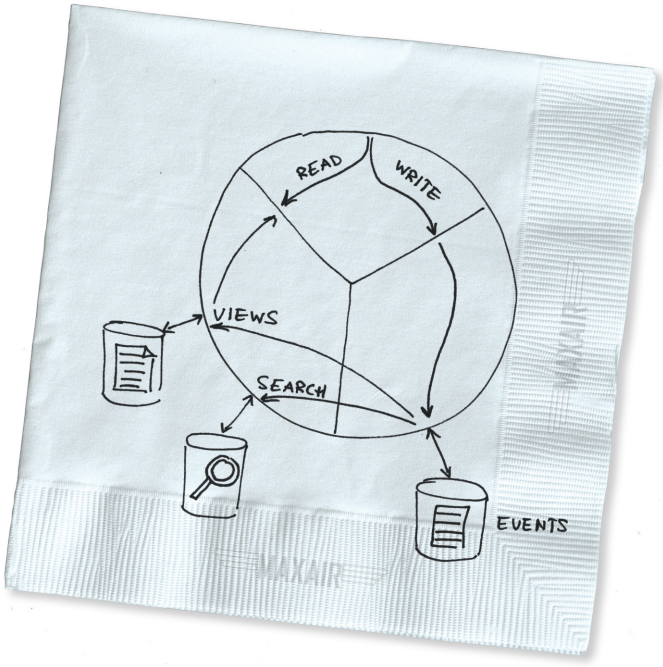
Harold is really getting into his stride again. "So what if you design the functionality around planning as two separate services? Then you can create one microser-

vice that is responsible for registering appointments, and another one for viewing appointments. Those two services are both part of the planning domain, but have different objectives. That pattern is also known as CQRS, which stands for Command Query Responsibility Segregation. Basically, that means separating the reading side from the writing side. In your case, the service responsible for registering appointments is the writing side, and the service for viewing the appointments is the reading side.

The great thing about it is that the system is no longer as vulnerable. If the writing side is unavailable for a while, for example due to an error during the implementation of a system update, the reading side is still available, and staff can still view the appointments. Nobody needs to be turned away."

Peter has to admit that it sounds extremely appealing. "But, if those two microservices are completely separate, how do they stay in sync? There has to be some sort of link, right?"

Again, Harold looks like he expected the question. "The microservice that deals with registering appointments always contains the truth. We call that the 'system of record' or 'single point of truth'. From the writing side, we can publish messages, based on which the reading side can update its data. Every time a new appointment is scheduled, a message can be sent, for example, containing information relevant to the



appointment. The read side can use the information in the message to update its own database. So the reading side you consult is completely separate from the writing side."

And still there's that nagging little voice in Peter's head, and again it's talking about data duplication. "So you're saying that there can be some lapse of time between the scheduling and synchronization with the reading side? If I schedule an appointment, shouldn't that be immediately visible to the staff?"

"It depends on your definition of 'immediately'," Harold answers. "Say it takes five or ten seconds longer until the reading side is updated. Is that a problem?"

"No, not really. When you put it like that, it's not so bad. Appointments are always scheduled at least a day in advance."

"That is often the case," Harold agrees. "This is one of those situations in which you are dealing with eventual consistency. The advantage is that it increases the availability of the individual services."

"Using CQRS increases the availability of systems"

"Okay, you've convinced me that this could work for our planning system. But it seems to me that it wouldn't always be possible."

"Not always," says Harold. "There are some situations in which you simply must have transactions in order to keep two or more systems synchronized. It's always a consideration. In the end, it's about what's more important in a particular situation: availability or consistency. And ultimately, this decision will be determined by what provides the most value for the business. In your situation, availability is clearly more important. Appointments could be registered on paper, so to speak, if the writing side is down. It's not ideal, but it is

an exceptional situation."
Just as the aircraft appears to have straightened out, and the seat-belt sign should go off at any minute, the intercom crackles on: "Unfortunately, we're not yet able to pass through the cabin. We're experiencing an unusual amount of turbulence at the moment. We ask you to remain seated with your seat-belt fastened."

Fear of flying is irrational, Peter thinks to himself. And turbulence is a challenge.

6. Time flies

Peter hadn't really needed that Johnnie Walker to help him sleep like a log throughout most of the flight. I must have been exhausted, he thinks to himself. He can see the lights of Las Vegas through the airplane porthole.

Just before the aircraft seems about to land, the engines suddenly surge. The nose is pulled up and the plane veers off to the right, away from the landing strip.

"Hey, a go-around!" Harold's voice betrays a certain degree of excitement. "Probably because of the strong side wind."

"Good evening, ladies and gentlemen. This is your captain speaking. As you may have noticed..."

Peter rolls his eyes. "One more time around the desert then." He's not sure what's making him more impatient: the discomfort of the airplane, the prospect of a hotel room with a bed or the prospect of possibly receiving a message about a potential new job.

Fortunately, the go-around works and the second attempt at landing is successful. Apparently, Peter is not the only one who is visibly relieved by this, as he hears a round of applause erupt from the economy class. Peter would usually associate that with a cheap

charter flight, but today he can see himself joining in thankfully. He's never experienced turbulence as bad as on this flight. His neighbor had tried his best to reassure him: an aircraft is not a monolith, and the chances of crashing are smaller than those of winning the national lottery, but even Harold had looked a bit worried now and then.

Just then, Harold takes out his mobile phone. "Look, this is our newest app. We launched it just last year. In the flight information section, you can see that we've just landed. To achieve that, we constructed a number of services that also use CQRS. If you look here, you can see that there's been a go-around."

"But, wait a minute – you don't want that kind of information being available to your customers, do you?" Peter sounds surprised.

"No, you're right, it's not relevant for everyone. I can access it because I use an in-house version of the app that is only for staff. This version consults other read models than the consumer's version."

"Cool! Hey, can we turn off flight mode yet?"

"We've got Wi-Fi on board, didn't you know?"

Peter could kick himself. He selects the Wi-Fi channel and while he's waiting for a number of messages to download from his service, Peter downloads the

MaxAir app. "Well I never, you're right! On my phone, I see that we've landed, with a twenty-minute delay."

"Exactly!" Harold agrees enthusiastically. "And don't forget that those events have led to many other events. The fact that we had a go-around probably means that, at this busy time, other aircraft have had to circle around, and the approach planning in the air traffic control software is being automatically updated. And when that leads to flight delays, that gets displayed on the flight information boards at the airport. All caused by a few events."

"And everything is ultimately 'eventually consistent' again?"

"That's right. The original events, such as 'Aircraft landed' and 'Aircraft going around' are stored in a microservice that represents the writing side. Other services form the reading side, and update the displays that you are now viewing in the app."

"And how do you store this sort of event in the writing side?" Peter asks. "Is that simply a relational database?"

"That's certainly one possibility. But we decided to use event sourcing. That's an alternative way of storing the sequence of events that has led to a specific state; not in a relational database, but as a list of events that have occurred in time. Since we append all events to

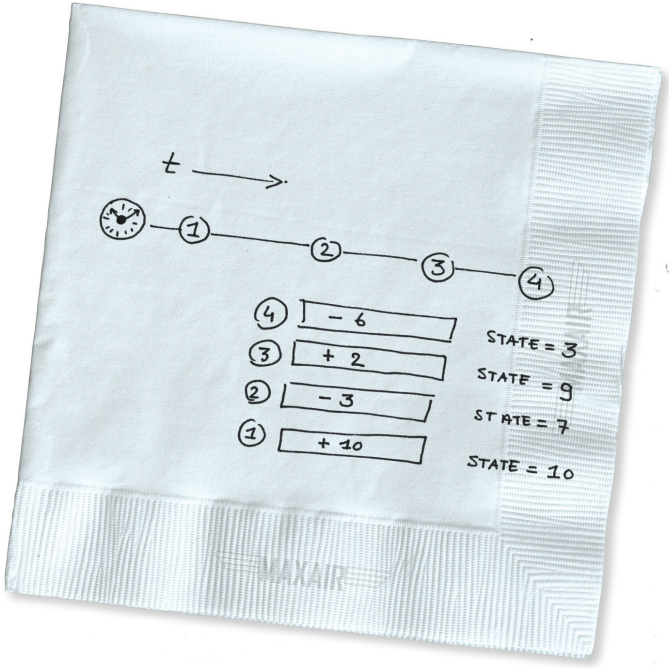
the list, updates are implemented faster than when using the traditional approach, where we have to lock data when we update it in order to prevent concurrency problems."

Now Peter's a bit confused. "Wait a minute, that sounds complicated. Aren't you basically recording the aircraft's status? Wouldn't it be easier to make a status field in the database, where you can place an update as new events come in?"

"On the contrary. The problem with updating databases is that you destroy information. If we were to change the information in the database, based on incoming events, we would only be maintaining the 'current state'. Then we would only know the most recent status of an aircraft. That wasn't enough for our system. We want to be able to pull up the complete status history of the aircraft, or in other words, all the events that have taken place. That way, we can see how much time there was between the go-around and the ultimate landing, for example."

Peter thinks about this for a moment. "That sounds a bit like an audit log."

"Absolutely! That's an automatic side-effect of event sourcing. And it isn't necessary to record the current situation with a separate audit log next to it, describing how that current situation occurred. Event sourcing fulfills both needs."



"But that would seem to me to be difficult to query then. For example, how can I easily find out which aircraft have a 'Landed' status?" Peter wonders.

As usual, Harold has the answer. "That's exactly what we use the read models for. Because we've disconnected the responsibilities for reading and writing, it's unnecessary to support an extensive query functionality in the event store, where all those events are stored. That makes the design of an event store simpler than that of a complete database. In addition, we can trace the 'current state' of an aircraft by reading the sequen-

ce of events from the beginning. Even better, we can also determine the status of an aircraft at any given time, simply by viewing all the events that occurred until then."

"With event sourcing, the dimension of time becomes an explicit concept in the software"

Peter wonders why MaxAir would want to save all this information. "Are you doing all this purely for auditing?"

"Sometimes you just need to review specific information," says Harold. "For us, for example, it's really useful to know how many people who book a flight also select their seat. Do they click directly on the place they reserve? Or do they click on various seats and come back to their first choice, when it turns out that extra legroom is much more expensive? If you want to be able to carry out that kind of analysis, you need the history of events."

"But what if it turns out that a wrong event has occurred?" Peter is pleased to note that despite the long flight, he's still on his toes. "What if a customer later changes the seat they selected when booking their flight? Then you would have to go through the entire sequence of events to trace the relevant one and change it. That's an awful lot of work."

"That's the great thing about event sourcing; you can't change events. Ever. That's equivalent to falsifying history. The events took place, that's a fact. But new events that effect a correction, they're allowed. You might compare it to the way accountants work. You do know that accountants never use a pencil and eraser? You must be able to prove where a transaction came from, at all times. Instead of disposing of an event, you add a compensating event."

No falsifying of history. Peter is really enthusiastic about this: this way, all the changes in the scheduling system would be traceable.

He checks his mail: yet another message from a potential employer. This time with an actual job offer, even better than the one in Groningen. And yet, there's a sudden doubt; now he's not so sure he's ready to leave his current employer. Maybe there's still too much to be done...

7. If the shoe fits...

"Passengers from flight WS7102, please proceed to luggage belt 42."

Excited by his conversation with Harold, Peter can't resist asking another question. "So, that system knows we've landed and where we can pick up our luggage. Is all that information retrieved from the same system as the app on my phone uses?"

"No. On the contrary: the system you see here is not an AirMax system at all, it belongs to McCarran International Airport. This airport does, however, use a mainframe; the system is updated based on the events that we, as an airline, publish, and so it stays up to date."

"And is there an event store behind it?"

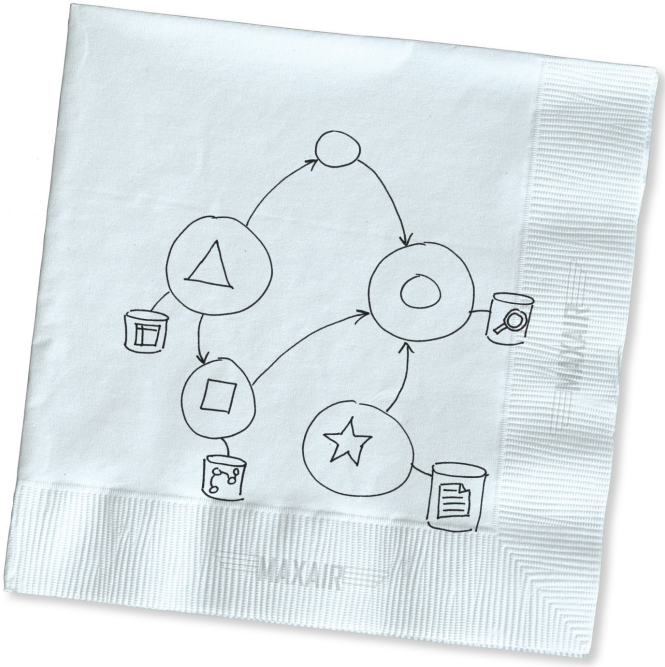
"No again," Harold says with a wink. "And there doesn't have to be. The airport is only interested in the most recent state of affairs based on the events we publish. If I'm not mistaken, there's a relational database behind it. But that's not necessarily always the best choice. For example, in our app, we offer the possibility of linking with other users. We use a graph database to record this, rather than a relational database. In a graph database, you can easily record different relationships without having to introduce all

kinds of junction-tables, as you would normally have to do in a relational database."

"I can see what you mean. So you can use different types of database, depending on the functionality you want to offer the users?"

"Exactly. Another example is search functionality. Often, a search engine is used that is good at indexing data and enabling complex search tasks. And that applies not only to storage, but also to your architecture and the technology stack you're using. It has to suit the solution you want to offer too. For example, for our microservices, we chose Java, but our front-end was built with ASP.NET, because our front-end team is at its most productive with this. So instead of enforcing that all the teams within the organization use a single standard, each team chooses what works best for them and for the desired solution. We call that Polyglot 'X'."

"A team that can make use of fit-for-purpose tools and frameworks is able to perform at its best"



This is music to Peter's ears. "So you no longer need to train all the teams if you want to introduce a new technology? Brilliant! But how do you prevent an explosion of various tech stacks and frameworks?"

"You want to give teams room to choose the best tools for the solution, but obviously within certain frameworks and guidelines. The application architecture and management aspects also have to be considered."

Freedom to choose. It sounds like a bit of a revolution

to Peter; no more haggling between Java and .NET developers. Each team chooses its own language, just as long as the microservices can communicate with each other.

He finds himself having to keep himself from being totally carried away by Harold's story. Would this approach be able to prevent any future crisis with the planning system? The picture looks right in his mind, and it all seems perfectly logical, but would it work in the hospital? First, I'll discuss it with my team, he reassures himself.
One thing is sure, though: Peter is no longer dreading that first team meeting after Las Vegas; he's actually looking forward to it!

8. Positive results

"Hey, Harold! Has the pain gone down a bit?"

"Yes, thanks. And just as well. It was unbearable."

"I know how it feels. I've had a broken leg. Though mind you, mine was from a skiing holiday, not the moving walkway at Schiphol Airport..."

"And despite the nice lady's warning: mind your step... That's the last time I ever go on one of those stupid things."

The unfortunate end to the Las Vegas trip has obviously soured Harold's mood. "How long will you be in a cast, do you think?"

"The doctor was just here: the results of the tests are positive, so I think I should be up and about soon."

"That depends on the kind of tests. In a hospital, positive test results aren't always a good thing..."

"Ha! No, I mean positive in the sense of favorable."

"Occupational hazard," Peter says, laughing. "You wouldn't believe how easy it is to confuse terms in the medical world. That even goes for something as simple as 'left' and 'right'; are you looking at it from

the patient's point of view, or your own? If you don't introduce stringent rules to cover it, before you know it, the wrong limb has been amputated."

"In my field of work, that's much the same, of course." Harold sits up slowly. "We've struggled hard with that in the past. Initially, when we only had a couple of microservices, it all seemed quite transparent. But as more and more departments became involved, confusion of terms between developers and system users also increased. We realized that we would have to make better agreements about the terminology to be used and so we started applying DDD, domain-driven design, in our projects."

Just look at us, Peter thinks. It's only been five minutes, and we're back to discussing our work again. Talk about occupational hazard...

"One important element of DDD is the setting up of a universal language. We call that the 'ubiquitous language'," Harold continues. "It makes sure that developers, users and domain experts can all understand each other properly. This language is even used in the programming code, as a way of preventing errors in the software."

"Using DDD ensures that domain experts, developers and end-users can understand each other better"

Peter is skeptical; Harold's explanation only raises more questions. "We tried to set up such a universal language for the hospital a few years ago. At that time, it was known as an enterprise-wide canonical data model. The project didn't succeed though. The model simply became too complex. The financial software, in particular, requires completely different information on a patient than the software that schedules surgeries, which in turn is different from the information used by the hospital's pharmacy. Integrating all these differences is pretty much impossible."

"A canonical data model, like the one you're describing, would be great, but in our experience, it just isn't feasible in a large organization," Harold agrees.

"Larger organizations tend to consist of islands with their own responsibility, their own definitions and their own jargon. In DDD, we call these islands bounded contexts, and each bounded context has its own ubiquitous language. It means that the financial department can continue to work with its own definition of a patient."

"I'm not sure I'm following you completely. I thought the whole point was that departments could exchange information?"

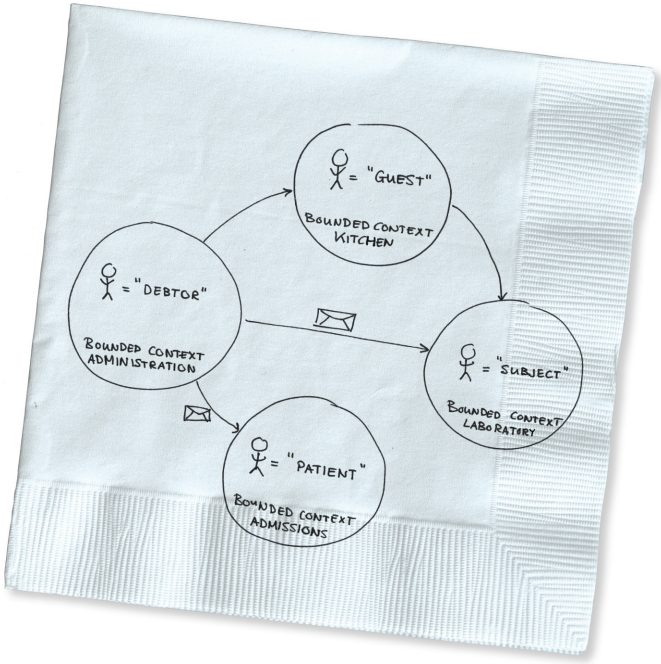
"That's right, but that means we only have to agree upon the data to be exchanged between the various bounded contexts. That's often much less data than

is actually used within the bounded context. So you could use a number of generic terms that have been agreed upon company-wide. I imagine that in your hospital, you make use of a patient number, and that everyone uses that same patient number, which allows everyone to uniquely identify a patient."

Peter is starting to understand. "But how do I know which bounded contexts there are? Do I have to go brainstorming about it with domain experts and then continue to fill them in?"

"Yes, that's right," confirms Harold. "In order to discover those contexts, and decide what really belongs together, we apply event storming. Using brown paper and sticky notes, we quickly chart which processes we have to support, together with department staff. We also define which relevant events or business events occur within these processes. When you've arrived at a clear view of these dependencies, it's relatively easy to discover elements that have a strong relationship. And then we place them together in a bounded context."

As interesting as it all sounds, Peter can't hide the fact that he's still a bit skeptical. "So after just one of these event storming sessions, you've charted your complete domain with all the dependencies and interactions? I have to say that sounds a bit too good to be true. How do you ever get everyone involved to agree to that whole model? You just told me that a canonical data model was a bad idea, but this seems very much



like a variation of one, but further complicated by the inclusion of the processes."

"Maybe I didn't explain it properly," Harold says with a laugh. "The idea behind all those independent domains is precisely that they will also change independently of each other. That's why a model doesn't have to be agreed upon company-wide. The teams themselves can compose the content of the domains, together with the domain experts. Then you will have to make translations of terms between the bounded contexts, from one context to the other, but that will generally only be necessary for a small percentage of

all terms. Components that collaborate intensively will, after all, have been placed within the same bounded context."

"Yes, that does sound great," Peter answers. "It would seem that DDD offers you a whole lot of tools, with which to deal with the various domains within your system landscape. But on the other hand, it feels a bit like overkill for some situations. I think complete bounded contexts could be filled in by standard applications in our hospital."

Harold continues: "You certainly don't need to apply it in detail everywhere. In fact, you can decide for each bounded context whether to apply the DDD principles there or not. And quite often, you'll find it's only necessary for a few core activities, in which you want to be distinctive, as an organization. Our airline, for example, has chosen to be competitive in its cost price. So it's really important that our aircraft can carry as many passengers as possible, and be on the ground for maintenance as little time as possible. We built a smart planning service ourselves, for the logistics behind that, because that's a crucial functionality for us, and we couldn't find it anywhere on the market."

"So, as I understand it, you create independence by cutting your landscape up into bounded contexts and applying microservices within them. And that gives you flexibility, which enables you, for example, to locally adjust previous choices, if and when required.

Whether that's because you came across a standard application that can replace your current customized software, or because one of your company processes has changed due to an internal reorganization."

"That's right. So you have the option of changing the content of bounded contexts," Harold continues. "This is also recommended by the best practices of DDD. As soon as a model begins to deviate from the reality of the company process, you just need to implement the necessary changes. Even if they are quite rigorous. Since the services within the bounded contexts are mutually independent, such adjustments are simpler to implement. In DDD terms, that's also known as supple design."

Peter scratches his head thoughtfully, letting Harold's last words sink in. Harold keeps coming up with good ideas. "Oh dear, what time is it, actually?" Peter suddenly realizes they've been talking for some time and he's due in an important meeting with the Board.

"Harold, despite the fact that you're lying in our hospital with a fracture, you've done me a greater service in this short time than we have done you. I'm afraid I have to rush off now. If you need anything else from me, let me know and I'll take care of it. And make sure you're up and about very soon!"

"I'm sure I will be. And don't worry, Peter, they're taking really good care of me here. I hope to speak to

you again soon. I'll contact you via WhatsApp."

With a firm handshake and a sincere smile, Peter walks out of the ward toward the boardroom. As he walks along the corridors, all the new ideas and concepts he has heard from Harold, and at Gartner, are filling his head. Immediately after the conference, Peter had succeeded in convincing the Board of the importance, and even necessity, of using web-scale architecture to cope with the problems occurring within the IT landscape. At that time, he also charged his architecture team with the task of immersing themselves in the new concepts, and tentatively applying a few changes. In a process of ups and downs, which Harold had warned about, the teams had started to understand it, and achieve more insight. It didn't take long before the planning department delivered the most stable software, which also turned out to be the most productive. After that, other teams naturally became curious, and in the meantime, the entire IT landscape within the hospital is being constructed from small, autonomous services and there have been no more problems with total blackouts.

Peter is thinking back on all that as he puts his hand on the door of the boardroom. These meetings used to be something he dreaded, but not today. Not anymore. Now these meetings are all about delivering good news, and securing investments. He takes a deep breath, feels a smile coming on, and steps confidently into the boardroom.

9. A couple of tips

What a year! Hammering the last tent peg into the ground, Peter realizes that it's been nine months since the crisis at the hospital. And just after that, the trip to Las Vegas...

You might call this a well-deserved holiday. First, the search function had been patched, and then the whole Hospital Information System had been made more agile, a bit at a time. And along the way he had turned down one great job offer after another. No, he's been able to do a good job for his current employer, and he's even been given the necessary recognition for it. And anyway, there's still more than enough to do in the coming year.

But now it's time for two weeks of relaxation, at a simple campsite, near a small French village where he's sure not to encounter any Dutch CIOs.

"You really don't have to check your mail, you know," the hospital CEO had assured him. And he was right, of course. They had an excellent team, and should one of the system functions go down, at least the whole thing wouldn't collapse.

Not checking your mail, why is that so difficult?! I mean, Wi-Fi has penetrated even the most densely-wooded areas of the French countryside these

days. Oh well, the tent is up, the children are enjoying themselves in the playground, so why not?

The mailbox has little of interest to offer. 'Malfunction patient registering system solved' – good. 'Vacancy for head of ICT Ministry of Security' – delete. Hey, a message from Harold.

"Peter, everything okay? I'm doing fine. You have to read this!"

Harold has attached a newspaper clipping to the email: 'Not such a super day for supermarket chain.' The article quotes the CIO of TopMarkt Netherlands as saying that the problems with the group's information system have finally been solved. A minor update had resulted in a failure to replenish stocks. The upshot: angry customers, frustrated supermarket staff and distraught managers.

"Poor guy, I know what you've been going through," Peter thinks as he pours himself a beer.

"Good afternoon! Sorry to disturb you." Really? Another Dutch person at this campsite? Peter is taken aback.

"Hallo." He greets his fellow countryman, who has apparently parked his car a couple of spaces further along.

"It's a bit of a strange request, I know. I want to set up my tent, but I seem to have left my mallet at home. I left in a bit of a hurry. Could I possibly borrow yours?"

"Of course! And let me introduce myself: My name is Peter de Graaf."

"Hi, I'm Lieuwe van der Woude. Oh, that's great, thank you."

This can't be true. Peter has just seen that name somewhere. He unlocks his tablet, where the article about the problems at TopMarkt is still open.
"Not the Lieuwe van der Woude from..."

"Why, yes, the very same. Why do you ask? Are you also in IT?"

"Yes, I'm even a CIO too, at Albert Havik Hospital. Listen, there's no rush to put that tent up.
Care for a beer? I might just have a couple of tips for you."

Flight WS7102

fast forward to web-scale architecture

"Let me introduce myself. Peter de Graaf."

"Not the Peter de Graaf from..." Peter's neighbor points to the newspaper he's holding in his left hand. Almost simultaneously, they read aloud from a section of a column on page 9:

"According to IT manager Peter de Graaf, the problems with the information system at Albert Havik Hospital have finally been solved."

What are the odds? You've just experienced the low point of your career and spend the entire flight to a conference in Las Vegas seated next to a fellow CIO. During the trip, the two of them explore various concepts of web-scale architecture: from microservices, CQRS and event sourcing to domain-driven design. What do these terms mean, why would you choose to apply these concepts and what kind of things do you have to take into consideration? For Peter, Flight WS7102 turned out to be a real journey of discovery.

InfoSupport
Solid Innovator