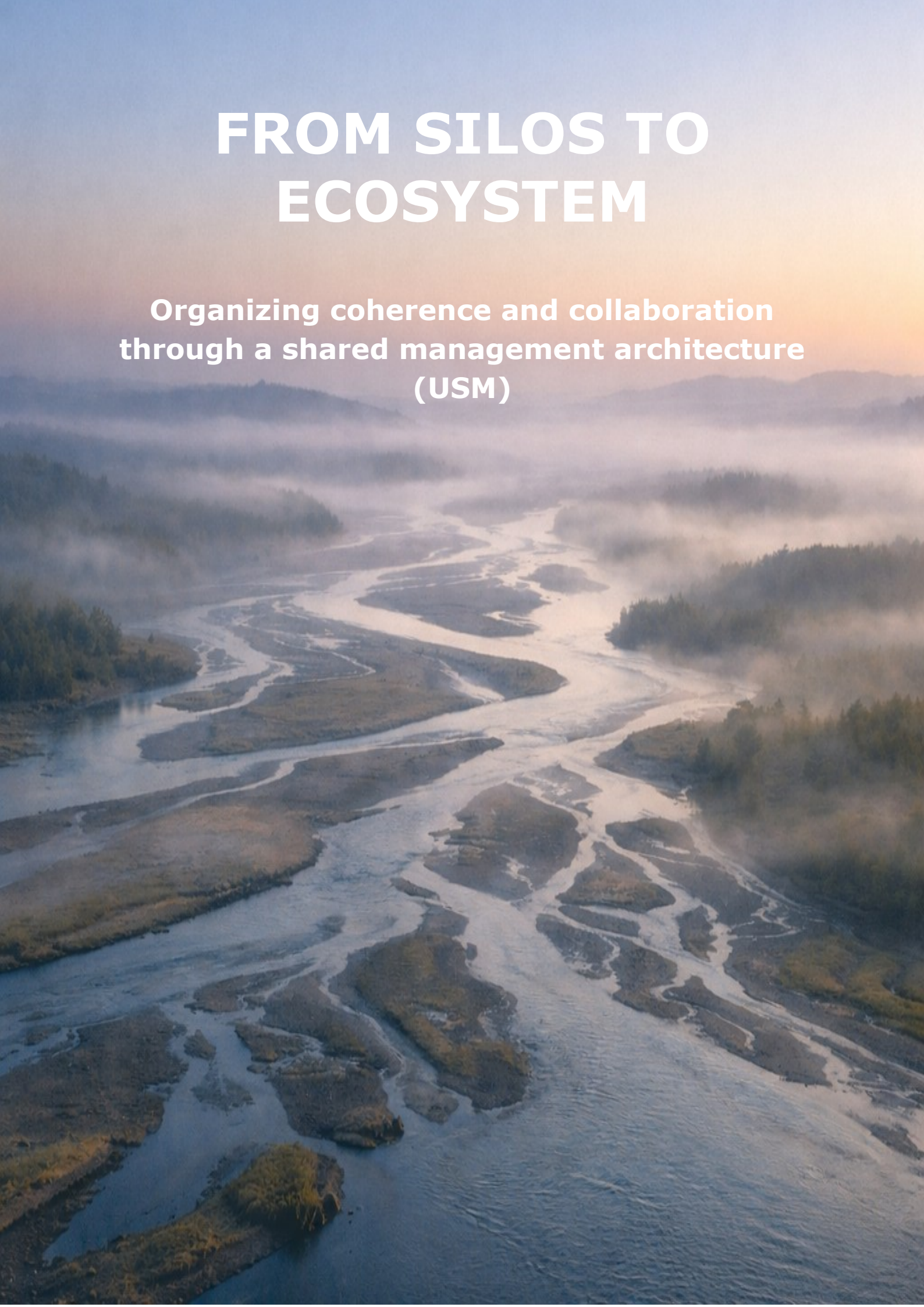

FROM SILOS TO ECOSYSTEM

Organizing coherence and collaboration through a shared management architecture (USM)

Colophon

Title	**From Silos To Ecosystem** *Organizing coherence and collaboration through a shared management architecture (USM)*
Author	Jan van Bon
Publisher	SURVUZ Foundation
Concept & methodology	Unified Service Management (USM)
Version	First edition — 2026 (20260205)

Websites	www.usm-portal.com; www.usmwiki.org

Foreword

This book did not originate on a whiteboard.
It originated in meeting rooms where everyone was right—and yet the system still failed. In organizations where people worked hard, reported honestly, managed properly... and where service delivery nevertheless remained difficult. As if all the cogs were turning perfectly, but never in the same clockwork.

For a long time, we explained this with words like complexity, change, leadership, and human behavior. Nice words. True words, too. But they were too often used as an excuse not to look further. Not to acknowledge that we had trapped ourselves in structures that once seemed logical, but now mainly got in each other's way.

We built silos and then tried to connect them. We divided responsibilities and then endlessly coordinated them. We introduced tools to create overview — and often discovered that they produced a different kind of confusion.

USM did not arise as a response to a theoretical problem, but from a practical necessity. As an attempt to regain control of cohesion. Not by adding more processes, but by reducing them to their essence. Not by steering harder, but by organizing more clearly where steering makes sense — and where it does not.

Yet this book is not about USM. At least, not only.

It is about maturity. About the moment when an organization stops optimizing parts and starts understanding the whole. About the courage to recognize that not everything can be engineered — but that you remain responsible for how the system behaves. Systems Thinking is the starting point.

The metaphor of river flows is no coincidence. Rivers do not ask permission to come together. They do not meet to discuss governance. They follow a shared logic: gravity, direction, cohesion. Anyone who tries to manage a river by optimizing each stream separately will end up with floods. Those who understand the river basin will find peace, predictability, and resilience.

This book is written for people who seek that peace.
Not the quick fix. Not the next framework. But overview. Understanding. Direction.

It is not a manual and it is not a final judgment. It is an invitation to take a fresh look. At your organization. At your role. At what you may have seen all along but did not name.

If this book triggers something—doubt, recognition, resistance—then it has done its job.

Coherence rarely begins with certainty. Usually with an uncomfortable but honest question.

This is my attempt to ask that question out loud.

Jan van Bon

Chair of the SURVUZ Foundation

Content

Reader – How to use this book

This book is not a manual or a description of a method.
It is a learning path that you can follow yourself.
You can read it linearly, but you don't have to.

If you are a manager

Read the book from cover to cover.

Not to memorize it, but to recognize where you:

- coordinate instead of manage
- compensate instead of design
- and confuse peace with risk.

If you are a professional or team leader

Read Parts I–IV to understand why your work is more difficult than necessary.

Read Part VI to see what changes when the system finally cooperates.

Use the reflection questions not to provide answers, but to open up conversations.

If you are an architect, consultant, or change agent

Read this book not as a solution, but as a deliberate limitation.

It shows what you should not fill in, what you should not decide on, and why designing is different from improving.

Be sure to use the Countervoice and the chapters on power and role distribution.

About the Countervoice

The Countervoice is not a counter argument. The Countervoice represents familiar, reasonable objections — not resistance, but habit. It is the voice of the reader who still thinks in the old paradigm. Do not skip it. Recognize it. And then read on.

About Rivertown

Throughout the book, we follow a case study: the municipality of Rivertown. This is a fictional municipality, but one that struggles with the usual problems of government organizations: fragmentation, which creates silos (islands) within the organization, resulting in a lack of cohesion and cooperation.

Concepts

This book uses the term 'service'. This term can be read as 'service' throughout.

Where 'he' is used to refer to a person, it can also be read as 'she'.

Finally

This book does not work if you 'apply' it. It works if you let it sink in.
Not in documents. Not in tools. But in decisions.

If, after reading this book, you make one decision differently than you would have otherwise, it has done its job.

"This book does not accuse people.

It describes patterns that every person would exhibit in this system.."

Management Summary

Reason

Many organizations struggle with persistent silos, difficult collaboration, and—as an inevitable consequence—unpredictable service delivery. Despite years of investment in leadership, culture, reference models, ITIL, Agile, and other best practices, fragmentation persists.

The dominant explanation is people-oriented: lack of collaboration, ownership, or leadership.

This book offers a different diagnosis: *the cause lies not with people, but with the design of the management system.*

The core of the problem

Most organizations lack an explicit management architecture. As a result, they also lack a coherent management system. This creates a pattern in which:

- services span multiple teams but are managed *by team*
- coherence is not designed, but compensated for personally
- managers mainly coordinate rather than manage
- exceptions and escalations actually control the system.

From this perspective, silos (islands) are not a failure, but a logical consequence of local optimization within a poorly designed whole.

Why existing solutions do not work

Frameworks and best practices (such as ITIL, Agile, Lean) and reference models (such as GEMMA) describe practices: how work can be organized locally and independently. They describe interesting fragments, but not how the whole can be managed.

When organizations stack these practices without a shared architecture, the result is:

- more variation in definitions
- more overlap in responsibilities
- more coordination
- and ultimately more complexity.

Doing more within the existing paradigm tends to exacerbate the problem.

The USM approach

The Unified Service Management (USM) method is not based on practices, but on a structural approach to controlling the whole.

The core consists of three design choices:

1. **Services as chain performance**
   A service is not a team, application, or product, but a supported facility that only delivers value to the end user when viewed in context.
2. **One Service Management Architecture (SMA)**
   The SMA acts as the steering wheel of the organization: one shared management principle for all services, regardless of domain, team, or supplier.
3. **The 1–5–8 logic**
   - 1 explicit service definition for *all* services
   - 5 universal management processes for *all* activities, non-redundant and integrated
   - 8 workflow patterns that allow *all* chains to flow in the same way.

This is not a choice of method, but a minimum condition for sustained manageability.

What does this mean for managers?

USM does not require working harder, but working *differently*.

Separation of duties has proven to be the most powerful control tool we know for managing chaos. The role of managers is shifting fundamentally:
- from coordination to system responsibility
- from operational adjustment to methodical steering
- from heroism to architecture.

Management is becoming a profession again: designing, monitoring, and improving cohesion.

Coordination does not disappear, but becomes an executive role within a clear system — no longer a personal burden for managers.

What does this mean for teams?

For teams, USM does not mean extra pressure, but rather relief:
- less coordination
- clearer responsibilities
- less compensation for system errors
- more room for craftsmanship.

Collaboration becomes easier because the system facilitates it.

Implementation: small but fundamental

USM is not 'deployed'. It is adopted as a perspective and applied to local conditions.

The minimal sequence that preserves architectural integrity consists of seven steps:
1. Choose USM as the direction in which you want to improve and determine the scope of a pilot or proof-of-concept (PoC).
2. Get to know the USM concepts and way of thinking.
3. Define the playing field for the chosen scope.
4. Define your services. Start with one visible service within the scope. Design it entirely according to USM, using the Service Model Canvas.
5. Define the routines for delivery and support based on USM templates.
6. Set up the tooling for handling calls and regular activities.
7. Achieve your improvement goals step by step using your management system.

The following applies:
- Protect the architecture.
- Learn through experience.
- Only scale up when it is stable.

External support is possible, but only in the role of a coach: not decisive, not executive, but reflective and limiting. Learning for yourself is paramount.

Conclusion

Silos do not disappear by changing people.
They disappear when organizations are finally designed as systems.

This book invites us to make a choice: do we continue to coordinate people — or do we start managing organizations?

Short-term pragmatism is not the same as professional manageability.

What seems pragmatic today undermines governability tomorrow. Professional leadership sticks to the design, especially when deviation is easiest.

PART I – WHY IT GETS STUCK (AND WHY THAT MAKES SENSE)

This book does not begin with solutions.

That is deliberate.

Too often, organizations try to solve problems by working harder, intensifying cooperation, or adding yet another layer of communication. When that fails, it is rarely because people are unwilling, but because the system in which they work rewards and reinforces their behavior.

In Part I, we therefore first take an unvarnished look at what is really happening.
Not at intentions, but at effects.
Not at individuals, but at patterns.

You will recognize how well-intentioned choices systematically lead to:

- increasing coordination
- expanding consultation
- declining overview
- and a growing dependence on 'indispensable' individuals.

This part is confrontational. At times uncomfortable. And necessary.

Because only when it becomes clear why things are getting stuck can it later become clear what needs to be fundamentally changed.

Diagnosis must precede design.

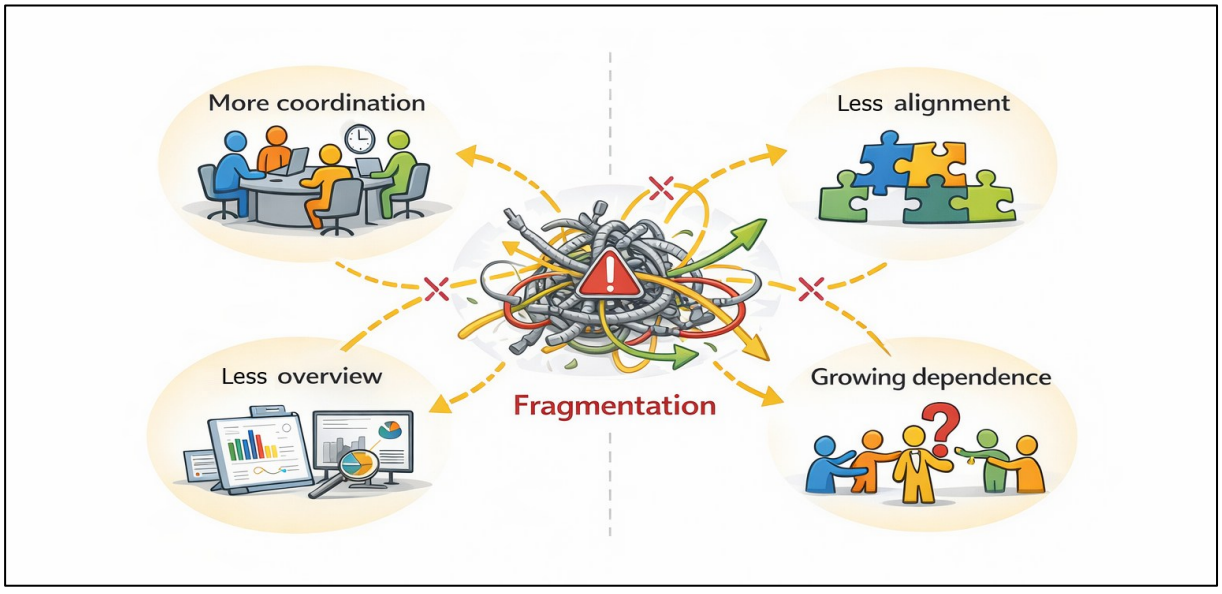

Figure 1. Chaos is not resolved by coordination. It is resolved by structure.

1 The daily reality of the fragmented organization

1.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, the week invariably begins with the same ritual. The municipal secretary receives reports of stalled files. The director of operations schedules additional coordination between teams. Team leaders join the meeting to "see where things are stuck." No one doubts the commitment of the staff. Yet everyone feels that it is becoming increasingly difficult to get the work done.

For many managers, this picture is immediately recognizable. Not one major failure, but a continuous series of small disruptions. A request that remains unresolved because it "falls between two formally defined responsibilities". A change that is taken up, but later has to be re-coordinated because another domain does not recognize itself in the chosen solution. Consultations that are not intended to make decisions, but merely to prevent work from stalling.

All of this is commonly labeled as 'complexity'. Like an inevitable consequence of scale, legislation, politics, and social expectations. And so it is accepted as "simply the reality of a modern organization — something to cope with, rather than to question."

But that is precisely where the problem begins.

1.2 What managers experience on a daily basis

What managers in fragmented organizations experience is not a lack of activity. Quite the opposite. There is constant action, coordination, and adjustment. A persistent sense of busyness dominates daily work.

What is missing is something else: calmness in the system.

That calmness would be visible if:
- work flows naturally
- responsibilities are clear
- and problems did not keep recurring in the same places.

Instead, managers increasingly see themselves acting as:
- liaisons
- escalation points
- temporary solutions.

They bridge differences between teams. They fix mistakes that arise during transfers. They make exceptions to "smooth things over." That feels like commitment, sometimes even leadership.

But in reality, it is compensatory behavior.

The Countervoice
"Isn't that, traditionally, what managers are expected to do? To make sure things keep running smoothly?"

That is only true as long as the system is not designed to run smoothly.

1.3 Silos are not incidents

Silos are described as something that 'arises'. As if they are the unwanted by-product of growth or specialization. As if they form like weeds in an otherwise healthy landscape.

But silos are not a coincidence. They are predictable.

As soon as an organization:
- defines responsibilities per domain
- measures performance locally
- and implicitly allows for cohesion…

a structure emerges in which each part acts rationally *within its own boundaries*. That is not wrong behavior. It is logical behavior within a limited framework.

The problem is that *the whole* is not explicitly managed anywhere.

What then emerges is a situation in which:
- everyone is right *within their own context*
- but no one is responsible for the overall result.

That is the essence of fragmentation.

1.4 Why good intentions exacerbate the problem

One of the most insidious aspects of silos is that they are combated with good intentions. Managers see that cooperation is faltering and respond with initiatives that seem sensible in themselves:
- additional consultation structures
- coordinating roles
- chain consultations
- steering committees.

Each of these interventions has a logical reason. And each of these interventions has the same effect: they hide the design problem by temporarily compensating for it.

This does not improve the system. It makes it more dependent on people.

The Countervoice

"But without that extra coordination, things really get stuck."

Exactly. And that is the signal that the system itself is failing.

1.5 The human toll of fragmentation

Fragmentation has not only organizational consequences, but also human ones.

In fragmented organizations, we see that:
- professionals no longer recognize their work in the end result
- teams become defensive about their own responsibilities
- and managers become structurally overburdened.

People do not become exhausted by work itself, but by pointless remedial work. By mistakes that have to be resolved over and over again. By decisions that do not hold up because they are not anchored elsewhere.

This forces cynicism:
"It doesn't matter what we agree on."
"The system always wins."

And with that, fragmentation becomes normalized.

1.6 Key message

Silos are not a cultural problem, a behavioral problem, or a capacity problem.

Silos are the predictable result of a management system in which the whole is not explicitly designed and managed.

As long as that does not change, any improvement will remain local and temporary.

What emerges is a paradoxical situation: the organization is constantly active, yet structurally ineffective. Energy is spent keeping work together, not on improving how the work is organized. This is not accidental behavior — it is system behavior.

1.7 Reflection questions

- Where in your organization do people have to compensate structurally to get the work done?
- Which consultations exist primarily to fill gaps in responsibilities?
- Who is explicitly responsible for the whole — and who is not?

1.8 Learning objectives

Having read this chapter, the reader will be able to:

- recognize why silos are not incidents but structural patterns
- explain why good intentions reinforce fragmentation
- distinguish between doing work and system compensation.

1.9 Preview

In the next chapter, we will deepen this insight.
We will show that silos are not a choice made by people, but emergent behavior of a poorly designed system.
This definitively shifts the question from “who is not collaborating?” to “what have we designed in such a way that this is logical?”

2 What silos really are (*islitis* dissected)

2.1 Context & recognition

> 🧭 **Rivertown moment**
> Within the Social Domain of Rivertown, there is a weekly case consultation.
> Within Environmental Management, there is also a weekly case consultation.
> Both consultations are carefully organized, well prepared, and led by experienced professionals. When cases overlap, they are "briefly coordinated." No one doubts the intentions. Nevertheless, the cohesion remains fragile and dependent on personal relationships – cooperation, not collaboration.

When silos are discussed, the conversation in many organizations almost automatically turns to behavior. About *people* who do not contribute ideas, do not cooperate as expected, or "think too much from their own silo."

That seems logical. After all, the behavior is visible. But visibility is not proof of cause.

2.2 The misconception: silos as a behavioral problem

The idea that silos are primarily a behavioral problem is persistent. It forces interventions such as:

- collaboration training
- cultural programs
- leadership programs
- brainstorming sessions about 'the intention'.

These interventions can have a temporary effect. People begin to understand each other better, speak the same language, and show more willingness to contribute ideas. But remarkably, the same problems return over time.
Not because people "fall back," but because *the system* remains unchanged.

Behavior always adapts to the context in which it occurs. When that context makes fragmentation logical, silo behavior is not a deviation but rational action.

> 💬⚡ **The Countervoice**
> *"But some teams really don't cooperate, do they?"*
>
> That may be true. But the question is: "What makes this non-cooperation rational here?"

2.3 Silos as emergent system behavior

USM does not approach silos as a characteristic of people or structures, but as emergent behavior. That is to say: behavior that arises *naturally* from the underlying structure of the system.

Silos arise when:

- responsibilities are defined locally
- goals differ per domain
- success is measured locally
- and coherence is not explicitly assigned anywhere.

In such a context, it is logical that people:

- give priority to their own domain
- keep risks at bay
- and limit responsibility to what is formally expected of them.

This behavior is not selfish.
It is professional within a limited framework.

The problem is not with the people, but with the lack of an overarching management principle.

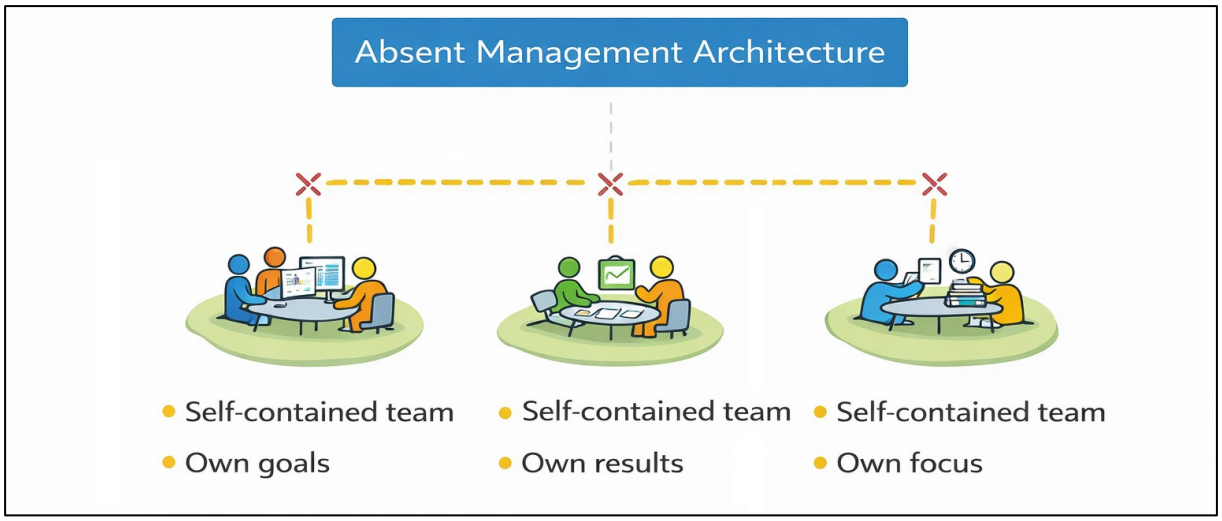

Figure 2. Silos (islands) are emergent system behavior – especially in the absence of that system

2.4 Local optimization and the watermelon effect

A classic symptom of *islitis* is the watermelon effect: reports are green, while the experience in the chain is red.

Each team can demonstrate that it:
- achieves its goals
- has its processes in order
- and works according to agreement.

Yet the customer experiences delays, repetition, and errors.

This is not a paradox. It is the direct result of local optimization without chain control. When performance is measured only within domain boundaries, there is no incentive to optimize the whole.

In fact, optimizing the whole can even be seen as a risk locally.

> 🧭 **Rivertown moment**
> In Rivertown, every team achieves its KPIs.
> Yet the turnaround time for chain services remains unpredictable.
> Problems do not arise within teams, but during transfers.
> And it is precisely these transfers that fall outside everyone's explicit responsibility.

2.5 Consultation as a symptom, not a solution

A common response to *islitis* is consultation. More consultation, more coordination, shorter lines of communication.

Consultation is not a problem in itself. It becomes problematic when it is structurally necessary to restore coherence that was not designed.

When:
- decisions have to be discussed over and over again
- exceptions become the rule
- and progress depends on who knows whom...

consultation is no longer a tool, but a band-aid.

The Countervoice
"But without consultation, we really can't get it done."

That is not an argument in favor of consultation.
It is a diagnosis of design failure.

2.6 Why *islitis* is persistent

Islitis is difficult to break through because it reinforces itself.

As fragmentation increases:
- the need for coordination grows
- new roles and consultation structures emerge
- and the system becomes increasingly dependent on specific people.

Successful coordination masks the problem. As long as it "still just works," there is little urgency to question the design.

Only when:
- key figures leave
- the workload becomes structurally too high
- or external pressure increases...

does the fragility of the system become apparent.

***More coordination** is human compensation for a lack of architecture.*
***Management** is taking responsibility for cohesion.*
***Architecture** is the design that makes coordination unnecessary.*

Coordination does not decrease because people stop coordinating, but because the system finally does what coordination was once needed for.

2.7 Key message

Silos are not a choice, an attitude, or a cultural problem.
Silos are emergent behavior of a management system that does not explicitly organize cohesion.

As long as that system does not change, silos will continue to reappear — regardless of good intentions or extra effort.

Throughout this book, a distinction is made between *cooperation* and *collaboration*.
- **Cooperation** refers to people helping each other across boundaries, often to compensate for missing structure.
- **Collaboration** refers to people working together within shared interfaces toward a shared outcome.

Cooperation depends on effort. Collaboration depends on design.

2.8 Reflection questions
- Where in your organization is consultation needed to restore missing cohesion?
- Which performances are "green" while the chain fails?
- Which responsibilities exist mainly by virtue of personal commitment?

2.9 Learning objectives

Having read this chapter, the reader will be able to:
- explain why silos are not a behavioral problem
- apply the concept of emergent system behavior to their own organization
- recognize how local optimization causes chain failure.

2.10 Looking ahead

In the next chapter, we shift our perspective to the outside.
We show how *islitis* translates into a loss of customer value, and why organizations can appear manageable internally while becoming unreliable externally.

3 The impact on customers, patients, and citizens

3.1 Context & recognition

🧭 Rivertown moment
A resident of Rivertown moves within the municipal boundaries. For him, it is a single event.
For the organization, it is a series of separate actions: change of address in the Citizen Database, adjustment of waste collection, change of parking permit, possibly a reassessment in the Social Domain.
The resident calls after two weeks to ask where the confirmation is. The answer is: "That's with another team".

This moment is no exception. It is exemplary of how fragmented organizations have structured their services.

What is logical internally becomes incomprehensible externally.

3.2 The illusion of internal control

Within organizations, services are assessed on the basis of internal criteria:
- Have the processes been followed?
- Have the systems been updated?
- Have the agreements been fulfilled?

When these questions can be answered positively, the feeling arises that "everything is going well."
But for the customer, patient, or citizen, that is not the yardstick.

The outside world does not experience teams, processes, or systems. It experiences:
- turnaround time
- predictability
- clarity
- and reliability.

The tension arises when internal control is confused with external value.

💬 ⚡ The Countervoice
"But if everyone does their job well, shouldn't everything work out fine?"

That would only be true if the whole is managed somewhere.

3.3 Chain value as a hidden dimension

Services that span multiple teams are, by definition, a chain performance. This applies to:
- a request for benefit
- a permit
- a care process
- a move
- or a complex customer request.

In a chain, value is not created within a single link, but between links. That is precisely where transfers take place, decisions are translated, and errors occur.

When chain performance is not explicitly managed:
- responsibility shifts
- problems are passed on
- and a system is created in which no one has an overview of the overall result.

🧭 Rivertown moment
In Rivertown, each team is assessed on its own KPIs. The turnaround time of chain services falls outside this. As a result, no one is formally responsible for the citizen's experience as a whole.

3.4 Why customers suffer from internal logic

From an internal perspective, silos are defensible. They provide overview, specialization, and clarity of responsibilities. But the same logic is destructive externally.

For the customer, fragmentation means:
- repetition of information
- unclear status
- contradictory communication
- and unpredictable lead times.

What is perceived as 'complex' internally is perceived as 'unreliable' externally.

💬 ⚡ The Countervoice
"But citizens understand that we are organized in a complex way, don't they?"

Perhaps. But understanding does not replace service – and it certainly does not justify failure.

3.5 The vicious circle of recovery and mistrust

When service provision falters, a familiar reaction follows:
- complaints increase
- pressure on the organization grows
- ad hoc solutions are put in place.

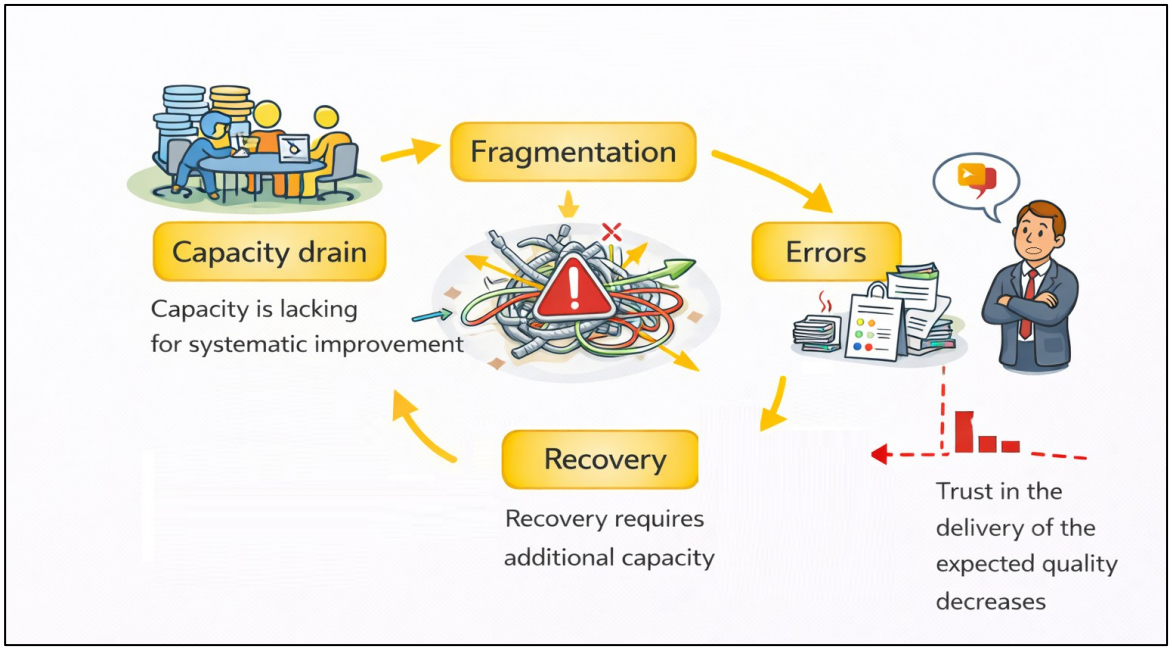

Figure 3. The vicious circle in the absence of systematic improvement

These recovery measures are rarely structural. They resolve individual cases, but do not change the system.

The result is a vicious circle:
1. fragmentation causes errors
2. errors lead to recovery
3. recovery requires extra capacity
4. there is a lack of capacity for structural improvement.

Meanwhile, customer confidence declines (Figure 3).

3.6 The strategic significance of customer impact

What is underestimated is that customer impact is not an operational detail. It affects the very *raison d'être* of the organization.

For public organizations, this means:
- declining confidence
- increasing political pressure
- and a growing accountability burden.

For private organizations:
- customer loss
- reputational damage
- and higher costs.

Fragmentation is therefore not only an internal problem, but also a strategic risk.

3.7 Key message

The customer, patient, or citizen always experiences *the whole*.
Without explicit chain management, internal control becomes external unreliability.
As long as organizations do not manage the whole, customer value remains coincidental.

3.8 Reflection questions
- Which services in your organization span multiple teams?
- Where is success measured internally, but the experience fails externally?
- Who is formally responsible for the overall result?

3.9 Learning objectives

Having read this chapter, the reader will be able to:
- explain why customer value is a chain performance
- recognize how internal logic causes external damage
- analyze where chain management is lacking in their own organization.

3.10 Preview

In the next chapter, we will shift our focus to the past.
We will show how organizations, with the best of intentions, have gradually arrived at the fragmentation we are now experiencing.

PART II – WHY EXISTING SOLUTIONS FAIL

In Part I, we saw **what goes wrong**.

Not because of bad intentions. Not because of unwillingness. But because organizations try to organize cohesion without designing it.

In Part II, we go one step further. Here, we examine **the reflex** that virtually every organization develops when complexity increases:

More tuning, more consultation, more coordination.

That feels logical. It feels responsible. And it slowly makes everything worse.

This part shows why coordination:
- is not a form of management
- does not offer a sustainable solution
- and ultimately becomes a source of complexity itself.

Not as a judgment on people. But as an analysis of system behavior.
Only when it becomes clear **why coordination fails** does space arise for the next question:

If coordination does not solve the problem — what does?

Part II thus prepares for the tipping point.

Not toward a new framework, but toward a different understanding of management.

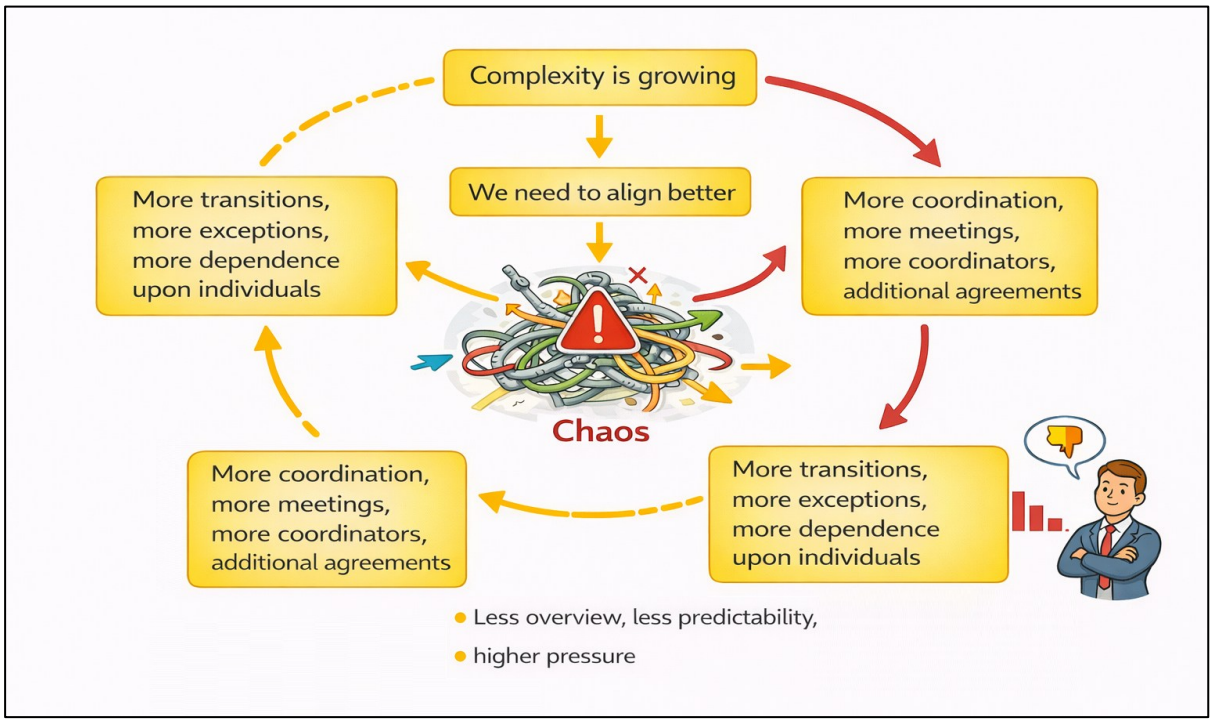

Figure 4. Coordination as a reflex — not as a solution

4 How we got here

4.1 Context & recognition

🧭 Rivertown moment
Over the past fifteen years, Rivertown has seen an impressive list of initiatives: a new case management system, a process optimization program, an agile transformation in IT, a chain management project in the social domain, and, most recently, a data-driven working initiative. Each project had a clear purpose. Each project delivered visible improvements. And yet, today, the cohesion is weaker than ever.

When managers try to understand how their organization has become so fragmented, they express surprise: "But we've improved so much, haven't we?"

That's not a paradox. It's precisely the crux of the problem.

4.2 Professionalism as a sum total

Most organizations did not *start out* as fragmented systems. Fragmentation is not a starting point, but an end result.

That end result arises when professionalization is designed as a series of separate improvements:
- each problem gets its own approach
- each discipline gets its own framework and techniques
- each team gets its own optimization.

This seems rational. It is in line with specialization, craftsmanship, and responsibility. But it introduces a subtle mechanism: professionalism becomes a sum, not a coherent whole.

What is missing is an explicit design for how all these improvements should relate to each other.

💬 ⚡ The Countervoice
"But you can't design everything at once, can you?"

No—that's true. But you can define the system within which improvements take place.

4.3 Best practices as a historical accelerator of fragmentation

Best practices have brought organizations a lot. They offered guidance in times of growth, digitization, and increasing complexity. They gave professionals language, structure, and legitimacy.

But best practices also have a **blind spot**: they describe how something can be done without deriving it from an underlying architecture. They do not describe how the whole remains manageable.

When best practices are implemented:
- without a common framework
- without explicit interfaces
- and without shared process logic...

a landscape emerges in which each domain develops its own truth.

🧭 Rivertown moment
In Rivertown, IT uses different definitions for 'incident', 'change' and 'service' than the Social Domain. Both work professionally. But their professionalism does not align.

Fragmentation is not a mistake here, but a logical historical consequence.

4.4 Accumulation of solutions

A second mechanism that reinforces fragmentation is accumulation. When a new challenge arises, the organization falls back on what worked before:

- a new program
- an additional role
- a supplementary technique.

The existing remains intact. The new is added on top.

This creates layer upon layer:

- processes alongside processes
- roles alongside roles
- consultation structures alongside consultation structures.

Each one understandable on its own. Together, confusing.

The Countervoice

"But stopping something old is often politically impossible."

That's true. And that's exactly why a central management architecture is necessary.

4.5 Why no one sees the big picture anymore

Over time, a situation arises in which:

- no one has an overview of all initiatives
- dependencies remain implicit
- and coherence exists mainly in the minds of experienced people.

Control thus shifts from structure to individuals.

This feels efficient, but it is risky. Because as soon as those individuals:

- become overburdened
- leave
- or have other priorities...

the cohesion disappears.

Rivertown moment

In Rivertown, it turns out that three people "know how things really are." They are constantly involved in making decisions. Their schedules are permanently full. The system relies on them.

This is not leadership.
This is a system that has outsourced its design to individuals.

4.6 The myth of failure

When fragmentation becomes apparent, organizations look for the cause in:

- poor implementations
- insufficient maturity
- resistance to change.

But these explanations miss the point.

Most organizations are not fragmented despite their improvements, but because of the way they have organized those improvements.

The problem is not failure.
The problem is the lack of an explicit management architecture.

4.7 Key message

Fragmentation is not an accident and not a sign of incompetence.
Fragmentation is the logical end result of years of professionalization without architecture.
As long as this is not recognized, every new improvement will continue to contribute to the same problem.

4.8 Reflection questions

- Which improvement initiatives have been piled on top of each other in your organization?
- Which connections are currently being maintained mainly by people?
- Where is there a lack of an explicit structure for new changes?

4.9 Learning objectives

Having read this chapter, the reader will be able to:

- explain how fragmentation arises historically
- analyze why best practices accelerate fragmentation
- recognize how the accumulation of solutions undermines cohesion.

4.10 Preview

In the next chapter, we will zoom in further on best practices themselves.

We will show why, however valuable they may be, they structurally create new silos as long as they are not embedded in an explicit management architecture.

5 Best practices as a structural cause of new silos

5.1 Context & recognition

Rivertown moment
In Rivertown, IT works according to ITIL principles, the Social Domain uses its own case methodology, and Environmental Management focuses on legal deadlines and legal due diligence with a Case System.
Each domain can explain why its routines are logical and professional.
When files overlap, discussions arise about definitions, responsibilities, and priorities.
Not because anyone is being unreasonable, but because no one has the same starting point.

Best practices feel safe. They offer guidance, legitimacy, and recognition. They give professionals the feeling that they are "doing it right." And that is the case—within their own context.

Yet it is precisely in organizations rich in best practices that fragmentation grows most rapidly.

5.2 What best practices actually are

A best practice is essentially an example. A description of how another organization, in a specific context, has solved a particular problem.

Best practices are:
- context-bound
- historically grown
- and focused on local improvement.

That makes them valuable. But it also makes them *fundamentally unsuitable* as a design principle for the whole.

Rivertown moment
Rivertown works with many best practices, provided by the VNG – the Association of Dutch Municipalities:
* The GEMMA (Municipal Model Architecture) provides numerous practical examples for the organization and daily routines of municipalities.
* Case-based working provides an enormous database of practical examples of everyday municipal activities, under the heading of 'process-based working'.
* Service Blueprinting is recommended as a technique for recording routines based on practice – not based on an underlying process design.
* Customer journeys describe routines based on practical experience.
* Process description is promoted by using BPMN – a toolkit that is not designed from an architectural perspective – and therefore supports all conceivable practices.
* Common Ground is based on practical technology, not on a single shared management vision.

When best practices are used as building blocks of an organization-wide management system, a fundamental error occurs: examples are treated as architecture.

The Countervoice
"But aren't they based on experience and success?"

That may be. But experience is not design.

5.3 Processes versus practices

A crucial distinction that is missing in many organizations is the difference between **process** and **practice**.

USM applies a strict definition here:
- a **process** consists exclusively of actions, the things we have to do
- a **practice** (practical routine) is the concrete implementation of this with people, roles, and tools, in the form of local procedures and work instructions.

In real life, anything that is repeatable is quickly referred to as a 'process'. But as soon as roles, systems, and authorities become part of the description, we no longer talk about a process, but about a practice – a practical routine.

Best practices almost always describe such practical routines. They describe:
- who does something
- with what tool
- in what order
- under what conditions.

What they do not do is explicitly record the underlying, general process logic that connects different routines.

Rivertown moment
In Rivertown, IT has an 'incident process', the Social Domain refers to 'escalations' and Environmental Management to 'deviations'. The actions are similar, but the meaning differs. Integration fails not because of effort, but because of logic.

5.4 How best practices create silos

When each domain implements its own best practices:
- different definitions arise for similar events
- responsibilities are assigned differently
- and each team develops its own internal consistency.

That internal consistency is precisely what makes silos strong.

Best practices encourage local optimization. They make it possible to work professionally within a single domain, without the need for an explicit relationship with other domains.

The result is an organization that consists of:
- professionally designed silos
- with informal bridges between them
- that require constant maintenance.

The Countervoice
"But without best practices, won't it become arbitrary?"

Without architecture, yes. With architecture, no.
Those who bring their best practices under that architecture utilize the experiences of others, but now in a structured way.

5.5 Why more best practices exacerbate the problem

When fragmentation becomes visible, the reflex is to:
- add yet another best practice
- introduce an integration technique
- or create an additional coordinating role.

This exacerbates the problem because:
- variety increases
- coordination becomes more complex
- and the need for coordination grows.

The system becomes saturated. Not because of a lack of professionalism, but because of too much *uncoordinated* professionalism.

> 🧭 **Rivertown moment**
> After the introduction of a chain management model in Rivertown, the number of consultations increases.
> Coherence seems to have improved, but is completely dependent on the coordinators involved. As soon as they are absent, the chain comes to a standstill.

5.6 Best practices and the illusion of maturity

Many maturity models suggest that organizations develop through:
- more processes
- more roles
- more standards.

But without explicit cohesion, this forces a mature façade: everything looks professional, but the whole is fragile.

USM asks a different question: not "How mature is each component?" but "How manageable is the whole?".

That is a fundamentally different perspective.

5.7 Key message

Best practices are valuable within their context, but dangerous as a substitute for architecture.
Best practices standardize routines. Architecture standardizes controllability.
Without that distinction, fragmentation occurs, no matter how professionally one works.

5.8 Reflection questions
- Which best practices are considered "the norm" in your organization?
- Where do definitions for similar events differ between domains?
- Which coherence is currently achieved mainly by people rather than by structure?

5.9 Learning objectives

Having read this chapter, the reader will be able to:
- explain the difference between process and practice
- explain why best practices reinforce silos
- analyze where local optimization causes systemic damage.

5.10 Preview

In the next chapter, we will examine why adding yet another framework or model does not solve this problem, but rather deepens it — and why true simplification is only possible through a paradigm shift.

6 Why yet another framework solves nothing

6.1 Context & recognition

Rivertown moment
A new problem is becoming apparent in Rivertown: the turnaround time for complex cases continues to increase. The analysis is quickly made. "We lack cohesion."
The proposed solution follows the familiar pattern: a new framework for chain cooperation, supported by an external party, with accompanying training and tooling.

The reaction is recognizable. When existing solutions fall short, it seems logical to add something new. Something that promises to bring clarity. Something that can be placed 'above' the existing approaches. "The Next Shiny New Thing That Really Helps."

But that is precisely where the next round of fragmentation begins.

6.2 The reflex: add rather than redesign

Organizations are good at improving. They are less good at redesigning.

When problems persist, the reflex is almost always to:
- introduce a new routine
- appoint an additional role
- introduce a supplementary model.

This reflex is understandable. It offers speed, visibility, and a sense of action. But it ignores a fundamental question: why don't the existing approaches solve the problem?

As long as that question is not asked, every new framework becomes an extra layer on top of an unchanged design.

The Countervoice
"But we can't shut everything down to redesign first, can we?"

Redesigning does not mean shutting down.
It means stopping the accumulation.

6.3 Frameworks operate within their own paradigm

Every framework, however valuable, is built within a specific conceptual context. It defines:
- what is relevant
- what is measured
- and where responsibility lies.

When an organization uses multiple frameworks side by side, the result is not a richer whole, but a multiple paradigm:
- *different* definitions of success
- *different* moments of control
- *different* images of what is 'good'.

Frameworks do not compete explicitly, but implicitly. They each demand attention, compliance, and interpretation. The whole becomes fragmented.

Rivertown moment
In Rivertown, GEMMA, Common Ground, ITIL, agile working, chain management, and data-driven management coexist. Each initiative has its own dashboards, consultation structures, and responsible parties. No one can explain how these frameworks relate to each other.

6.4 More frameworks means more coordination

When multiple frameworks coexist, a new problem arises: coordination *between* frameworks. The complexity increases quadratically with the number of components.

This forces:
- additional consultation to harmonize interpretations
- coordinating roles that must oversee 'the whole'
- exceptions that fall outside any framework.

Complexity is thus combated with even more coordination. And that is precisely the mechanism that perpetuates fragmentation.

The Countervoice
"But isn't that coordination necessary?"

Yes — as long as the design is lacking. But the more coordination is needed, the clearer the design problem becomes.

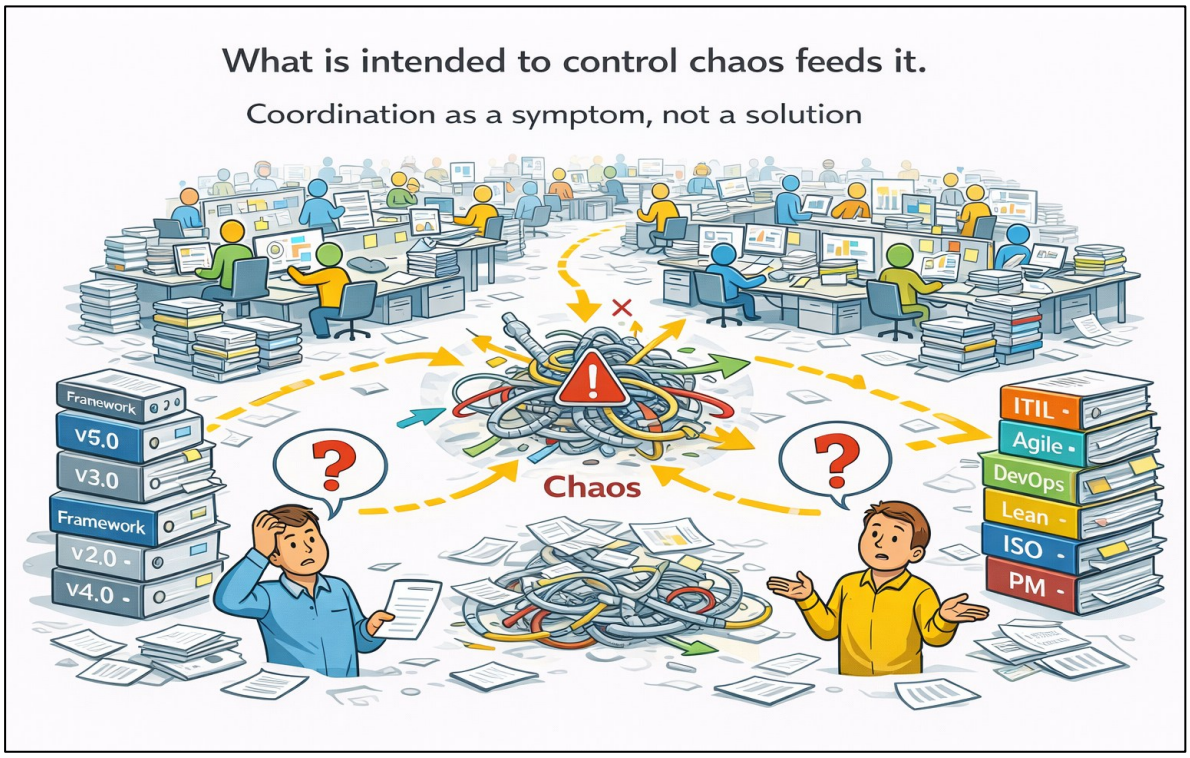

Figure 5. What is intended to control chaos feeds it.

6.5 The illusion of progress

New frameworks bring a temporary sense of progress:
- there is energy
- there is attention
- there are new words.

But as long as the underlying paradigm does not change, the old patterns remain intact. Over time:
- new exceptions are created
- new interpretations arise
- and the need for additional agreements grows.

What remains is an organization that appears 'mature' but is becoming increasingly difficult to manage.

6.6 Why simplification is so difficult

True simplification requires something that many organizations find challenging:
- explicit choices
- clear boundaries
- and letting go of familiar frameworks.

Adding frameworks feels safe. Removing frameworks feels risky.

Yet it is precisely this reduction that is needed to restore coherence.

> 🧭 **Rivertown moment**
> When Rivertown proposes to question an existing framework, resistance arises. Not because it doesn't work, but because it works *somewhere*. No one feels ownership of the whole, but everyone protects their part.

6.7 Key message

As long as problems are approached within the same framework, solutions remain variations on the same theme.
Without a paradigm shift, each new framework increases the complexity it claims to combat.
Simplification does not come from addition, but from redesign.

6.8 Reflection questions
- Which frameworks or techniques coexist in your organization?
- Where is coordination needed between frameworks rather than between activities?
- Which problems keep recurring despite new initiatives?

6.9 Learning objectives

Having read this chapter, the reader will be able to:
- explain why framework-stacking reinforces fragmentation
- analyze how coordination grows in the absence of design
- recognize why true simplification requires a paradigm shift.

6.10 Preview

In the next chapter, we will make that paradigm shift explicit.
We will show why treating the symptoms feels logical but is structurally inadequate — and why the time has come to redefine management itself.

PART III – THE TURNING POINT: FROM COORDINATING TO MANAGING

After Part II, it has become clear what does not work. Not because people are doing it wrong, but because coordination cannot replace what has not been designed.

That insight evokes an uncomfortable void. Because if coordination is not the answer, what is the alternative?

In Part III, we make that shift explicit. Not by introducing a new solution, but by practicing a different way of looking at the same things.

Here, the perspective shifts:
- from behavior to system
- from *how* we work to *what* we manage
- from keeping people together to designing cohesion.

This is the part where many readers get stuck. Not because it becomes complicated, but because familiar certainties have to be let go. Unfortunately, *unlearning* is one of the hardest things to do. Yet:

We cannot solve our problems with the same thinking we used when we created them. [Einstein]

Part III does not ask for agreement. It only asks for a willingness to entertain one thought:

Perhaps the problem is not that we manage too little,
but that we never really learned what management is.

With that realization, the steering wheel comes into view. Still abstract. But inevitable.

And from here, there is no turning back to "just better coordination."

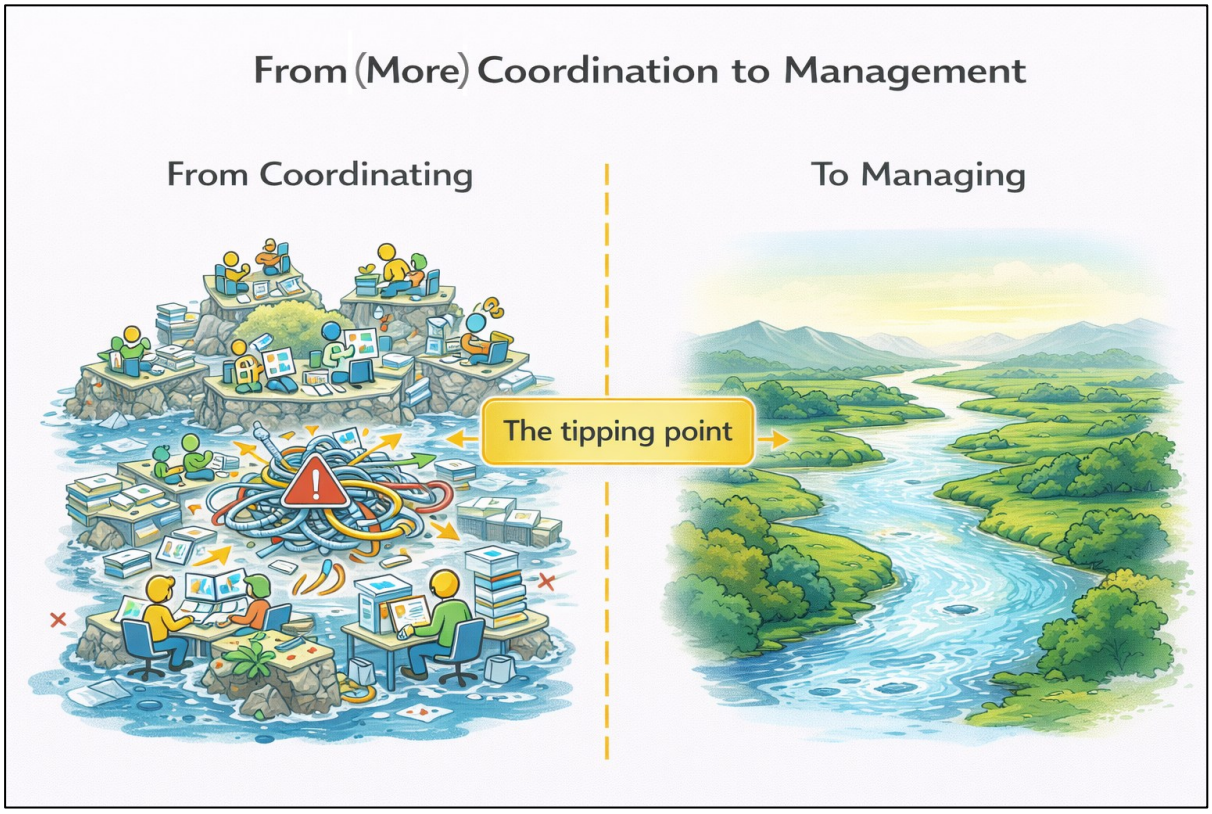

Figure 6. From (more) coordination to management

7 Stop treating the symptoms

7.1 Context & recognition

Rivertown moment
After a series of stalled projects, Rivertown decides to set up a new 'chain consultation'. The agenda is tight, the participants are knowledgeable, and the intentions are good. After a few weeks, progress seems to be made. After a few months, the consultation has become indispensable. When a consultation is canceled, work immediately comes to a standstill.

This pattern is recognizable in many organizations.
When problems recur, the reflex is to actively combat them. Not by changing the system, but by building something around it.

Treating the symptoms feels decisive. It shows that action is being taken. But precisely because of this, the underlying problem remains intact.

7.2 What treating the symptoms really is

Treating the symptoms focuses on visible disruptions: incidents, escalations, exceptions, and complaints.

That is understandable. Symptoms are concrete, urgent, and politically sensitive. They demand action, and quickly.

But treating the symptoms has a structural limitation: it implicitly accepts the existing design.

When a problem is solved without questioning the system, it confirms that the system is 'fundamentally sound'. The solution is added as a corrective mechanism.

The Countervoice
"But we can't question the entire system every time a problem arises, can we?"

No. But if the same problem recurs, that is exactly what is needed.

7.3 The shift from design to repair

In organizations where symptom control becomes the norm, the focus of management shifts:
- from design to repair
- from setting frames to intervention
- from prevention to repair.

As a result, managers increasingly become:
- firefighters
- escalation points
- and connectors between silos.

Meanwhile, the system learns nothing. It only becomes more dependent on intervention.

Rivertown moment
In Rivertown, it appears that a small number of managers are systematically flown in when problems arise. Their schedules are filled with 'just checking in' and 'coordinating'. As soon as they are gone, the problems pile up.

This is not a leadership model.
It is an alarm signal.

7.4 Why treating symptoms is so persistent

Treating symptoms is not only tempting, but also rational within the existing paradigm.

It offers:
- quick visible results
- political reassurance
- and a sense of control.

In addition, redesign evokes uncertainty:
- it makes responsibilities explicit
- it forces choices
- and it breaks existing power relations.

The Countervoice

"But redesign takes time, and we don't have that."

Treating the symptoms takes more time, only spread out and invisible.

7.5 The effect on complexity

Every symptom solution adds something to the system:
- an extra consultation
- an additional role
- an exception procedure
- a temporary workaround.

Individually, these additions seem harmless. Collectively, they increase complexity.

The system becomes:
- less transparent
- more difficult to explain
- and increasingly difficult to manage.

Complexity is not solved, but institutionalized.

7.6 When treating the symptoms fails

Treating the symptoms does not fail immediately. It fails slowly.

Only when:
- key people drop out
- work pressure becomes structurally unsustainable
- or external pressure increases...

does the fragility of the system become apparent.

Rivertown moment

When an experienced coordinator in Rivertown is temporarily unavailable, several chains come to a standstill. No one knows exactly who is responsible for what. The cohesion was in one head.

Upon closer inspection, it turns out that not only chains, but also less complex tasks depend on the availability of a single employee.

This is the moment when organizations speak of a 'crisis'.
In reality, it is the result of years of treating (only) the symptoms.

7.7 The necessary reversal

USM calls for a fundamental reversal in thinking:
- not: "How do we solve this problem now?"
- but: "What in our design explains why this problem occurs?"

That means:
- using problems as a design signal
- seeing coordination as a symptom
- and not confusing recovery with management.

This reversal is uncomfortable, but inevitable.

7.8 Key message

Treating symptoms is not a lack of professionalism, but a sign that professionalism is misplaced.

As long as problems are solved without changing the design, the system itself will continue to produce the problem.

7.9 Reflection questions

- Which problems keep recurring in your organization?
- Which solutions mainly add extra coordination?
- Where is coordination structurally necessary to keep the system working?

7.10 Learning objectives

Having read this chapter, the reader will be able to:
- explain what symptom control is and why it is structurally inadequate
- recognize how recovery mechanisms increase complexity
- analyze which problems in their own organization are design signals.

7.11 Preview

In the next chapter, we will take the next step.
We will return to the core question: what is management, really?
And why does what is called management in many organizations turn out to be coordination in reality?

8 Back to the essence of management

8.1 Context & recognition

Rivertown moment
The Director of Operations at Rivertown has a schedule full of meetings. Chain meetings, escalation meetings, coordination meetings "to get things moving."
When he is absent for a week, it becomes clear how vulnerable the system is: decisions are left unmade, exceptions are postponed, and teams wait for direction.
Not because they do not want to work, but because no one is structurally responsible for the coherence of the whole.

In many organizations, this is seen as proof of importance: "See, he's indispensable."

USM interprets this differently: the system is immature.

8.2 The confusion surrounding the concept of management

In practice, the word *management* is used to describe a wide range of activities:
- managing people
- making decisions
- solving problems
- setting priorities
- coordinating between parties.

These activities can all be part of someone's job. But together they do not constitute a definition of management.

What is missing is a clear distinction between:
- working *within* the system, and
- taking responsibility *for* the system.

In fragmented organizations, managers are mainly concerned with working *in* the system. They compensate for what the system cannot do. This is visible, intensive, and often appreciated.

But it is not management in the systemic sense.

The Countervoice
"But if I don't do this, nothing will happen."

That's right.
And that is precisely what shows that the system is not designed.

8.3 Management as system responsibility

USM returns to a classic but often forgotten definition:

Management is the design, monitoring, and improvement of the system in which work takes place.

This means that management focuses on:
- making responsibilities explicit
- defining interfaces
- limiting variety
- and establishing decision moments.

Management is therefore not about solving individual problems, but about ensuring that the same problems do not structurally recur.

This requires a different kind of attention:
- less ad hoc
- less visible
- but structural and normative.

8.4 Coordination: necessary, but incorrectly positioned

Coordination is not bad. On the contrary. Coordination is necessary in every system, not only to enable implementation, but above all to manage it efficiently.

The problem arises when coordination:
- is structurally 'heavy'
- is person-dependent
- and remains necessary to achieve coherence.

Then coordination is no longer a support, but a symptom of a lack of management.

Rivertown moment
In Rivertown, several 'coordinating' functions have been created to keep chains running. They resolve bottlenecks, organize coordination, and make exceptions. The work gets done. But only as long as they are there.

USM does not consider this a success, but a signal.

The Countervoice
"But isn't coordination inevitable in complex organizations?"

Yes. But structurally 'heavy' coordination is the result of a structural design gap.

8.5 Why managers have become coordinators

The fact that many managers mainly coordinate in practice is not a personal shortcoming. It is a logical consequence of how organizations are structured.

When:
- architecture is lacking
- interfaces are implicit
- and responsibilities are diffuse...

a vacuum is created. That vacuum is filled by people who take responsibility. Those people are managers.

Their role thus shifts unnoticed from:
- designer to problem solver
- from frame setter to mediator
- from manager to coordinator.

The system rewards this behavior because it works in the short term. But in the long term, the system becomes dependent on individuals rather than on structure.

8.6 The reorganization that management requires

Once management is redefined as a system responsibility, an inevitable consequence arises: a reorganization of work.

Not everyone who is currently called a 'manager' will remain so.
Not because people fail, but because roles are being separated.

A distinction is made between:
- **managers**: responsible for design and coherence
- **coordinators**: responsible for coordination within that design
- **operators**: responsible for execution.

The number of managers is decreasing.
The number of coordinating roles is becoming more explicit.

Rivertown moment
In Rivertown, several 'managers' are being reappointed as team or process coordinators.
A smaller group is explicitly tasked with monitoring the frame.
The number of meetings is decreasing. Decision-making is accelerating.

This shift in profiles is often seen as a demotion by incumbent managers: the title *manager* is highly valued and has historically been associated with better remuneration.

The organization will therefore need to make it clear to those involved that the *coordinator* profile is just as important — or perhaps even more important — than the *manager* profile. Remuneration systems must be adjusted accordingly.

The *manager* profile has different tasks, authorities, and responsibilities than the *coordinator* profile and therefore requires different skills, knowledge, and behavior. These profile requirements must be made clear before the shift is implemented.

It must be clear to everyone which profile suits them best: are you better at designing, coordinating, and supervising, or are you better at organizing, steering, and monitoring?

The appointment of a *manager* to the role of *coordinator* can then be seen as a reward.

In practice, an organization needs far less managers than coordinators.

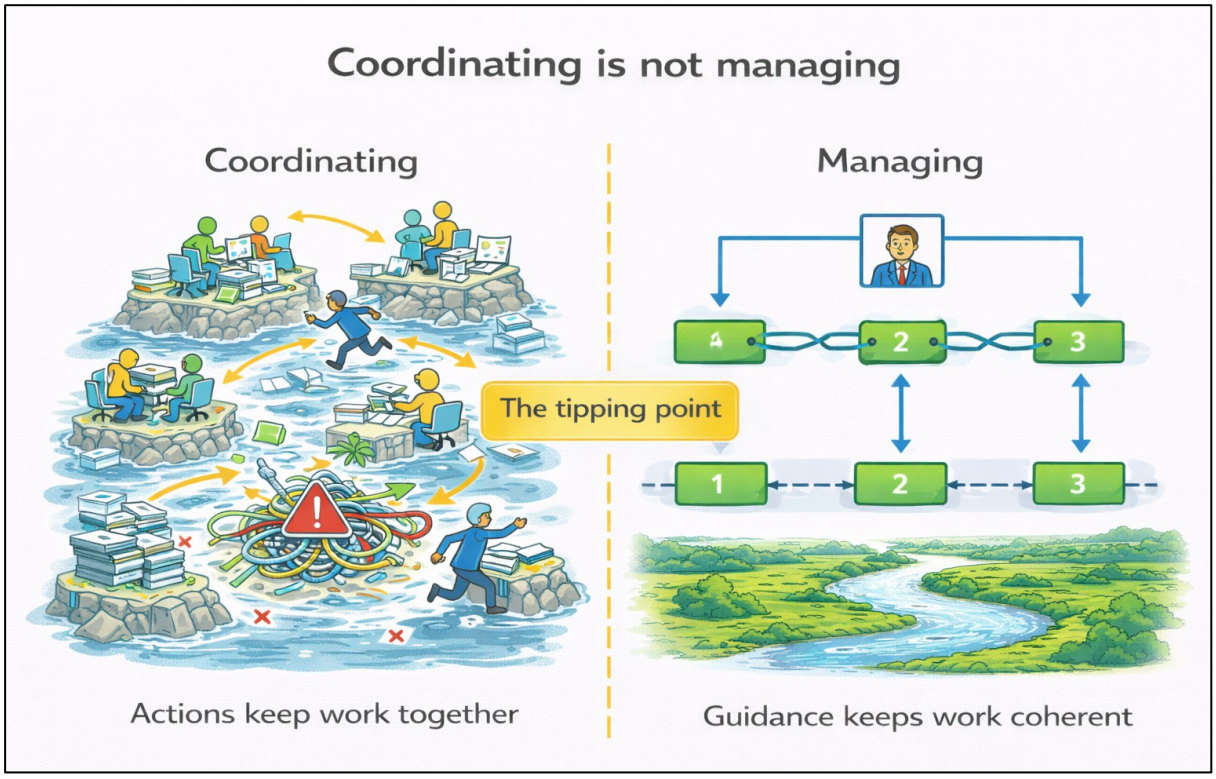

Figure 7. Coordinating is not managing

The Countervoice
"But that feels like a loss of status."

Only if status is more important than manageability.

8.7 Management without heroics

One of the most difficult aspects of this transition is that good management seems boring.

When the system is well designed:
- there are fewer crises
- fewer escalations
- fewer visible interventions.

The manager disappears from the spotlight. Not because he is unimportant, but because the system is doing its job.

This is at odds with many prevailing images of leadership. But it is *exactly* what distinguishes mature organizations.

8.8 Key message

What is called management is in reality often coordination.
Management is not an activity, but a responsibility for the design of the whole.
As long as that responsibility remains implicit, managers remain trapped in repair work.

8.9 Reflection questions

- Which tasks in your role are actually coordination?
- Where do you mainly work *in* the system rather than *on* the system?
- What would change if management were explicitly defined as a design task?

8.10 Learning objectives

Having read this chapter, the reader will be able to:
- explain what management is at its core
- identify the difference between management and coordination
- recognize why structural coordination is a design signal.

8.11 Preview

In the next chapter, we shift our perspective again.
We show why thinking in terms of teams inevitably forces fragmentation — and why chain thinking is the only sustainable basis for management in complex service delivery.

A warning for the reader: you now know the difference between manager and coordinator. In practice however, the term manager is still often used for roles that actually are coordinators. In the rest of this book, we'll stick to the term manager – even when it is clear that we're actually talking about a coordinator.
Every time you read 'manager', ask yourself: "Is this a manager, or actually a coordinator?"

9 Think in chains, not in teams

9.1 Context & recognition

🧭 Rivertown moment
In Rivertown, the organization is neatly divided into IT, HR, Social Domain, Environmental Management, Services, and Finance. Each domain has its own manager, goals, and consultation structures. When something gets stuck, the first question is always: "Which team is responsible for this?"
That seems logical. It fits with how the organization is structured.

But for the citizen waiting for a decision, this question is irrelevant.

What is a team question for the organization is a chain question for the outside world.

9.2 Teams as an internal organizing principle

Teams were created to make work manageable. They bring:
- specialization
- overview
- and internal efficiency.

As an internal organizational principle, teams are functional. The problem arises when they are also used as a management principle.

As soon as teams:
- become responsible for "their part"
- are judged on local performance
- and consider their boundaries as a given...

the responsibility for the whole disappears from view.

Teams are then no longer a means, but an end in themselves.

💬⚡ The Countervoice
"But without teams, won't it be chaos?"

Without teams, yes. Without chain management, too.

9.3 The chain as a true unit of value

USM starts from a different reality: value is not created within teams, but across teams.

A service:
- rarely starts in one domain
- rarely ends in that same domain
- and is delivered through a series of contributions.

We call this series the chain. Others call it a 'value stream'.

In a chain:
- dependencies are inevitable
- transfers determine quality
- and cohesion is essential.

Without explicit chain management, the chain remains implicit — and therefore vulnerable.

🧭 Rivertown moment
In Rivertown, it turns out that no one can outline the entire process of a move. Everyone knows their part. No one knows the whole.

9.4 Why chains remain invisible

Chains remain invisible because organizations manage what they see:
- departments
- teams
- functions
- systems.

Chains are not formal entities. They have:
- no hierarchical place
- no fixed owner
- and no explicit name.

What has no name is not managed. That is why chain problems are treated as incidents, even though they are structural.

The Countervoice
"But chains are constantly changing, aren't they?"

Yes. And that is precisely why they must be manageable.

9.5 Thinking in terms of dependencies

Chain management starts with recognizing dependencies. Not as an exception, but as a given.

Instead of asking:
- "Who is responsible?"

chain thinking asks:
- "What links are needed to provide this service?"
- "Where do responsibilities overlap?"
- "Which transfers are critical?"

This perspective reveals where:
- coordination is structurally necessary
- errors occur
- and decision-making stalls.

9.6 Interoperability as a prerequisite

When chains become explicit, a new requirement arises:
organizational **interoperability**.

Organizational interoperability does not mean:
- the same tools
- the same routines
- or the same people.

It means:
- understandable interfaces
- shared meaning of events
- and predictable process logic.

USM does not consider interoperability to be a technical problem, but a management issue.

Rivertown moment
IT, HR, and the Social Domain in Rivertown use different systems and terms.
That is not a problem.
The only problem is that they do not know when and why they need each other.

9.7 From team thinking to chain thinking

As soon as chains are explicitly named, the nature of responsibility also changes.

Responsibility shifts:
- from team results
- to chain performance.

This requires a different role for management:
- less defending of their own domain
- more monitoring of cohesion
- and explicit choices about priorities in the chain.

This is not an additional task.

It is a different definition of *the same* task.

The Countervoice
"But then the team manager loses control."

No. He only loses the *illusion* of control. A team manager was never responsible for the chain.

Figure 8. Not as silos, but jointly responsible for the end result

9.8 Key message

Teams organize work. Chains deliver value.
Without explicit chain management, team logic and hierarchy become the steering mechanism of the organization.
And that inevitably forces fragmentation.

9.9 Reflection questions

- Which services in your organization demonstrably span multiple teams?
- Who is responsible for the chain performance as a whole?
- Where are problems passed on instead of being solved?

9.10 Learning objectives

Having read this chapter, the reader will be able to:

- explain why teams are not a suitable unit of control once value delivery spans multiple domains,
- identify chains as carriers of customer value,
- analyze where chain responsibility is lacking.

9.11 Preview

In the next chapter, we will make the transition from thinking to designing.
We will answer the question that has now become inevitable:
What is USM really — and why is it not a framework, but a management architecture?

PART IV – ARCHITECTURE AS A STEERING TOOL

Up to this point, we have mainly explored why organizations get stuck.
In Part IV, the perspective shifts definitively from analysis to design.

Not yet another framework.
Not yet another optimization.
But the question that changes everything:

How do you organize management in such a way that cohesion no longer needs to be coordinated?

This part introduces the core of the book: Service Management Architecture (SMA) as an explicit control mechanism. Not as a technical diagram, but as an administrative foundation.

Here we see:

- how services become chains
- how processes are control mechanisms
- how roles are separated
- and how power is established in design rather than behavior.

From this point onward, USM is no longer a vision, but an **explicit architectural principle**.

What follows is sober, normative, and inevitable.
This is the part where improvisation ends and control begins.

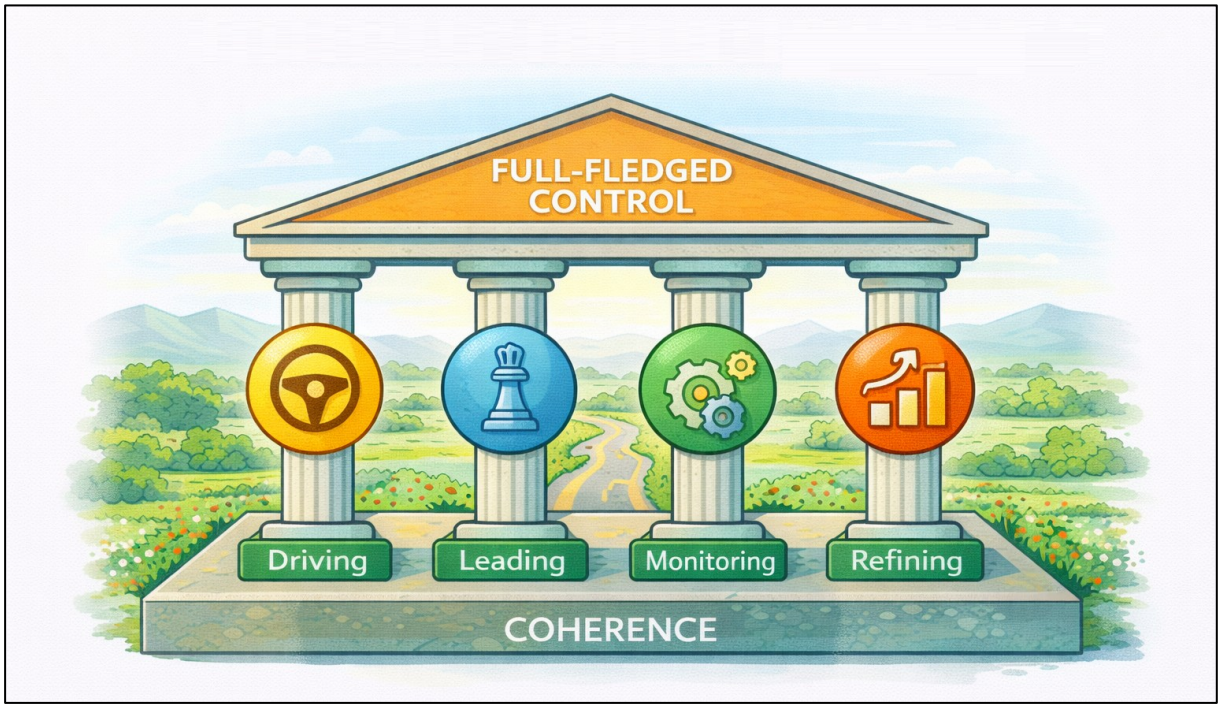

Figure 9. Coherence is not just a need. It is the foundation of full-fledged control.

10 What is USM really?

10.1 Context & recognition

🧭 **Rivertown moment**
Something has shifted in Rivertown.
Not because something has been 'introduced'.
Not because a project has been started.
But because, after the previous chapters, managers can no longer ignore something:
* services run right through the organization
* no one is managing the whole
* consultation replaces design.

Slowly, discomfort is setting in. Not among the staff, but among management.
And then the inevitable question arises: "Okay. But what is the alternative?"

At this point, many approaches stop short.
Or they reflexively jump to:

- a framework
- a maturity model
- a roadmap
- or a set of best practices "that will solve this."

Not this book.

The Countervoice
"So... this is the moment when you come up with a new model?"

No. This is the moment when we stop collecting models.

10.2 Key message

Frameworks describe how parts can work well.
USM specifies how the whole remains governable.

USM is a universal management architecture that specifies a universal operating system — a management system — for every team, for every organization, regardless of the nature of that team's or organization's services.

And that difference is crucial.

As long as you read USM as "a different way of working," you're missing the point.
USM is not about how you work in practice, but about how the whole becomes manageable.

10.3 Why USM is not what you think

🧭 In **Rivertown**, this pattern is all too familiar:
* a problem becomes apparent
* external expertise is called in
* a framework or technique is implemented
* local improvements are made
* cohesion is lacking.

USM does not fit into that pattern.

USM does not promise:

- better people
- better tools
- faster implementations.

USM does something more fundamental: it defines the minimum requirements for management in a complex organization.

The Countervoice

"But don't other frameworks say the same thing?"

No. Frameworks describe practices. USM is not a framework. USM describes the control principle that allows you to learn how to use any set of practices in a coherent way, regardless of which combination of practices you prefer.

10.4 The fundamental question that USM asks

USM does not start with:

- incidents
- changes
- services
- teams
- tooling.

USM starts with one question: "*What is the service that needs to be delivered as a whole?*"

And immediately after that:

- Which links are needed?
- Which interfaces connect those links?
- Which process logic controls that coherence?

Without explicit answers to those questions:

- there can be no management
- only reactive coordination.

Rivertown moment
There is a lot of coordination in Rivertown.
But no one can formally indicate:
* where a service begins
* where that service ends
* and who controls that service as a whole.

10.5 Service ≠ team ≠ application

In **Rivertown**, the word service is used in different ways:
* as an IT service
* as a product
* as a desk
* as a system.

USM is mercilessly clear on this point:

A service is a supported facility that consists of a combination of goods and actions that delivers value to the customer.

This means:

- a welfare service is not an application
- a permit is not a process
- a relocation is not a team task.

These are chain performances to which multiple components and internal and external teams contribute.

The Countervoice

"But then everything becomes a service, doesn't it?"

No. Only that which must be delivered across boundaries for the benefit of the customer.

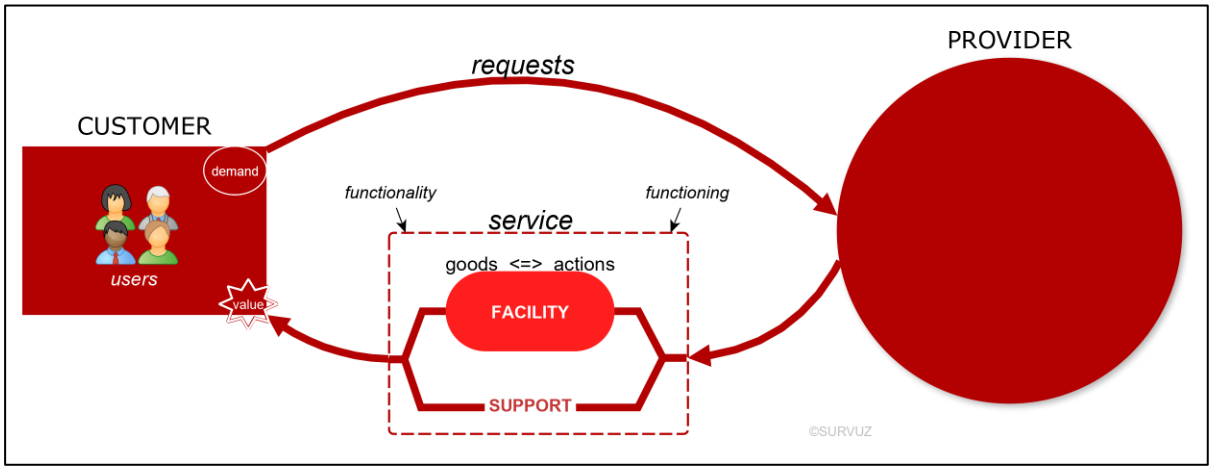

Figure 10. A service is a supported facility (Source: USM wiki)

10.6 Why USM is universal

USM makes no distinction between IT, HR, Facilities, Healthcare, or Government.

Not because they are the same, but because their management must be.

In **Rivertown:**
* IT uses different tools than HR
* the Social Domain works differently than Environmental Management
* Public Services provides very different facilities than Public Order & Safety
* suppliers have their own routines.

And that's fine.

USM does not standardize the work, but:
- the process logic
- the interfaces
- the controllability.

The Countervoice

"So everyone has to do the same thing?"

No. Everyone has to work together in a way that is understandable and mutually compatible. Because in service provision, we depend on all links in the chain.

10.7 The minimum promise of USM

USM does not promise perfection.

USM promises something more important:
- predictability
- transparency
- controllability.

🧭 In **Rivertown**, that means:
* problems land somewhere
* responsibilities are explicit
* improvements affect the system, not just a team.

That is not a 'level of maturity'.
That is basic professional hygiene.

10.8 Why USM is not an implementation project

This point is crucial.

Implementing USM like frameworks:
- renders it harmless
- reduces it to mere implementation
- and creates silos again.

USM is:
- a design choice
- an architectural decision
- a way of looking before you act.

💬 ⚡ The Countervoice
"But where do you start in concrete terms?"

Not with tooling. Not with processes. But with the organization's *raison d'être*: defining the service and its links.

10.9 The uncomfortable implication for managers

USM exposes something that many managers would rather avoid:

If you don't have an explicit management architecture,
then you don't have explicit management responsibility either.

Then you manage:
- incidents
- escalations
- exceptions.

USM forces managers to:
- take ownership of the whole
- not just their team.

That is not a role you do "on the side."
That is a different definition of management.

10.10 Reflection questions

- Which services in your municipality exist only because of multiple teams?
- Where is performance currently being managed without an explicit service definition?
- What would happen if you first defined the what and only then the how?

10.11 Learning objectives

Having read this chapter, the reader will be able to:
- explain why USM is not a framework or best practice
- identify the difference between management architecture and a framework
- explain why USM is universally applicable
- recognize why implementation without design undermines USM.

10.12 Preview

In the next chapter, we will make USM tangible.

We will introduce Service Management Architecture (SMA): the minimum design required to make chains manageable — and in which interfaces are central.

▵ Anti-misreading

This book is not a manual.

USM is not a method that you implement, a framework that you roll out, or a set of best practices that you apply.

- The five processes are not job descriptions.
- The eight workflow patterns are not step-by-step plans.
- The 1–5–8 logic is not a checklist.

USM only works when the existing mindset is replaced.

As soon as old patterns continue to guide you, USM is rendered harmless.

Warning:

If, while reading, you think, “We already do this,” read it again — but more slowly this time.

11 The Service Management Architecture (SMA)

"What is designed here cannot be worn by outsiders — only mirrored."

11.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, the conversation has definitely shifted.
Not everyone is convinced.
Not everyone is happy'.
But one illusion has disappeared: that 'better collaboration' will solve the problem.

After the previous chapters, something irreversible has happened.
The organization no longer sees itself as a collection of teams, but as a single system.

And that raises a question that is both sobering and dangerous:

"If this is a single system... where is the steering wheel?"

Not figuratively.
Literally.

💬⚡ The Countervoice
"But we already have control, don't we? MT, P&C cycle, governance?"

Yes. That's exactly what Rivertown thought too.
USM doesn't work because you've named the parts,
but because you consistently refuse to deviate from them locally.

11.2 Key message

Without an explicit SMA, an organization has no steering wheel, but a collection of separate levers.

The SMA is that steering wheel. Not:
- as an extra layer
- not as an abstract model
- not as an IT image.

But as the one, common control principle for all services — organization-wide.

11.3 What architecture is (and what it is not)

🧭 In **Rivertown**, the word architecture is associated with:
* IT diagrams
* application landscapes
* principles on PowerPoint.

That is understandable. And incorrect.

The SMA is not about practical systems, but about control.

Architecture in USM means:
- explicitly defining what needs to be controllable
- before you determine how it will be implemented.

💬 ϟ The Countervoice
"So this is enterprise architecture, but broader?"

No. This is not domain architecture.
This is not technology architecture.
This is not solution architecture.
This is management architecture.

11.4 SMA in one sentence

SMA describes how services are defined, controlled, measured, and improved as coherent chains — across the entire organization.

Not per team.
Not per supplier.
Not per tool.
But per service as a whole.

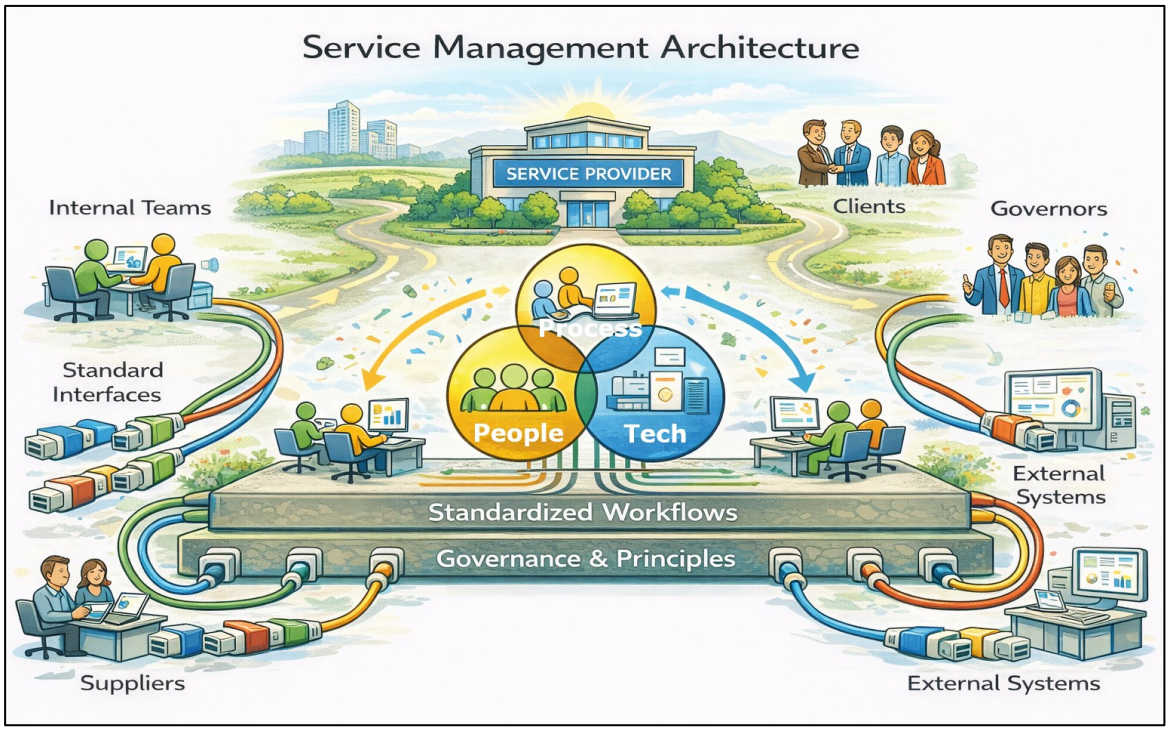

Figure 11. Service management architecture

11.5 Why Rivertown needs this

🧭 Without SMA, **Rivertown** sees the same pattern over and over again:
* teams optimize locally
* suppliers optimize contractually
* escalations go up
* decisions trickle down
* no one sees the big picture.

With SMA, it's not the effort that changes, but the order of thinking:
1. the service is explicitly defined
2. the chain becomes visible, including all actors
3. interfaces are defined
4. processes drive cohesion
5. improvement affects the system.

💬⚡ The Countervoice
"But that sounds like a lot of work."

No. It is mainly work that has never been done explicitly before.

11.6 The SMA as a universal interface

The core function of the SMA is **interoperability**.
Not technical.
But administrative.

🧭 In **Rivertown**, that means:
* IT understands when HR needs something.
* HR sees when the Social Domain gets stuck.
* Suppliers know where their contribution begins and ends.
* Managers see where decisions need to be made.

Not because people have become smarter, but because the system speaks clearly.

💬⚡ The Countervoice
"But aren't teams fundamentally different?"

Yes. And that's why their management must be the same.

11.7 One steering wheel, multiple vehicles

A crucial misunderstanding is that the SMA would enforce uniformity.

It does not.
It makes things controllable.

Compare it to traffic:
- cars differ
- trucks differ
- bicycles differ.

But without shared traffic rules, there is chaos.

The SMA is:
- not a vehicle design
- not a driving style
- not an engine.

The SMA is the traffic system.

11.8 What the SMA explicitly defines (and what it does not)

The SMA always explicitly defines:
- what a service is (and what it is not)
- which links are necessary
- how transfers take place
- which process logic applies
- where decisions are made
- how performance is measured across the chain.

What the SMA does not define:
- how teams do their work
- which tools they use
- how suppliers deliver.

The Countervoice
"So we only centralize the 'rules'?"

Yes. And that is precisely why local freedom can continue to exist. And why USM does not have to generate resistance.

11.9 Why governance without SMA fails

Rivertown does have governance:
* steering groups
* escalation procedures
* audits.

But governance without SMA is blind.

You can only steer if:
- you know what you are steering
- you see where things are getting stuck
- you understand which button has which effect.

Without architecture, governance remains:
- reactive
- political
- person-dependent.

11.10 The uncomfortable consequence

The SMA forces a decision that many organizations postpone:

Who owns the service as a whole?

Not:
- the budget
- the team
- the contract.

But the chain performance.

The Countervoice
"That will clash with existing responsibilities."

Yes. That's not a bug. That's precisely the point.

11.11 Reflection questions
- Which services in your municipality do not have an explicit owner?
- Where is managed without an overview of the entire chain?
- Which decisions are currently being made without an architectural framework?

11.12 Learning objectives

Having read this chapter, the reader will be able to:
- explain what an SMA is
- distinguish what the SMA does and does not standardize
- explain why governance without SMA is insufficient
- recognize why a single steering wheel is essential.

11.13 Looking ahead

Now that the steering wheel is visible, the next question is inevitable:

What does that steering wheel actually consist of?

In the next chapter, we introduce the core of USM: the five processes and eight workflow patterns — not as a method, but as the minimum control logic for each service.

This makes USM unavoidably concrete.

12 Five processes and eight workflow patterns

Why this minimal set makes all the difference.

12.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, the Service Management Architecture has now been explicitly named.
That alone causes unrest.

Not because people disagree with the idea of cohesion,
but because the following—and dangerous—question immediately arises:

"Okay. But how do we actually manage this?"

The reflex is predictable:
- inventory processes
- compare variants
- add exceptions
- protect local particularities.

Before you know it, the simplicity has disappeared and the architecture is once again plastered over with details.

The Countervoice
"But doesn't every team really need different processes?"

Yes. And that is precisely why we first need to clarify what is meant by "process" here.

If you think of 'process' as steps, roles, or procedures: stop.
Then you are reading USM incorrectly.

12.2 Key message

USM manages complexity not through more processes, but through minimal, universal process logic.

The five processes and eight workflow patterns are not best practices.
They are not a menu.
They are not an implementation method.

They form the lower limit for manageability.
Less is not possible. More is superfluous.

12.3 The fundamental difference: process logic versus process implementation

🧭 In **Rivertown**, there are hundreds of 'processes':
* HR processes
* IT processes
* licensing processes
* procurement processes.

USM does not deny that reality.
USM goes beneath it.

The five processes of USM explicitly do not describe:

- what people do in practice
- how teams work
- what products they deliver
- what tools are used to do so.

They describe something else: the logic of the interaction at hand, from a service perspective that focuses on the requester.

That is process logic. Everything above that is process implementation — and that may differ for each team and for each service.

But those are not processes: they are practical routines.

> **The Countervoice**
> *"But aren't we already doing that implicitly?"*
>
> Implicit management is not management.
> It is hoping that things will go well.

12.4 Why always five processes?

Because every service — in Rivertown and beyond — inevitably raises the same five questions:

1. What do we agree on?
2. How do we adapt that?
3. What do we do if things go wrong?
4. What do we deliver?
5. How do we improve?

USM turns these questions into five explicit processes:

1. Agree
2. Change
3. Recover
4. Operate
5. Improve

Not as teams. Not as silos. But as management tasks that affect every service.

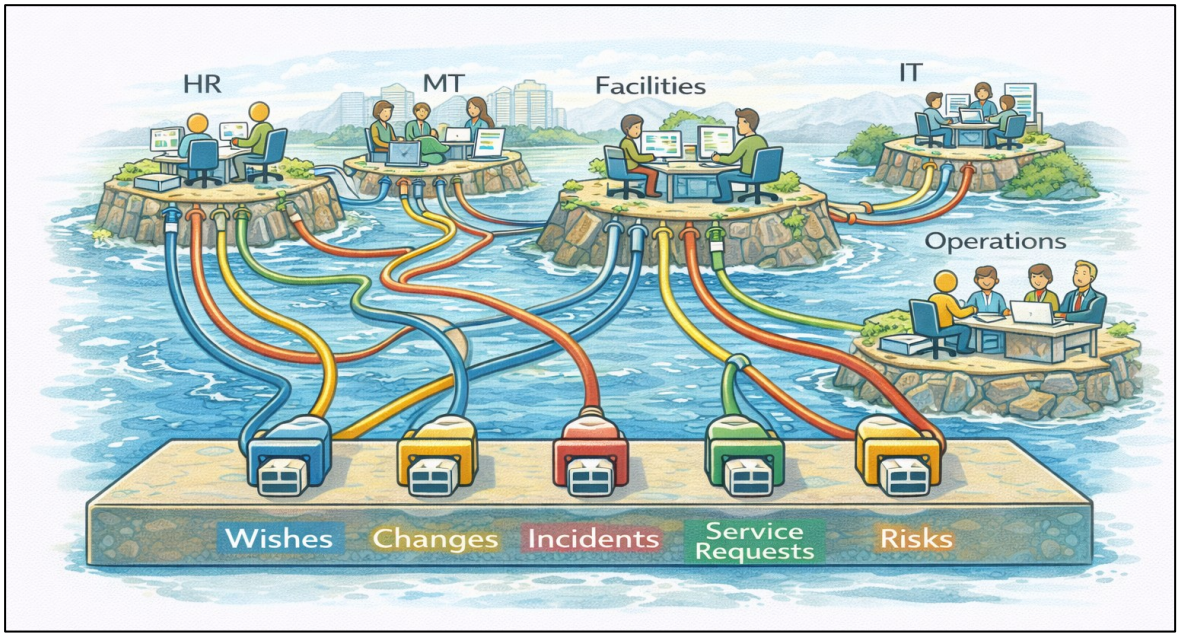

Figure 12. The five USM processes as universal connections for interactions

Redundancy is the arch enemy of efficiency. That is why the USM **process model is non-redundant**.

- All of the service provider's activities fall within those five processes, but no more than once.
- This produces a unique process model: integral and non-redundant.

The Countervoice

"But that looks suspiciously like ITIL..."

It does.
Until you see what ITIL does with it in the IT sector. ITIL's practice-based approach produces enormous redundancy that fuels complexity, encourages silo formation, and ultimately delivers a result that is at odds with the original objective: to control the service provider's performance.

12.5 Why ITIL derails here and USM does not

ITIL – a framework for best practices in IT – describes a large number of practical tasks of an IT service provider and translates these into:

- separate practices (called 'processes')
- with their own roles
- their own tooling
- their own optimizations.

The result: local maturity, but no cohesion.

USM does the opposite:

- one set of pure processes
- with one logic
- to which each team can add its own people
- and determine its own tools
- for all services.

🧭 In **Rivertown**, this means:
* HR and IT speak the same language about change.
* Social Domain and Environmental Management use the same logic for agreements.
* Suppliers are managed using the same processes.

Not because their work is the same. But because their management must be.

12.6 The role of the eight workflow patterns

Where the five processes define the what of management, the eight workflow patterns show the next step.

They describe how work flows through the system, across boundaries.

Workflow patterns are not step-by-step plans.
They are not procedures.
They are not work instructions.

They are:

- transfer patterns
- decision logic
- and feedback loops.

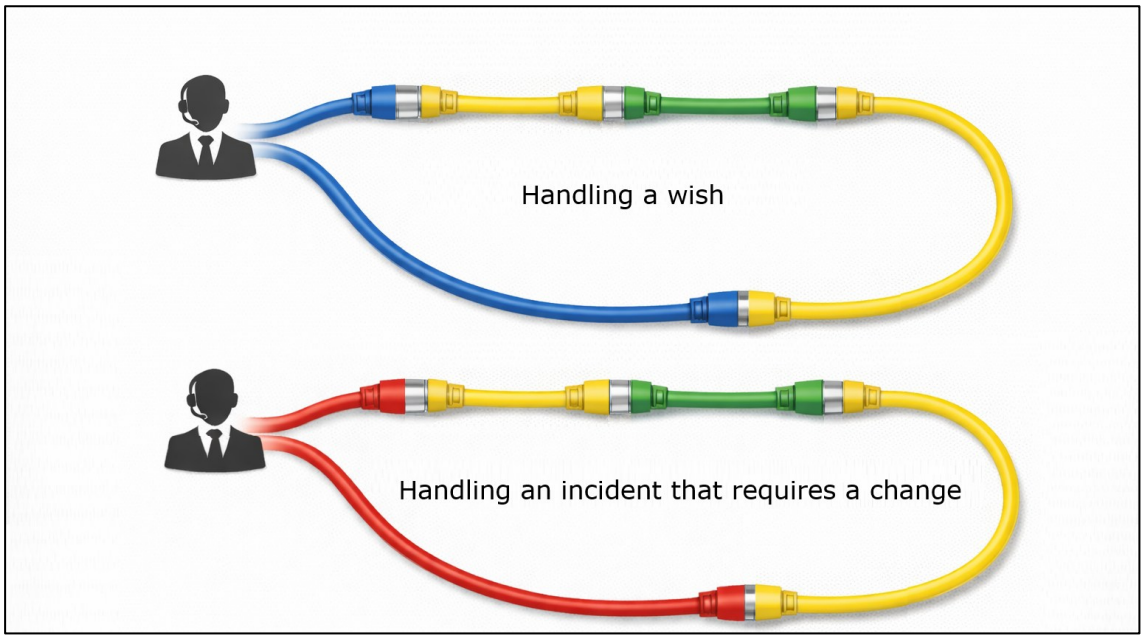

Figure 13. USM workflow patterns are composed of the components of the process model.

> **The Countervoice**
> *"But aren't workflows different by definition?"*
>
> The implementation is. The logic behind them is not.

12.7 Why eight workflow patterns are sufficient

USM reduces all chain interactions to fundamental workflow patterns. Not because the world is simple, but because control must be simple.

These patterns are a direct result of the unique process model with its five non-redundant processes. This maximum simplification through non-redundancy has an extremely pleasant consequence: there appear to be only **eight logical patterns** for handling all interactions in and with the management system.

In **Rivertown**, these workflow patterns cover:
* establishing laws and regulations and making policy
* implementing changes in municipal facilities and infrastructure
* repairing disruptions
* executing arrangements and activities
* structural improvement.

Everything that seems to fall outside of this:

- is a variant
- a specialization
- or a local interpretation.

> **The Countervoice**
> *"But what do we do with exceptions?"*
>
> Exceptions belong within workflows.

The USM processes cover all tasks that the organization must perform for its services. If there were exceptions to this, there would be a design flaw in the management system.

Ten years of experience working with the USM management system have shown that there are no exceptions: USM is a universal model for all forms of service provision. And that is exactly what it was designed for.

12.8 Why this simplicity feels uncomfortable

Because managers are used to:
- control through detail
- grip through specification
- certainty through documentation.

USM turns this around.
It states: *if you master the logic, you can tolerate variation.*

> 🧭 In **Rivertown**, this forces:
> * fewer coordination meetings
> * fewer escalations
> * fewer surprises.

Not because everything becomes predictable, but because the system becomes readable.

12.9 The painful mirror for the organization

> 🧭 When **Rivertown** takes the five processes and eight workflow patterns seriously, it becomes clear:
> * where there is duplication of work
> * where responsibility is lacking
> * where tooling has become dominant
> * where suppliers are actually in control.

This does not feel like process improvement.
It feels like unmasking.

> 💬 ⚡ **The Countervoice**
> *"This is going to provoke resistance."*
>
> Yes. Because simplicity takes away people's hiding places.

12.10 Reflection questions

- Which processes in your organization exist only because there is a lack of coherence?
- Where are the same decisions being made multiple times?
- Which workflows are currently getting stuck at team boundaries?

12.11 Learning objectives

Having read this chapter, the reader will be able to:
- explain why USM works with exactly five processes
- identify the difference between process logic and process implementation
- explain why eight workflow patterns are sufficient for chain management
- recognize where unnecessary complexity undermines manageability.

12.12 Looking ahead

Now that the steering wheel is visible and its control logic is known, the following question naturally arises:

How can all this be standardized without losing autonomy and craftsmanship?

This tension forms the heart of Part V: Freedom in captivity – freedom within constraints — meaning autonomy without local reinvention.

That is where we are headed now.

Interim conclusion – One steering wheel, finally visible

So far, nothing has been 'deployed'.

Nothing has been:
- rolled out
- implemented
- set up
- or configured.

And yet something fundamental has already changed.

In Rivertown — and for the reader — what had previously remained implicit has now been made explicit:

The organization is a single system, but was managed as if there were multiple systems.

That insight alone is irreversible.

What we now know (and can no longer deny)

After Part IV, four insights have become definitive.
Not as opinions.
But as design consequences.

Insight 1. Services are chain performances

It is not:
- teams
- functions
- or applications...

that form the unity of control. It is services.

As soon as a service touches multiple links, team control is by definition insufficient.

The Countervoice
"But didn't we already know that?"

Knowing is different from designing.

Insight 2. Without architecture, there is no control

Governance, consultation, and KPIs only work if:
- the whole is explicitly defined
- transfers are visible
- control is not dependent on individuals.

The SMA is not an extra layer.
The SMA is the foundation.

The Countervoice
"So this is mainly a conceptual framework?"

Yes. And every sustainable system starts with a conceptual framework.

Insight 3. Management requires minimal, shared logic

The five processes and eight workflow patterns define the *maximum* variety the system allows.
They are not best practices.
They are not a matter of taste.

They form the minimum set that makes every service controllable. Not per domain, per supplier, per team. But organization-wide.

The Countervoice
"But why exactly this set?"

Because everything outside of it is either interpretation or variation.

That is the essence of architecture.

"Minimalism here does not mean less responsibility, but fewer escape routes."

Insight 4. Simplicity is not simplification, but reduction to the essence

USM does not reduce the work, but the complexity of control.
By:

- eliminating redundancy
- sharing logic
- allowing differences where they belong.

That feels exciting. That *is* exciting.

Because simplicity reveals where the real problems lie.

The 1–5–8 logic as a turning point

This is the moment when USM ceases to be an idea and becomes a design decision.

- 1 service definition for all services and for all actors within the scope of the ecosystem
- 5 processes that control each service
- 8 workflow patterns that keep all chains flowing.

Together they form: *one steering wheel for the entire organization.*

Not as a framework.
Not as a model.
But as non-negotiable control logic.

The Countervoice
"So if we don't do this, we'll keep improvising?"

Yes. With increasingly better intentions, but the same result.

What this means for the next step

So far, we've talked about understanding, unlearning, reframing, designing.

What has deliberately not been the focus yet: power, autonomy, politics, freedom, resistance.

That's no coincidence.

First, we had to be clear about:

- what is needed to steer
- why it hasn't worked so far.

Only now can we ask the question that everyone already senses:

How can all this be standard without stifling people and teams?

That tension is not a side issue.
It is at the heart of successful change.

Looking ahead

In Part V – Freedom in Captivity, we show:

- why standardization is not synonymous with bureaucracy
- how autonomy actually increases through architecture
- and why USM does not create a new center of power, but makes existing power transparent.

That is where fear ends.
And work begins.

PART V – FREEDOM IN CAPTIVITY

Up to this point, this book has mainly dismantled.

It showed why silos arise logically, why coordination is not a solution, and why architecture is not a luxury but a necessity.

But at this point, a new tension arises.

Because as soon as cohesion becomes explicit, rules are laid down, and boundaries become visible, the same question inevitably arises:

What then remains of freedom?

Part V is not about more structure. It is about what structure enables.

This part shows why autonomy without architecture is an illusion, and why true professional freedom *arises* precisely when the system is clearly designed. Not by restricting people, but by freeing them from improvisation, political noise, and personal dependence. Freedom in captivity means *autonomy* without local reinvention.

Here it becomes clear:

- why power always arises where design is lacking
- why transparency is not a control instrument but a protective mechanism
- and why freedom does not disappear through limitation, but arises from it.

Part V is not a plea for centralization. It is a plea for maturity.

For organizations that no longer rely on good intentions, but dare to trust in a fair system.

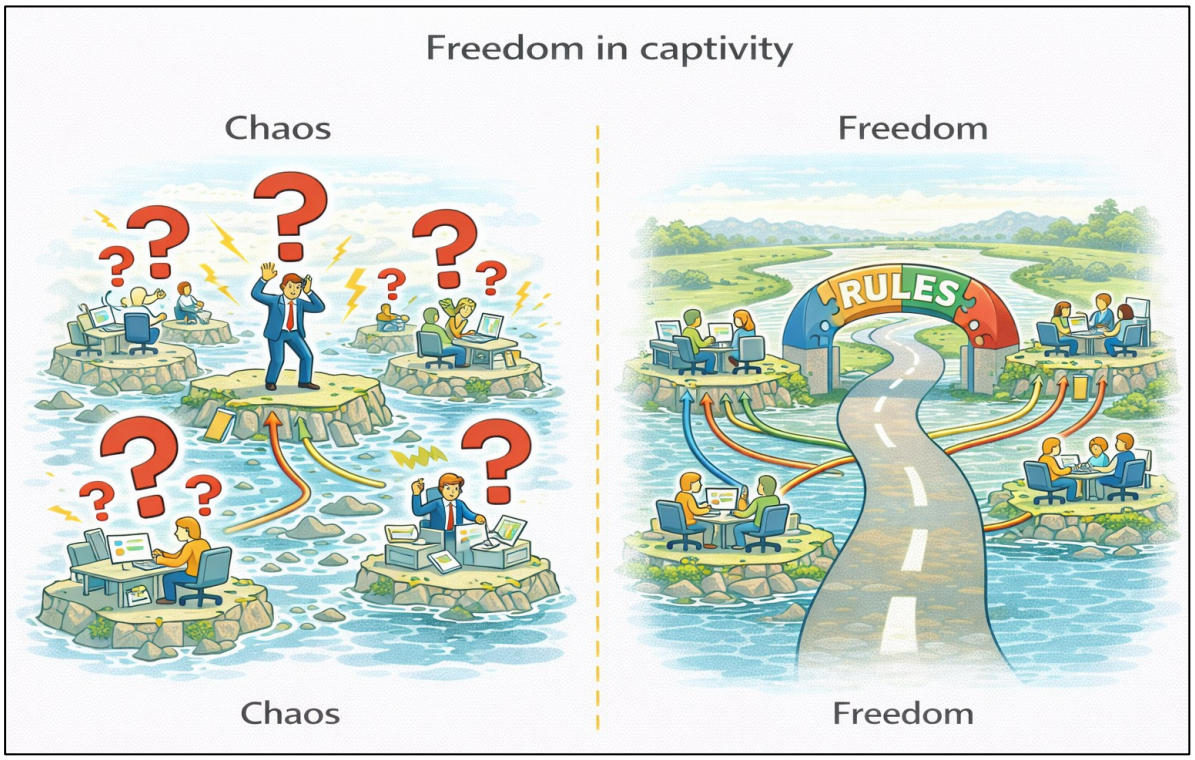

Figure 14. Freedom in captivity

13 Standardizing without bureaucracy

13.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, the word standardization is mentioned—and the temperature in the room drops.

Managers think of:
* forms
* procedures
* control
* central coordination.

Professionals think of:
* loss of craftsmanship
* less space
* more accountability.

And everyone thinks the same thing: "This is going to slow us down."

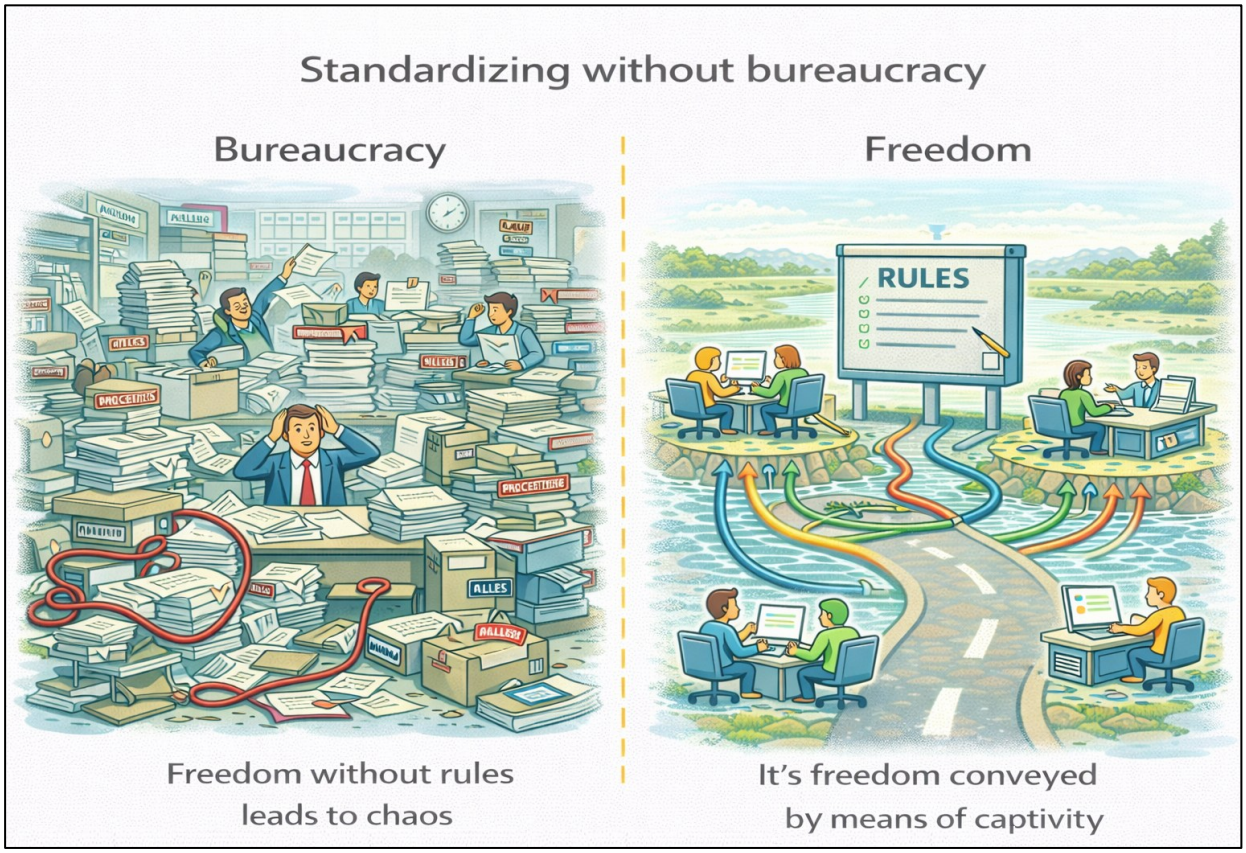

Figure 15. Shared rules provide freedom

💬 ⚡ **The Countervoice**
"We've been through this before. Standardization always ends in bureaucracy."

That's right. If you standardize at the wrong level.

Anyone who starts standardizing practices — *without* an underlying architecture from which those practices are derived — will indeed end up with bureaucracy, as the past three decades have made abundantly clear.

13.2 Key message

Bureaucracy is not caused by standardization.
It arises from standardizing the wrong things.

USM does not standardize the work.
USM standardizes the control logic.

That difference is *everything*.

13.3 Why standardization has such a bad reputation

🧭 In **Rivertown**, standardization has historically been used to:
* prevent errors
* limit variation
* increase control.

This led to:
* thick manuals
* exception procedures
* local workarounds
* and ultimately more complexity.

Standardization became synonymous with: **distrust**.

Not because standardization is wrong, but because it was applied without architecture.

The Countervoice
"But aren't rules necessary to maintain control?"

Yes. But rules without architecture are arbitrary.

13.4 What USM does standardize

USM is extremely selective.

It only standardizes what is necessary to make *the whole* manageable:
- what a service is (and what it is not)
- where transfers take place
- how coherence is monitored.

In concrete terms:
- one service definition for all services
- five processes for all management tasks
- eight workflow patterns for all chains
- one way of looking at performance and improvement.

That's all.

The Countervoice
"That still sounds like a straitjacket."

Only if you confuse it with the practical implementation.
USM limits itself to the logic.

13.5 What USM explicitly does not standardize

USM leaves completely open:
- how teams work
- what services they deliver
- what tools they use
- how suppliers deliver
- how professionals practice their profession.

🧭 In **Rivertown**, that means:
* IT can excel technically
* HR follows its own professional logic
* the Social Domain retains professional space.

As long as:
* the interfaces are clear
* the process logic is respected
* and *the whole* remains manageable.

💬 ⚡ The Countervoice

"So autonomy remains, but within certain boundaries?"

Exactly. Without boundaries, autonomy is mainly noise.

13.6 Why this actually produces more freedom

This is the tipping point that many managers miss.

Without a shared architecture, teams must:
- constantly coordinate
- resolve conflicts themselves
- fill in the gaps
- escalate in case of uncertainty.

That feels like freedom.
But it is extra work.

With USM:
- everyone knows where they stand
- transfers are predictable
- coherence no longer rests on the shoulders of individuals.

Freedom arises because people can focus on their profession again instead of on the system.

13.7 The role of management is fundamentally changing

🧭 In **Rivertown**, USM shifts management from:
* controlling
* coordinating
* intervening...

to:
* monitoring architecture
* steering towards cohesion
* improving the system.

The Countervoice
"But that requires maturity."

Yes. And that doesn't come from extra rules, but from clear design.

13.8 Why bureaucracy doesn't stand a chance here

Bureaucracy grows where:
- exceptions become the norm
- responsibilities are vague
- systems don't understand each other.

USM cuts away that breeding ground by:
- eliminating redundancy
- sharing logic
- making power visible.

What remains is not bureaucracy, but transparency.

13.9 Reflection questions
- Where in your organization is standardization avoided out of fear?
- Which rules exist only because there is a lack of coherence?
- Where would clear frameworks actually provide more space?

13.10 Learning objectives

Having read this chapter, the reader will be able to:
- explain why standardization is feared unjustifiably
- identify what USM does and does not standardize
- explain how architecture increases autonomy
- recognize where bureaucracy stems from design flaws.

13.11 Preview

Standardization inevitably raises the following fear: "*And who will then have the power*?"

In the next chapter, we will show why USM:
- does not create a new center of power
- separates functions and domains
- and prevents old silos from being replaced by new ones.

That conversation will be uncomfortable.
But it is necessary.

14 No new centers of power

14.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, the architecture is now explicit.
There is one steering wheel.
There is one shared logic.

And that is precisely where the next, almost reflexive concern arises: "Fine. But who gets the power here?"

Not out loud.
But in the corridors. In management teams. In project groups.

The Countervoice
"Won't this just end up in a new steering committee that decides everything?"

That's not cynicism. That's experience.

14.2 Key message

USM prevents the concentration of power not by ignoring power, but by explicitly separating it, limiting it, and making it visible.

New centers of power arise where:
- responsibilities are vague
- decisions are made implicitly
- architecture is lacking.

USM reverses this mechanism.

14.3 Why organizations naturally create centers of power

🧭 In **Rivertown**, centers of power are not "invented."
They arise.

Where:
* there is a lack of cohesion
* escalations linger
* decisions are needed now...
* someone takes the initiative.

This could be:
* a dominant team
* a powerful manager
* a supplier with a knowledge edge
* or a staff club with an informal mandate.

The Countervoice
"But someone has to make decisions, right?"

Certainly. But the question is: who, about what, and on what basis.

14.4 Power without architecture is always personal

Without explicit management architecture:
- power shifts to individuals
- influence becomes informal
- decisions are not traceable.

In Rivertown, you hear phrases like:
- "Give John a call, he'll take care of it."
- "That's a sensitive issue, we have to be careful with that."
- "This is how it has historically developed."

These are not arguments.
These are markers of power.

14.5 What USM does fundamentally differently

USM accepts one simple truth:

Power does not disappear through good intentions.
Power only disappears through design.

USM designs power through:
- separation of duties
- separation of domains
- explicit decision-making logic.

The Countervoice
"That sounds theoretical. How does it work in practice?"

With an architecture for the distribution of powers.
By specifying exactly who can and cannot make decisions, based on a logical structure of profiles.

14.6 Separation of duties: no dual roles, no mixing

In **Rivertown**, profiles are intertwined:
* those who design also decide
* those who deliver also manage
* those who control have an interest.

USM draws clear lines:
- Architecture determines the what.
- Management focuses on coherence.
- Coordination manages the implementation.
- Operation ensures that it is done – in accordance with the agreed intention.

Separation of duties is not only hierarchical. Above all, it is logical.

USM applies separation of duties recursively:
- Managers consult with coordinators and operators about the work, but they do not carry it out themselves.
- Coordinators organize the execution of the work by operators, but they do not carry it out themselves.

Coordination has two dimensions:
- **line steering** via hierarchical authority: team leadership, team coordination
- **process steering** based on the logic of the process, across team boundaries.

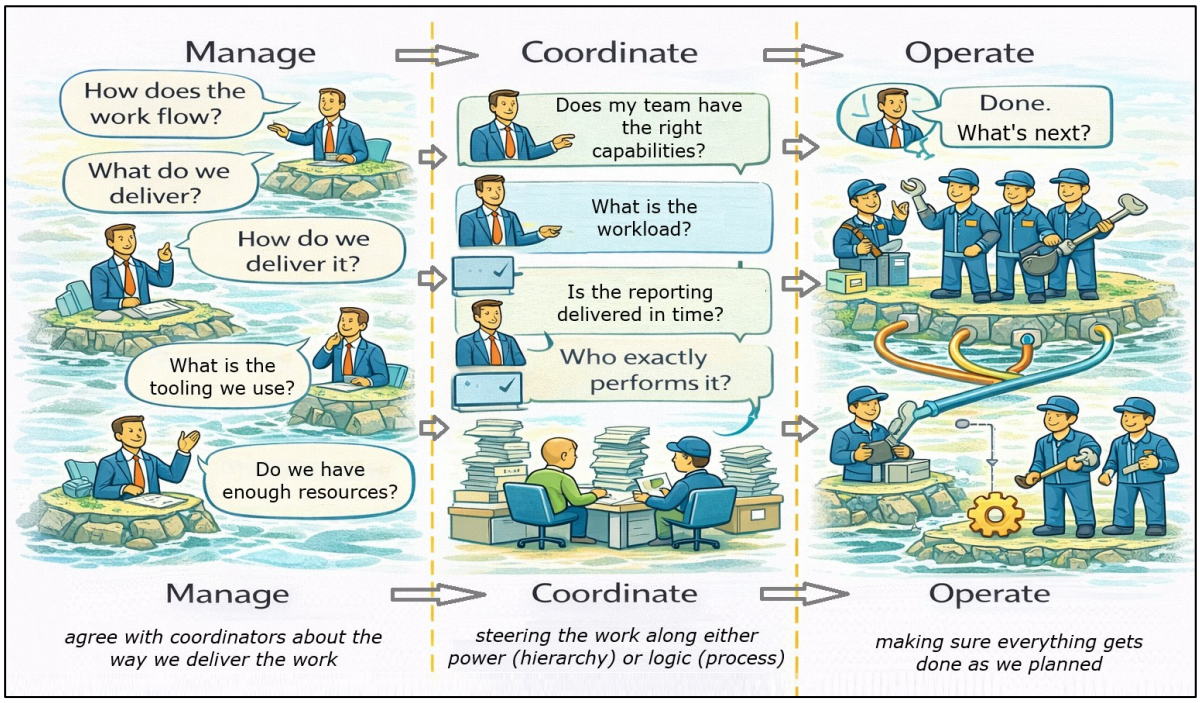

Figure 16. Separation of duties separates specification from realization, in a recursive pattern

No one is allowed to:

- design their own work
- assess their own performance
- legitimize their own exceptions.

The Countervoice

"But doesn't that slow everything down?"

On the contrary. It prevents endless rounds of corrections afterwards.

14.7 Domain separation: no hidden hegemony

Separation of duties can have organizational effects. Tasks can be clustered based on separated sets. This can lead to teams that are engaged in separate tasks:

- defining services
- drafting agreements
- purchasing goods
- managing process logic
- performing specialist tasks
- managing tooling
- supplying facilities by suppliers.

In **Rivertown**, this means:
* IT does not determine the service.
* HR does not determine the chain.
* Suppliers do not determine the architecture.

Everyone has a place.
No one has the whole picture.

The Countervoice
"So no one is ultimately responsible?"

No. Someone is always responsible for cohesion.
That is different from having power over everything.

14.8 Why 'retained organizations' go off the rails

Organizations try to resolve power issues in a complex multi-supplier ecosystem with:
- steering committees
- coordination centers
- integration teams.

The intention is good.
The design isn't. Read "The Retained Organization" for details.

Without architecture:
- management grows into content
- coordination shifts to control
- a new silo emerges.

The Countervoice
"But don't we need management when outsourcing?"

That's right. And that's exactly why architectural sovereignty is essential.

Not control as power, but control as monitoring of the design.

14.9 Transparency as an antidote to power

USM combats the concentration of power not with procedures, but with visibility.
- agreements are explicit
- decisions are traceable
- performance is chain-wide
- improvements are systemic.

What is visible cannot be manipulated unseen.

The Countervoice
"That requires mature behavior."

No. It requires a mature system.

"Transparency is not a means of control here,
but a protection against arbitrariness."

14.10 The uncomfortable truth for leaders

USM asks for something that many leaders find challenging: *giving up informal power in favor of formal cohesion.*

That means:
- less heroism
- less improvisation
- less political leeway.

But also:
- more predictability
- more trust
- more peace of mind.

14.11 Reflection questions

- Where in your organization is power concentrated informally?
- Which decisions cannot be traced back to architecture?
- Where does personal influence replace structural control?

14.12 Learning objectives

Having read this chapter, the reader will be able to:
- explain why centers of power arise naturally
- identify the difference between responsibility and power
- explain how separation of duties and separation of domains prevent abuse of power
- recognize why transparency is essential for sustainable control.

14.13 Looking ahead

After freedom and power, one more step is needed before we move on to practice.

In Part VI, we leave the design behind and step into reality: IT, HR, Social Domain, suppliers — not as separate worlds, but as links in a single system.

There, it becomes clear: USM is not a theory.
It is a way of thinking that forces a way of working.

Reflection – Make power visible before you proceed

Up to this point, USM has been primarily rational.
Logical. Consistent. Design-driven.

But Chapter 14 touched on something else: *power*.

Not as an abstract concept, but as an everyday reality in organizations.

This is the point at which many readers will want to skip ahead—and at which it is precisely worthwhile to pause and reflect.

Power does not disappear through good intentions

Many managers have grown up with the idea that:
- power is something dirty
- power should be avoided
- power regulates itself in a professional organization.

Practice teaches us otherwise.

Where:
- there is a lack of cohesion
- responsibilities are vague
- decisions are made implicitly...

power always arises.

Not organized. Not discussed. But effective nonetheless.

USM does not increase that power. USM makes it visible.

The Countervoice
"But making it visible makes it political, doesn't it?"

No. It is already political — just not open to discussion.

Architecture as a moral framework

What this book has done so far is not to propose restructuring.

It has introduced a moral framework:
- no one should decide what they do themselves
- no one should define their own performance
- no one should normalize exceptions.

Not out of mistrust.
But out of respect for the system.

Architecture is not a technique here.
It is ethics in design form.

Why this is uncomfortable for leaders

Because it requires:
- less informal influence
- less heroism
- less "just take care of it."

And more:
- explicit choices
- traceable decisions
- verifiable logic.

For some, that feels like a loss.
In reality, it is professionalization.

The Countervoice
"But what then remains of leadership?"

Exactly what matters: providing direction, not making adjustments.

Consider this before you continue reading

Before you move on to Part VI, ask yourself these questions—not to answer them, but to reflect on them:

- Where in my organization do I rely more on people than on the system?
- Which decisions can I make myself, without it being clear why I am allowed to do so?
- Where do I feel resistance to transparency—and why?

If these questions cause discomfort, that's not a problem.
It's a sign that you are no longer thinking automatically.

Preview

In Part VI, we will put the design into practice.

Not in theory.
Not in ideal organizations.
But in real departments, real teams, and real suppliers.

There it will become clear:

USM does not require perfect people.
It requires an honest system.

PART VI - USM IN PRACTICE

So far, we have dissected, redesigned, and refined the system.
We know what it takes to break down silos and why old patterns keep recurring.
But a design that does not translate into everyday work remains theory.

In Part VI, we therefore make a conscious shift:

- from architecture to practice
- from logic to behavior
- from systems thinking to action.

Not by adding new techniques, but by showing what actually changes for the people who have to make it happen every day: IT, managers, and teams.

This part is not about ideal images.
It is about recognition.
About what brings relief, what causes friction, and what suddenly brings peace when the system starts to work.

Here it becomes clear that USM is not an extra layer on top of the work, but a way to finally stop compensating — and start working normally.

15 From IT to Enterprise Service Management (ESM)

15.1 Context & recognition

🧭 **Rivertown moment**
In Rivertown, USM starts where almost everything began: with IT.

That makes sense:
* IT affects all services.
* IT is the first to feel the pain in the chain.
* IT sees dependencies derail on a daily basis.

The first conversations are therefore about:
* incidents that are not 'IT'
* changes that get stuck at HR or Procurement
* suppliers who block each other
* citizens who blame IT for organizational problems.

The Countervoice
"See? This is just an IT story."

It seems that way. Until you look at what is really happening.

15.2 Key message

USM often starts with IT.
But it fails as soon as it stays there.

Those who reduce USM to 'better IT service management' miss the essence:

USM is Enterprise Service Management — or it is nothing.

15.3 Why IT is always the first mirror

🧭 In **Rivertown**, IT is the crossroads:
* every service affects systems
* every disruption becomes visible
* every chain break ends up at the service desk.

Not because IT performs poorly.
But because IT exposes the interdependence.

The Countervoice
"But doesn't IT just have its own responsibility?"

Certainly. But no IT service delivers value on its own.

15.4 The classic misunderstanding: improving IT to fix the organization

🧭 **Rivertown** has already gone down this path:
* ITIL implemented
* tooling renewed
* processes refined
* SLAs improved.

The result?
- IT became more mature
- the organization did not.

Why?
Because you don't solve chain problems by perfecting a single link.

15.5 What changes when USM starts with IT

When USM starts with IT, something subtle but fundamental happens:
- IT stops optimizing for itself.
- IT starts optimizing for the service.

That means:
- Defining services outside of IT
- Making dependencies explicit
- Establishing chain-wide agreements.

> 💬 ⚡ **The Countervoice**
> *"But then IT gets extra power, right?"*
>
> On the contrary. IT loses the privilege of implicit control.

15.6 The transition that determines everything

🧭 The tipping point in **Rivertown** comes when someone says out loud:
"IT is not a supplier. IT is a link."

That one word changes everything.

If IT is a link:
- IT cannot solve everything
- IT cannot determine everything
- IT cannot bear everything.

But:
- IT becomes understandable
- predictable
- reliable within the chain.

15.7 From ITSM to ESM: not an expansion, but a repositioning

When it comes to Enterprise Service Management, many organizations think:
- "roll out ITSM (IT Service Management), or ITIL, to HR"
- "use the same tool"
- "copy processes."

That is not what is happening here.

USM is not rolling anything out.
USM repositions.

🧭 In **Rivertown**, ESM means:
* HR manages its services using the same process logic.
* Social Domain uses the same workflows.
* Environment Management speaks the same language about change and recovery.

💬⚡ The Countervoice
"But isn't HR fundamentally different from IT?"

In implementation: yes. In management: no.

15.8 Why ESM only works with architecture

Without SMA, ESM becomes:
- a tool project
- a process copy
- a political battle.

With SMA, ESM becomes:
- logical
- scalable
- inevitable.

Because:
- everyone uses the same steering wheel
- no one makes up their own rules
- differences only exist where they make sense in practice.

15.9 The role of suppliers in this whole

🧭 In **Rivertown**, something becomes painfully clear: many suppliers actually have more influence than is desirable.

Not out of malice.
But because:
* they have the knowledge
* they manage the tooling
* they know the exceptions.

USM reverses this:
- Suppliers deliver within the SMA.
- Workflows are leading, not contracts.
- Architecture is owned by the organization.

💬⚡ The Countervoice
"But suppliers won't just accept that, will they?"

That's right.
And that's exactly why the organization has to design this itself.

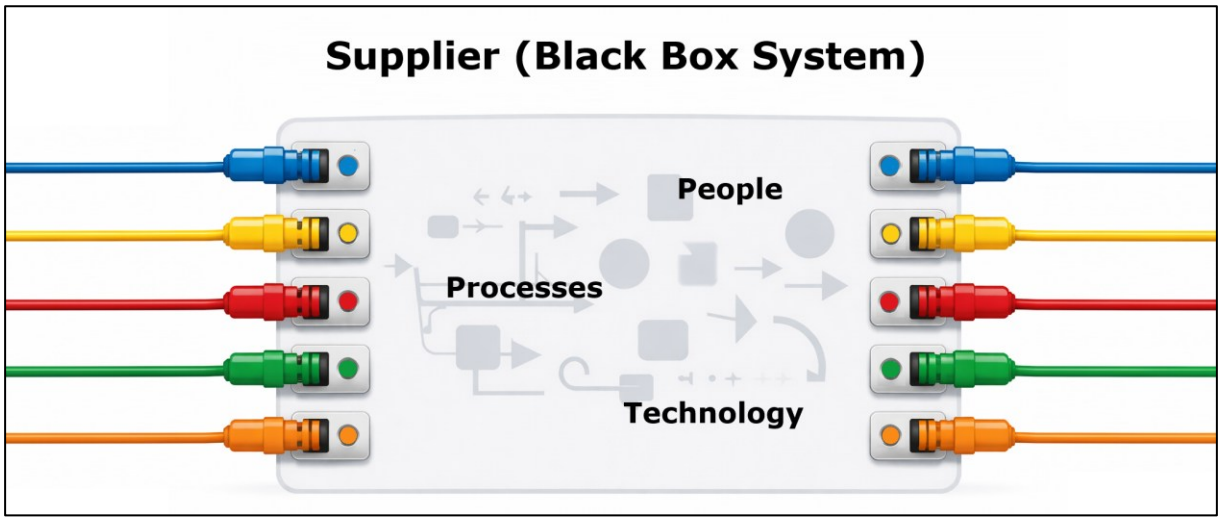

Figure 17. A supplier can be regarded as a black box that organizes its own people, processes, and resources; it is the interface that determines how the black box fits into the system

15.10 What ESM delivers in concrete terms in Rivertown

As soon as USM is applied enterprise-wide:
- transfer conflicts disappear
- escalations become rarer
- predictability arises.

Not because everything is perfect.
But because everyone knows:
- where they contribute
- where they stop
- and where the whole is monitored.

15.11 Reflection questions

- Where does USM start in your organization — and where should it end up?
- Which services are currently wrongly regarded as 'IT problems'?
- Which teams are already functioning as links?

15.12 Learning objectives

Having read this chapter, the reader will be able to:
- explain why USM often starts with IT but should not end there
- identify the difference between ITSM and ESM
- explain why ESM only works with a shared architecture
- recognize how IT is shifting from problem carrier to reliable link.

15.13 Looking ahead

Now that USM is positioned organization-wide, the next question is inevitable:

How does this change things for managers?

Not in theory.
But in their daily work.

In Chapter 16, the focus shifts:
- from structure to behavior
- from design to leadership.

That's where it gets personal.

16 What will change for managers?

By 'manager', we mean anyone who makes decisions that affect the coherence of service delivery — regardless of position.

Coordination is not inferior work —
it is simply not a substitute for management.

16.1 Context & recognition

🧭 Up to this point, a manager in **Rivertown** might have thought: "Interesting. Logical. Good story."

But now comes the question that no one asks out loud, but that everyone feels: "And what does this mean for me?"

Not for the organization. Not for IT. Not for 'the transition'.

But for:
* your agenda
* your decisions
* your position.

💬 ⚡ The Countervoice
"This all sounds great, but I was hired to get results, not to become an architect."

That's exactly where the shift lies.

16.2 Key message

USM doesn't change management by adding extra tasks, but by introducing a fundamentally different responsibility: *ownership of the system.*

Not of people.
Not of teams.
But of cohesion.

16.3 From firefighter to system manager

🧭 In **Rivertown**, managers spend a large part of their time on:
* escalations,
* coordination
* exceptions
* recovery after mistakes.

That feels like management.
But in reality, it is damage control.

USM shifts that role:

Less:
- ad hoc intervention
- mediating between interests
- "just sorting it out".

More:
- asking design questions
- monitoring boundaries
- enforcing logic.

The Countervoice
"But problems never completely disappear, do they?"

No. But they don't always have to land on your desk.

16.4 Shifting decisions from content to framework

One of the biggest changes for managers is this:

You decide less about what teams should do,
and more within which frames they decide.

In **Rivertown**, that means:

Not:
* substantive discussions about tools
* interference with routines
* exceptions "because it has to be done now".

Instead:
* sticking to service definitions
* monitoring process logic
* rejecting solutions that damage the system.

The Countervoice
"But then I'll lose control, won't I?"

No. You lose detail—and you gain overview.

16.5 Less power, more authority

USM asks managers to do something paradoxical: relinquish informal power.

No more:
- making decisions based on position
- steering through back doors
- solving problems through personal intervention.

Instead:
- making decisions traceable
- letting architecture speak
- being consistent, even when it's uncomfortable.

At first, this feels like a loss of authority.
In practice, however, it creates trust.

The Countervoice
"But what if people don't comply?"

Then it's not a behavioral problem.
Then it's a design flaw or a leadership choice.

16.6 The manager's agenda changes

🧭 Managers in **Rivertown** notice something striking over time: Their agenda is becoming calmer.

Not emptier. Calmer.

Less:
* crisis meetings
* escalation reports
* coordination meetings.

More:
* system improvement
* decision-making on broad outlines
* reflection on chain performance.

USM makes management a profession again, instead of a series of incidents.

16.7 What USM explicitly does not ask of managers

USM does not ask:
- that you understand everything
- that you are on top of everything
- that you become an expert in every domain.

It does ask that you take the system seriously and refuse to undermine it 'for convenience'.

💬 ⚡ The Countervoice

"But sometimes you have to be pragmatic, right?"

Pragmatism without architecture is just postponing misery.

16.8 The uncomfortable mirror

USM exposes something that many managers would rather not examine: Part of the daily hustle and bustle is the result of their own interventions.
- allowing exceptions
- bending rules
- personally resolving escalations.

USM asks for something more difficult: doing nothing when the system needs to learn.

That is not passivity. That is leadership.

16.9 Reflection questions
- Which problems keep recurring in your agenda?
- Where do you intervene because the system is failing?
- Which decisions do you make that should actually be supported by architecture?

16.10 Learning objectives

Having read this chapter, the reader will be able to:
- explain how USM fundamentally changes the role of managers
- identify the difference between content-based management and frame-based management
- recognize why informal power perpetuates system failure
- understand why peace on the agenda is a sign of maturity.

16.11 Looking ahead

When managers start managing differently, something inevitably changes for teams.

Not because they “have to work differently”, but because the system in which they work is changing.

In Chapter 17, we look at the other side: what changes for teams—and why that is a relief.

This concludes Part VI.
And then it gets really practical.

17 What will change for teams?

17.1 Context & recognition

🧭 In **Rivertown**, teams have been ‘changing’ for years.

They have:
* learned new routines
* gained additional consultation structures
* taken on more coordination tasks
* and placed more and more responsibility ‘low in the organization’.

And yet, many teams feel overburdened, dependent, and structurally inadequate.

Not because they are unable to do their work, but because they have to compensate for the system.

💬⚡ The Countervoice
“But shouldn't teams just take more ownership?”

That sounds logical.
Until you look at what that ownership actually entails.

17.2 Key message

USM relieves teams by freeing them from system problems that were never their responsibility.

Teams don't change because they start working differently, but because the system finally does its job.

17.3 Less coordination, more flow

🧭 In **Rivertown**, teams spend a significant amount of their time:
* coordinating with other teams
* explaining what they do
* waiting for decisions
* fixing mistakes made elsewhere in the chain.

This is often labeled ‘collaboration’, but in practice it is compensatory cooperation.

USM reveals this for what it really is: friction caused by a lack of architecture.

When:
- services are clearly defined
- workflows are predictable
- transfers are explicit...

much of that coordination disappears automatically.

💬⚡ The Countervoice
“But consultation is necessary, isn't it?”

Certainly. But not to fill structural gaps.

17.4 Clear responsibilities, finally

For teams, this is perhaps the biggest change.

Without USM:
- responsibilities overlap
- ownership is diffuse
- mistakes are passed on.

With USM:
- it is clear where a team starts
- where it stops
- and where it transfers.

This feels strict at first.
But soon it also feels safe.

Teams need to:
- defend less
- escalate less
- improvise less.

> **The Countervoice**
> *"But aren't teams then being judged on things beyond their control?"*
>
> On the contrary. They are actually protected from that injustice.

17.5 Collaborating without extra pressure

🧭 A striking effect in **Rivertown**: teams experience more collaboration, but less collaboration stress.

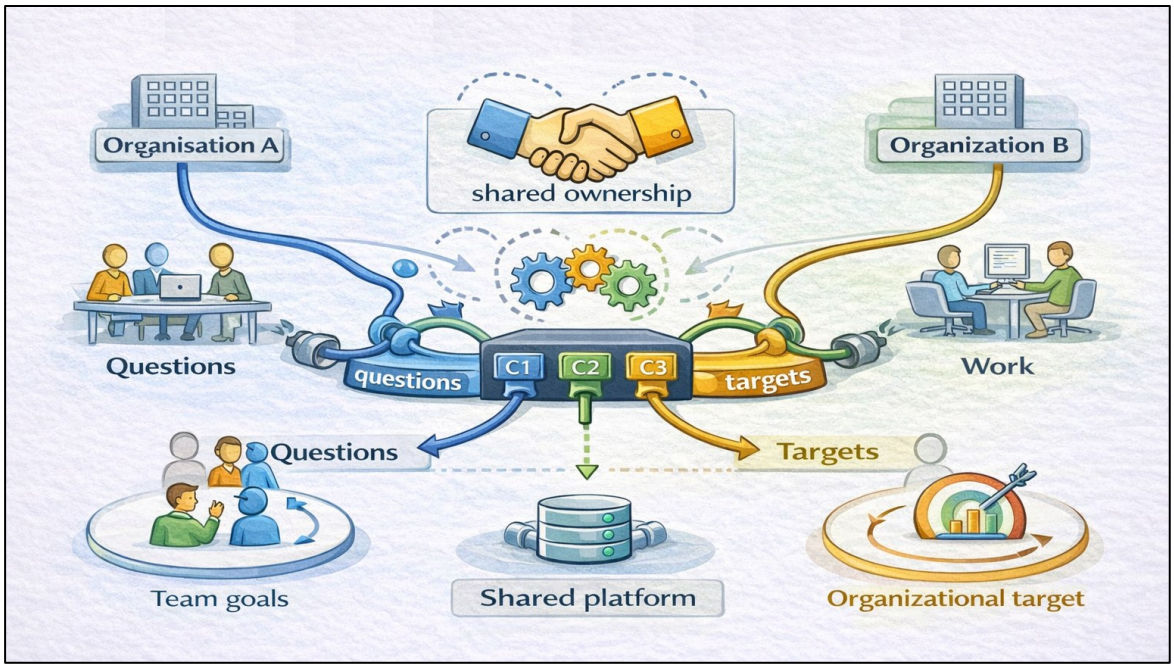

Figure 18. Collaboration does not arise from coordination, but from shared interfaces and clear goals.

Why? Because collaborating no longer means:
- being involved in everything
- knowing everything
- filling in the gaps.

Collaboration now means:
- delivering reliably within a clear link
- relying on other links
- letting problems land where they belong.

That is not naivety.
That is professional trust.

17.6 Craftsmanship is given space again

Many professionals have come to confuse their profession with:
- forms
- checks
- exceptions
- explaining things to others.

USM reverses this.

Because:
- the system monitors consistency
- teams no longer have to do so.

This creates space for:
- substantive quality
- learning within the profession
- real improvement.

The Countervoice

"But some people thrive in that complexity."

That's true. But organizations are not playgrounds for heroism.

17.7 Fewer heroes, more reliability

USM is bad news for:
- fire extinguishers
- informal fixers
- 'indispensable' links.

And good news for:
- stable teams
- predictable performance
- sustainable quality.

Teams no longer have to compensate for what was never designed.

That's not leveling.
That's professionalism.

17.8 The psychological shift

Perhaps the most underestimated effect: Teams feel less personally affected by system failures.
- A malfunction is not a lack of commitment.
- A delay is not a lack of involvement.
- A chain problem is not a team problem.

That takes away pressure.
And pressure is the greatest enemy of quality.

The Countervoice

"But doesn't that reduce the urgency?"

No. It shifts from emotion to responsibility.

17.9 Reflection questions

- Where do teams in your organization compensate for system errors?
- Which coordination tasks only exist due to a lack of architecture?
- Where would clear boundaries actually bring peace of mind?

17.10 Learning objectives

Having read this chapter, the reader will be able to:

- explain why USM relieves teams instead of burdening them
- recognize how clear links simplify collaboration
- explain why craftsmanship benefits from system design
- understand why less heroism forces better performance.

17.11 Looking ahead

Chapter 17 concludes Part VI.

We have seen what USM means for IT, for managers, and for teams.

The next step is inevitable and practical:

How do you get this moving without it getting bogged down in misery?

Part VII is about:

- starting small
- choosing wisely
- avoiding pitfalls.

There, USM is no longer a vision, but an achievable reality.

PART VII – DEPLOYMENT WITHOUT HASSLE

This is the point in the book where we stop tinkering with individual parts and look at the bigger picture again. Not yet another model. Not yet another checklist. But the question we dare to ask too little along the way: “What are we actually doing?”

The previous sections were about structure, roles, architecture, agreements. About how to get a grip on an ecosystem that behaves like a bowl of spaghetti under high tension. Necessary work. But dangerous too. Because before you know it, the system becomes more important than the intention.

In this part, we zoom out. We look at what all those building blocks do together. Where they create value — and where they unintentionally cause friction. We connect practice with principle, and governance with common sense.

No final conclusion, but a recalibration. No answer, but a direction.

Because if you only optimize what's in front of you, you miss the system you're part of.

Figure 19. How to apply USM without causing friction

18 Start small, make a big impact

18.1 Context & recognition

🧭 In **Rivertown**, the analysis is complete. The architecture is clear. The division of roles is understood.

And then doubt strikes.

Not in terms of content.
But in practical terms:
* “This is too big.”
* “This affects everything.”
* “We don't have the capacity for this.”

The reflex is familiar:

- first make a plan,
- then create collective buy-in,
- then start a program,
- and then... delay.

💬⚡ The Countervoice
“This isn't a pilot project. You have to get it right the first time.”

That sounds mature. But it's exactly what paralyzes us.

18.2 Key message

USM doesn't work by starting big, but by really letting the system speak for itself in one place.

Not everything at once. Not organization-wide.

But fundamentally correct, on a small scale.

18.3 Why starting big almost always fails

🧭 **Rivertown** has experience with large projects:
* reorganizations
* transformations
* programs with steering groups.

What usually happens there:
* the design is watered down
* exceptions are built in
* political compromises pile up.

The result is rarely ill-intentioned.
But it is always half-hearted.

💬⚡ The Countervoice
“But if you start small, no one takes it seriously.”

Incorrect. People take it seriously if it works.

18.4 What 'small' really means here

Small does not mean:
- non-committal
- temporary
- experimental.

Small means:
- one service
- one chain
- one clear owner.

> In **Rivertown**, that could be:
> * a move
> * a benefit application
> * a permit request.

Not because they are simple, but because they are visible.

18.5 The power of one well-designed service

When a service in Rivertown is set up according to USM:
- with one service definition
- with the five processes
- with the eight workflow patterns
- with explicit links...

something remarkable happens.

Not only does that service improve.

For the first time, the organization sees:
- where friction exists
- where decisions are lacking
- where power was implicit.

> **The Countervoice**
> *"But then you only solve one problem, right?"*
>
> No. You expose the system.

18.6 Learning by doing, not by explaining

USM is difficult to explain. But extremely easy to recognize when it works.

> In **Rivertown:**
> * coordination problems disappear
> * escalations become rarer
> * better conversations arise.
>
> Not because everyone is convinced, but because the alternative is visibly worse.

This is not conviction. This is experience.

18.7 Why USM shows quick results

USM does not eliminate problems. It eliminates redundancy.

This immediately produces:
- less transfer
- less repair work
- fewer misunderstandings.

Even skeptics say, "This is calmer."
And calm is scarce. So it is valuable.

> 💬⚡ **The Countervoice**
> *"But doesn't this still take time and energy?"*
>
> Yes. But less than constantly compensating.

18.8 When to scale up (and when not to)

The biggest mistake after a successful start: wanting to roll out too quickly.

USM scales not by copying, but by repeating with understanding.

> 🧭 In **Rivertown**, that means:
> * only expand once the first service is stable
> * only broaden once the architecture is monitored
> * only accelerate once exceptions are visible.

Haste is not progress here.
It is sabotage.

18.9 What managers really need to do here

In this phase, leadership is boring. And that is precisely why it is effective.

Managers must:
- protect the design
- refuse exceptions
- resist pressure to "just be practical".

> 💬⚡ **The Countervoice**
> *"But sometimes you have to go with the flow, right?"*
>
> Yes. But not at the expense of the steering wheel.

18.10 Reflection questions

- Which service in your organization is visible enough to start with?
- Where would one well-designed chain be most visible?
- What pressure are you already feeling to water down the design?

18.11 Learning objectives

Having read this chapter, the reader will be able to:
- explain why USM must start small
- recognize what a suitable starting service is
- explain why visible success is more important than collective buy-in
- understand why scaling up requires discipline.

18.12 Looking ahead

Starting small only works if knowledge remains in-house.

The next chapter therefore presents a fundamental choice: train or hire.
Not as a question of cost, but as a question of sustainable control.

In Chapter 19 – The role of internal staff, that choice becomes inevitable.

19 The role of internal staff

19.1 Context & recognition

🧭 In **Rivertown**, sooner or later, the same question always comes up:
"Do we do this ourselves... or do we hire expertise?"

Not out of laziness. But out of experience.
* "This is complex."
* "We've never done this before."
* "We want to get it right the first time."

And so, before long, we see:
* an external agency
* a roadmap
* a program
* a set of deliverables.

💬⚡ The Countervoice
"But we simply don't have that expertise in-house."

That sounds rational. And that's exactly where the problem lies.

19.2 Key message

USM only works *sustainably* if ownership and understanding lie within the organization itself.

Not because external help is bad, but because management cannot be outsourced.

19.3 The role of the USM coach (external or internal)

USM does not rule out external help.
But it radically redefines its role.

The USM coach — external or internal — is not an implementer.
The coach:
- does not take ownership
- does not make design choices for the organization
- and does not manage processes.

The USM coach *coaches*.

In concrete terms, this means:
- He helps internal management make the right decisions.
- He contributes experiences and examples from similar situations.
- He asks questions that sharpen the manager's thinking.
- He highlights which choices are in line with USM — and which are not.

The goal is not for the coach to be right.
The goal is for the manager to do better *themselves* next time.

What the coach *does not* do

This must be clear to both the organization and the coach. The coach is:
❌ not making draft decisions
❌ not legitimizing exceptions
❌ not making substantive choices
❌ not becoming the owner of USM.

What the coach *does* do

A USM coach is successful when he becomes redundant.

19.4 Why organizations reflexively hire

🧭 In **Rivertown**, hiring has become logical:
* consultants bring pace
* they speak the language of administrators
* they provide structure and guidance.

And to be fair, they are often good.

But this pattern has a structural side effect.

💬⚡ The Countervoice
"But without external consultants, we'll never get started, right?"

Maybe. But with external consultants, you rarely get started on your own.

19.5 What happens when external consultants carry the design

When external consultants:
- create the design
- explain the architecture
- interpret exceptions...

something subtle but disastrous happens:

The organization learns what has been decided, but not why.

As soon as the external party leaves:
- the logic fades
- exceptions return
- the system is modified again.

Not out of unwillingness. But out of misunderstanding.

That is exactly where the USM coach plays his role.

Not by taking over, but by pausing at decision moments and asking:
- "What choice are you making here?"
- "What logic follows from this?"
- "What does the system learn from this?"

So that understanding arises during the decision-making process, not only afterwards.

19.6 USM does not require specialists, but carriers

USM is not a trick.
It is a way of looking at things and making decisions.

The USM coach:
- does not focus on implementation
- does not focus on speed
- does not focus on comfort.

He only intervenes when decisions:
- undermine the architecture
- normalize exceptions
- or deviate from the USM goal.

Not by intervening, but by referring the decision back to the manager.

This does not require:
- full-time USM experts
- consultants as a permanent intermediary layer.

It requires:
- people who understand the system
- who monitor the design
- and who have the courage to say ‘no’.

> **The Countervoice**
> *“But then we're putting a lot of pressure on a few people.”*
>
> That's right. And that's why they have to be the right people.

19.7 Who should do this?

In **Rivertown**, it rarely turns out to be the *usual suspects*.

Not necessarily:
* project managers
* change managers
* process experts.

But:
* experienced managers with an overview
* professionals with authority
* people who see systems instead of tasks.

USM does not land on job titles. It lands with mental maturity.

19.8 Training is not an expense, but risk mitigation

Many organizations see training as:
- something time-consuming
- something you do ‘besides work’
- something you can postpone.

In the context of USM, the opposite is true.

Not training means:
- remaining dependent
- being unable to defend the design
- falling back into old patterns.

> **The Countervoice**
> *“But training takes time that we don't have.”*
>
> That's exactly why you should stop outsourcing.

19.9 The right role for external parties

USM is not anti-external parties. On the contrary. But their role is fundamentally changing.

External expertise is suitable for mirroring, training, challenging, and temporary acceleration.
Not for ownership, design decisions, or structural management.

In **Rivertown**, external help only works if it makes itself redundant.

19.10 When a USM coach does and does not intervene

The USM coach does *not* intervene when:
- things get exciting
- resistance arises
- the pace slows down
- or discussion is needed.

These are learning-symptoms.

The USM coach does intervene when:
- architecture is exchanged for pragmatism
- structural exceptions are allowed
- or decisions fragment the system again.

Not with solutions, but with one question: *"Is this decision in line with the USM goal?"*

19.11 The tipping point: from project to profession

The moment when USM really takes hold is when people say:

"This is no longer a project. This is how we manage."

You can't hire that moment. But you can guide it.

Not by someone who does it *for* you, but by a coach who helps you learn to do it *yourself* — and thereby makes themselves redundant.

19.12 Reflection questions

- What knowledge in your organization is currently outsourced?
- Where are decisions dependent on people outside the organization?
- Who could be the natural carriers of USM?

19.13 Learning objectives

Having read this chapter, the reader will be able to:
- explain why USM ownership must be internal
- identify the difference between supporting and taking over
- explain why training is crucial for sustainability
- recognize who within the organization can be a carrier of the system.

19.14 Looking ahead

Now there is one important topic left before we conclude the book: *what usually goes wrong — even with the best of intentions?*

In Chapter 20 — Common pitfalls, we expose the final stumbling blocks.

This concludes Part VII and prepares us for the leap to the final picture.

20 Pitfalls as system behavior

Why good intentions can undermine USM.

In **Rivertown**, USM is no longer just an idea, but a reality:
* there is an SMA
* services have been appointed
* managers are managing differently
* teams are experiencing peace of mind
* and there is coaching instead of takeover.

And yet...

It is precisely now that things sometimes go wrong.
Not spectacularly. Not visibly. But insidiously.

The Countervoice
"But we're doing everything by the book, aren't we?"

That is precisely where many pitfalls begin.

This chapter does not describe mistakes. It describes **logical derailments**.

Almost all the pitfalls that organizations encounter when working with USM are rational, explainable, and even well-intentioned. They do not arise because people misunderstand USM, but because they apply new insights **within an old framework**.

That is why pitfalls are not presented here as warnings ("don't do this"), but as **patterns of system behavior** that occur when design and application do not coincide.

Those who recognize these patterns are not 'too late' — but are precisely at the tipping point for which this book was written.

20.1 The IT-start pitfall

Reducing USM to an IT improvement

One of the most common derailments is starting with USM **within IT**, with the implicit assumption that success there can later be scaled up automatically.

That seems logical:
- IT has experience with service management.
- IT has the tools.
- IT feels like the 'logical engine' for cohesion.

The problem is not that IT is starting.
The problem is **what IT is then supposed to be**.

When USM is positioned as:
- an ITSM improvement
- a new process model for IT
- or a maturity step within ITIL/ESM...

a structural distortion arises: the **SMA becomes the property of a single domain**, while it is intended as a **control principle for the whole**.

The result:
- other domains do not recognize themselves
- management remains out of the picture
- cohesion is once again compensated for instead of designed.

USM is then seen as 'something IT does' and loses precisely what it aims to restore.

20.2 The coordination pitfall

New architecture, old behavior

A second pitfall arises when USM is understood in terms of content but **not translated into organizational terms**.

Managers recognize the problem of fragmentation but continue to:
- resolve exceptions
- organizing coordination
- smoothing out escalations.

USM is then added **in addition to** existing coordination patterns, rather than replacing them.

As a result, the system learns nothing new.
It only learns that:

"When things get complicated, people will solve them."

Coordination remains dominant, but now with USM terminology.
This is not an implementation problem — it is a **design conflict**.

As long as management does not explicitly opt for:
- system responsibility
- role redistribution
- and the phasing out of repair work...

USM will remain an extra layer, not a tipping point.

20.3 The tool-first pitfall

Automating what has not been designed

A classic reflex is to quickly set up tooling:
- service portals
- workflows
- dashboards
- integrations.

Tooling feels tangible.
Tooling shows progress.

But without explicit choices about:
- service definitions
- interfaces
- responsibilities
- and decision moments...

tooling automates **uncertainty**.

What emerges is not coherence, but:
- faster escalations
- better registration of failures
- and more dependence on exceptions.

USM first asks for **design decisions**, only then for tooling.

Those who reverse that order will end up with a professional-looking system that continues to malfunction structurally.

A fool with a tool is still a fool.

20.4 The ownership pitfall

Giving everything an owner

In response to fragmentation, people look for ownership:
- service owners
- chain owners
- process owners.

Ownership sounds powerful.
But without clear architecture, it forces:
- overlapping responsibilities
- mutual claims
- and ambiguity about decision-making authority.

USM does not replace ownership with more ownership, but with:
- explicit role distribution
- clear interfaces
- and a limited number of controllable objects.

When everything has an owner, ultimately **no one is responsible for the whole**.

20.5 The scale pitfall

Too big too fast

When an initial application works, enthusiasm arises:

"We have to do this everywhere."

Scaling up before:
- the design is stable
- roles have crystallized
- and management has internalized the new thinking...

reproduces old patterns on a larger scale.

USM does not scale through repetition, but through **recognition**: only when people see why it works can it be implemented more broadly.

20.6 The moral pitfall

Framing pitfalls as failure

Perhaps the most dangerous pitfall is this: attributing pitfalls to unwillingness, resistance, or incompetence.

This forces:
- tighter control
- more explanations
- and pressure on people.

Whereas pitfalls actually show where the system has not yet been designed consistently.

USM only works when mistakes are interpreted as **design signals**, not as personal shortcomings.

20.7 The best practice pitfall

We are unique again (or still)

Over time, there is a tendency to say:
- "This is how we do USM here."
- "This is our variant."
- "That fits better with our culture."

Before you know it:
- local interpretations arise
- processes deviate
- and the shared logic disappears.

USM then changes from architecture to agreement.

> **The Countervoice**
> *"But isn't every organization unique?"*
>
> Certainly. But management does not have to be.

20.8 The coach pitfall

Using the coach as a safety net

A subtle but treacherous pitfall: the USM coach becomes the last resort.
- "Let's run this by the coach."
- "What does he think about this?"
- "Can't he just decide this?"

At that moment:
- ownership shifts
- learning stops
- and coaching becomes dependency.

A good coach senses this and withdraws.

20.9 Normalizing exceptions

But this is really different

Every organization has exceptions. USM does not deny that.

But when exceptions:
- become structural
- are not evaluated
- or do not lead to system improvement...

they become the new normal.

Then complexity returns.
Not suddenly. But inevitably.

> **The Countervoice**
> *"But this exception is really special."*
>
> That's what all exceptions say.

20.10 Confusing results with success

We have successfully implemented USM

Over time, the figures improve:
- lead times decrease
- escalations decrease
- satisfaction increases.

The danger: people think it is 'finished'.

USM is not a project with an end date.
It is a management system that continuously produces improvements.

As soon as attention wanes:
- improvisation increases
- architecture fades.

20.11 Relapsing under pressure

Politics wins the battle

The ultimate stress test comes with:
- political pressure
- incidents
- crises
- media attention.

Then the temptation is great to:
- bend the rules
- allow exceptions
- circumvent the system 'for now'.

That one moment determines whether USM is truly embedded.

The Countervoice

"But sometimes the situation is just too urgent."

That is precisely when leadership comes into play.

20.12 What these pitfalls have in common

The system is abandoned as soon as it becomes uncomfortable.

In all cases, the same thing happens: USM is applied within the existing paradigm, rather than replacing that paradigm.

Then it feels familiar, safe, manageable.

And the effect remains limited.

If you recognize these pitfalls:
- you haven't started off on the wrong foot
- you haven't done anything 'wrong'
- but you haven't started with USM yet.

USM only really begins when:
- the service is leading
- the architecture is explicit
- collaboration is self-evident.

20.13 Key message

Pitfalls are not deviations from USM.
They are **proof that old patterns are strong**.

Every pitfall raises the same question:

Where are we trying to apply new thinking without letting go of the old design?

Those who dare to ask that question are not off track — but exactly where real cohesion can arise.

USM does not demand perfection. It demands consistency. Not always the right decision. But always a traceable decision.

20.14 Reflection questions

- What forms of coordination are still necessary “because otherwise it won't work”?
- Where does it feel attractive to “deviate for a moment”?
- Who monitors the design when things get exciting?

20.15 Learning objectives

Having read this chapter, the reader will be able to:

- explain why USM fails despite good intentions
- recognize the most common relapse patterns
- explain why discipline is more important than enthusiasm
- understand where leadership really shows itself in crisis situations.

20.16 Transition

Chapter 20 concludes Part VII.

Everything needed to make USM work is now on the table:

- design
- roles
- start strategy
- coaching
- and pitfalls.

What remains is the final picture. In Part VIII, we look ahead: not to an ideal, but to an organization that functions well in a boring way.

This concludes the book.

PART VIII – The organization beyond silos

Organizations are champions at building silos. Teams optimize their own little patch of land, defend their boundaries, and call it ‘efficiency’. Meanwhile, service quality sinks into the surf between those silos, where no one is in charge and everyone passes the buck.

In this section, we leave behind the map of cubicles and functions. We look at the organization as customers experience it: as a single coherent system, or as a chain of interruptions. Here, it is no longer about teams, but about connections. Not about who sits where, but about how work actually flows.

Those who dare to look beyond the silos discover something uncomfortable: collaboration is not a soft skill, but a design choice. And you can make that choice.

Figure 20. The organization beyond silos

21 What does an organization without silos look like?

21.1 Context & recognition

Anyone who has read this book up to this point probably has two conflicting feelings.

On the one hand:
- "Yes. This is correct."
- "This explains a lot."
- "This could indeed be different."

On the other hand:
- "But does such an organization actually exist?"

That is a valid question.

> 💬⚡ **The Countervoice**
> *"This is starting to sound very idealistic."*
>
> That is precisely why we need to keep it concrete.

21.2 Key message

An organization without silos is not spectacular.
It is predictable, calm, and sometimes even a little boring.

And that is precisely why it works.

21.3 What you don't see in an organization without silos

Let's start with what's missing.

🧭 In **Rivertown** — over time — you won't see:
* endless coordination meetings
* structural escalations
* "that's their job" discussions
* heroes filling in the gaps
* managers who constantly have to make adjustments.

Not because people have become better, but because the system no longer provokes that behavior.

> 💬⚡ **The Countervoice**
> *"But problems never completely disappear, do they?"*
>
> No. They just end up where they belong.

21.4 One face to the citizen

For the citizen, client, or patient, something fundamental changes: the organization feels like a single entity.
- questions are not passed on
- answers are consistent
- mistakes are corrected without discussion.

Not because there is a single point of contact, but because behind the scenes, the chain is managed as a chain. Citizens do not notice any teams. And that was always the intention.

21.5 Internal calm is not laziness

🧭 A striking effect in **Rivertown**: internal calm is increasing.
* less ad hoc consultation
* fewer moments of crisis
* less tension between teams.

This is sometimes confused with:
- lack of urgency
- loss of focus
- "we have become very comfortable".

The Countervoice
"But without pressure, things will slacken, won't they?"

No. Pressure is a poor substitute for direction.

21.6 Predictable performance in complex chains

In an organization without silos:
- performance is not perfect
- but it is predictable.

That means:
- deviations are spotted early
- decisions are traceable
- improvements affect the system.

Management no longer focuses on incidents, but on patterns.

That is not control. That is maturity.

21.7 Less policy, more action

An unexpected consequence: the number of rules decreases.

Not because rules are being scrapped, but because many rules were created to compensate for system errors.

As soon as:
- transfers are clear
- responsibilities are explicit
- architecture is leading...

the need for micromanagement disappears.

The Countervoice
"But then we're letting things go?"

Yes. Things that are no longer necessary.

21.8 Technology as a servant, not a helmsman

🧭 In **Rivertown**, the role of technology is shifting:
* less discussion about tools
* less vendor dependency
* more focus on supporting the service.

Systems support workflows instead of dictating them.

This finally makes technology replaceable again.

21.9 People remain people

An organization without silos is not paradise.
There are still differences of opinion, mistakes, political tensions, difficult choices.

The difference: these take place within a clear framework.
Conflicts are resolved more quickly, are less personal, and are easier to interpret.

The Countervoice
"So culture no longer matters?"

Culture follows structure. Always.

21.10 Why this end goal is so difficult to sell

Because it contains no heroics.
- no grand transformation stories
- no revolutionary breakthroughs
- no permanent change.

It is an organization that does what it promises.

No more. No less. And strangely enough, for many organizations, that is revolutionary.

21.11 Reflection questions
- How would your organization feel if escalations were rare?
- Which tensions exist solely because of unclear cohesion?
- What would happen if peace and quiet were no longer a problem?

21.12 Learning objectives

Having read this chapter, the reader will be able to:
- sketch a realistic picture of an organization without silos
- explain why calm and predictability are signs of maturity
- recognize why heroism is giving way to reliability
- understand why simplicity is underestimated.

21.13 Preview

We are almost at the end.

In the final chapter, the focus shifts one last time: from organization to personal leadership.

Not as a style, but as a responsibility.

In Chapter 22 – The manager as architect, we come full circle.

22 The manager as architect

(and why coordinating is different from managing)

22.1 Context & recognition

🧭 In **Rivertown**, many people hold the title of manager. But in daily practice, they do something else.

They:
* coordinate
* resolve bottlenecks
* mediate between teams
* monitor agreements
* follow up on actions.

That is not a criticism.
It is an observation.

The Countervoice
"But isn't that management?"

No. That is coordination.

22.2 Key message

Many managers function as coordinators because there is no management system.

And as long as that is the case:
- they will continue to put out fires
- they will personally maintain cohesion
- and the system will never mature.

That is why USM does not start with better coordination, but with learning what management really is. If you hadn't understood this yet, you'd better go back to section 8.6 and chapter 16 and read again why this is a paradigm shift.

22.3 Coordination: a necessary symptom, not a solution

Coordination arises when:
- work crosses boundaries
- responsibilities are unclear
- interfaces are not designed.

🧭 In **Rivertown**, coordination is not a choice.
It is a necessity.
Without coordination:
* work comes to a standstill
* citizens lose their way
* escalations arise.

But that does not make coordination management.

The Countervoice
"Without me, everything falls apart."

That's right. And that is precisely the problem.

22.4 What management is (and coordination is not)

Management is not:
- solving exceptions
- smoothing out conflicts
- connecting loose ends.

As long as a manager is needed to:
- keep workflows together
- translate decisions
- fill gaps...

the system is not managed, but compensated.

Management is: designing and monitoring a system in which coherence is structurally organized.

22.5 Why managers must first relearn what management is

Many managers are highly trained in:
- communication
- leadership
- change management
- stakeholder management.

But hardly at all in:
- system design
- architecture
- governability.

USM therefore asks for something radical: not better leadership, but a better understanding of management.

The Countervoice

"But isn't that what architects are for?"

Architects design the system.
Managers are responsible for its functioning.

22.6 Understanding architecture is not a specialism, but a prerequisite

The manager as architect does not need to:
- draw models
- make diagrams
- master methods.

But they do need to:
- understand how the SMA works
- know where interfaces run
- recognize when coordination replaces architecture.

Without that insight:
- a manager cannot steer
- only adjust.

22.7 The crucial shift: from doing it yourself to having it done

USM requires a clear division of roles:
- **Managers** design, monitor, and improve the management system.
- **Coordinators** apply that system in daily operations.

That is not a demotion.
It is professionalization.

> **The Countervoice**
> *"But then I become dependent on others."*
>
> No. You finally become dependent on the system, not on individuals.

22.8 Why this separation is liberating

When managers stop coordinating:
- space is created to observe
- pattern recognition becomes possible
- structural improvement can take place.

And when coordinators:
- apply a clear system
- work within clear workflows
- don't have to improvise...

execution becomes calmer and more reliable.

This is not a hierarchical intervention.
It is a logical separation of responsibilities.

22.9 The uncomfortable truth

As long as managers:
- are proud of their indispensability
- derive satisfaction from resolving chaos
- and their schedules are filled with coordination...

the system remains dependent and fragile.

USM invites managers to give up something: the role of human glue.

> **The Countervoice**
> *"But that's what I'm good at."*
>
> Perhaps. But organizations don't need glue.
> They need architecture.

22.10 This is where the manager truly becomes an architect

Not through:
- more influence
- more decision-making power
- or more visibility.

But through:
- limiting coordination
- letting the system speak for itself
- and letting the implementation do what it needs to do.

That is not a loss of status.
That is mature management.

22.11 Reflection questions

- Where in your role do you mainly function as a coordinator?
- Which problems exist only because the system is lacking?
- What would happen if you stopped compensating today?

22.12 Learning objectives

Having read this chapter, the reader will be able to:

- explain the difference between coordinating and managing
- recognize why coordination is a symptom
- explain why understanding architecture is essential for managers
- understand why execution and control must be separated.

22.13 Looking ahead

This chapter brings us full circle.

We started with silos as a system problem.
We end with management as a system responsibility.

The key question remains — deliberately uncomfortable:

The choice is no longer whether to coordinate — but whether coordination remains a human burden or becomes a system property.

This concludes the book.
Not with a method.
But with a choice.

23 Conclusion: break down the walls, not the people

23.1 Back to the beginning

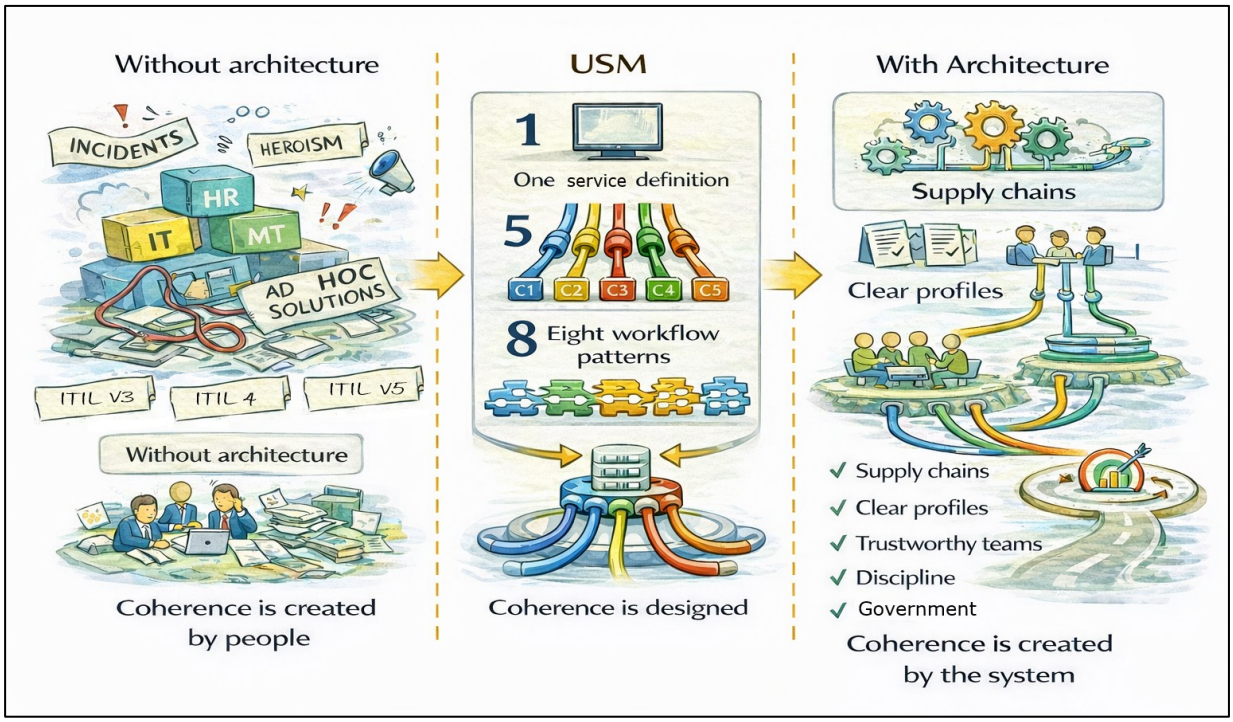

Figure 21. From compensating to managing. What was initially compensated for by people is ultimately supported by the system.

This book began with a simple observation: organizations work harder than ever, yet struggle to improve as a whole.
Not because people fail, but because coherence is left to be created by effort instead of design.

We noticed a familiar frustration: silos everywhere, good people getting stuck, organizations working harder but achieving less.

The reflex was familiar:
- cultural programs
- leadership training
- better cooperation
- more coordination
- more consultation
- yet another framework.

And always the same result.

Not because people didn't want to.
But because they were working in a system that forced them to fail.

The Countervoice
"But people make the organization, don't they?"

Yes. But systems determine what people can do.

"What felt like powerlessness in Chapter 1 now turns out to be a logical consequence of the lack of design."

23.2 Looking back at the beginning of this book

What felt like frustration, powerlessness, and organizational fatigue in Chapter 1 now turns out to have been something else: not a failure of people, but the predictable consequence of an organization without explicit design.

Where managers then had to compensate, organize consultations, and personally ensure cohesion, the final picture shows what happens when that same work is finally done by the system itself.

The problem has not disappeared.
It has been shifted — from people to architecture.
And that is exactly where it belongs.

23.3 What this book has really argued

This book was never about IT, processes, tools, or methods.

It was about one uncomfortable truth:

If an organization structurally fails to deliver,
it is almost always a design flaw — not a behavioral problem.

Silos are not a character trait.
They are a logical consequence of how we manage organizations.

Throughout this book, a distinction has been made between cooperation and collaboration.
Cooperation relies on people compensating for fragmentation.
Collaboration emerges when shared interfaces and clear goals make such compensation unnecessary.

23.4 Why fixing people is so appealing

Addressing people feels faster, more personal, more decisive.
You can have conversations, plan training sessions, express expectations.

And in the meantime, nothing fundamental changes.
Because fixing people is easier than:

- redesigning power
- clarifying responsibilities
- enforcing architecture.

The Countervoice

"But system change is slow and complicated."

That's true.
But changing people without changing the system is pointless.

23.5 What USM does differently

USM does not promise harmony.
USM promises manageability.

It says:

- define services as chain performance
- design a single management architecture for service delivery
- use minimal, shared logic
- separate control from execution
- and make power visible.

Not to restrict people, but to finally let them do honest work.

23.6 The real paradigm shift

The core shift is this:
- from coordinating to managing
- from improvising to designing
- from heroism to reliability
- from local optimization to chain value.

That doesn't require better people. It requires better choices.

> **The Countervoice**
> *"But this takes courage."*
>
> Yes. More than any other method.

23.7 What this requires of you as a reader

If you put this book down thinking: *"Interesting, but difficult"*
then you have understood it — but not yet accepted it.

This book asks for something concrete:
- that you stop compensating
- that you stop trying to fix people
- and that you start designing the system.

Not tomorrow. Not on a grand scale. But somewhere. Consciously. Consistently.

23.8 Start small. But really start.

Choose:
- one service
- one chain
- one place where you make the steering wheel visible.

Show that:
- coherence can be designed
- calmness is not weakness
- and simplicity is not naivety.

That moment changes conversations.
And conversations change organizations.

> **The Countervoice**
> *"What if we get stuck halfway?"*
>
> At least you'll know why. And that's already a win.

23.9 Finally

Silos don't disappear by working harder.
They don't disappear by better cooperation.
They don't disappear by changing people.

Silos disappear when we finally dare to recognize that organizations need to be designed — not coordinated.

So don't break the people. Break the walls.

And build something that will remain standing, even when you're not there.

The End.

Epilogue – This book is not an end point

This book was not written to prove a point.
It was written to loosen something that had been stuck for too long.

When you finish this book, you won't know any new tricks.
You won't have a checklist.
You won't have a roadmap.

What you will have is a different frame of reference.

You will no longer be able to *unsee* that:

- silos are not human failure
- coordination is operationally necessary — but structurally insufficient
- and collaboration without architecture is mainly hard work.

That is uncomfortable.
And that is precisely why it is valuable.

USM does not ask for belief.
It asks for consistency.

Not the consistency of 'everything at once', but that of starting somewhere and not falling back.

Perhaps this book will not change anything in your organization.
That is possible.

But if it does change something, it will be this: that from now on, you will recognize when you are deploying people to compensate for a design flaw.

And that you then have a choice.

Not to work harder.
But to design better.

That is where this book ends.
And that is where the real work begins.

Organizations are not stuck because people do not work together.

They are stuck because they are designed to compensate for cohesion through extra coordination rather than managing.

In “From Silos to Ecosystem”, this book shows why silos (islands) are not a cultural problem, but a logical consequence of how we have structured management. Why managers have become primarily coordinators in practice. And why new frameworks, techniques, and tools tend to reinforce fragmentation rather than solve it.

This book does not advocate for better cooperation.
It advocates for better design *with which* you can achieve better collaboration.

Based on the Unified Service Management (USM) architecture, it becomes clear, step by step, how organizations can become manageable again:
* by defining services as chain performance
* by using a single shared management architecture
* and by separating management from coordination.

No new frameworks or techniques.
No transformation program.
No human improvement.

But a fundamental *paradigm shift*:
* from improvisation to design
* from heroism to reliability
* from repairing people to breaking down walls.

This book is written for managers, administrators, and professionals in complex organizations (government, healthcare, education, large service providers) who feel that things need to change — and are willing to *think* differently before they start *doing* things differently.

The Red Thread case study in this book is dedicated to municipal secretaries and their directors of operations.

Curious about how the USM method can help you set up an effective service strategy, inspired by practices from popular frameworks, but based on architecture and systems thinking?

=> Read the USM Wiki, the USM portal, the USM book

You don't solve chaos with more coordination. You solve it with structure.

Publication data

A publication of the SURVUZ Foundation — Governor of the USM method.

www.survuz.org

SURVUZ FOUNDATION
Governor of the USM Method