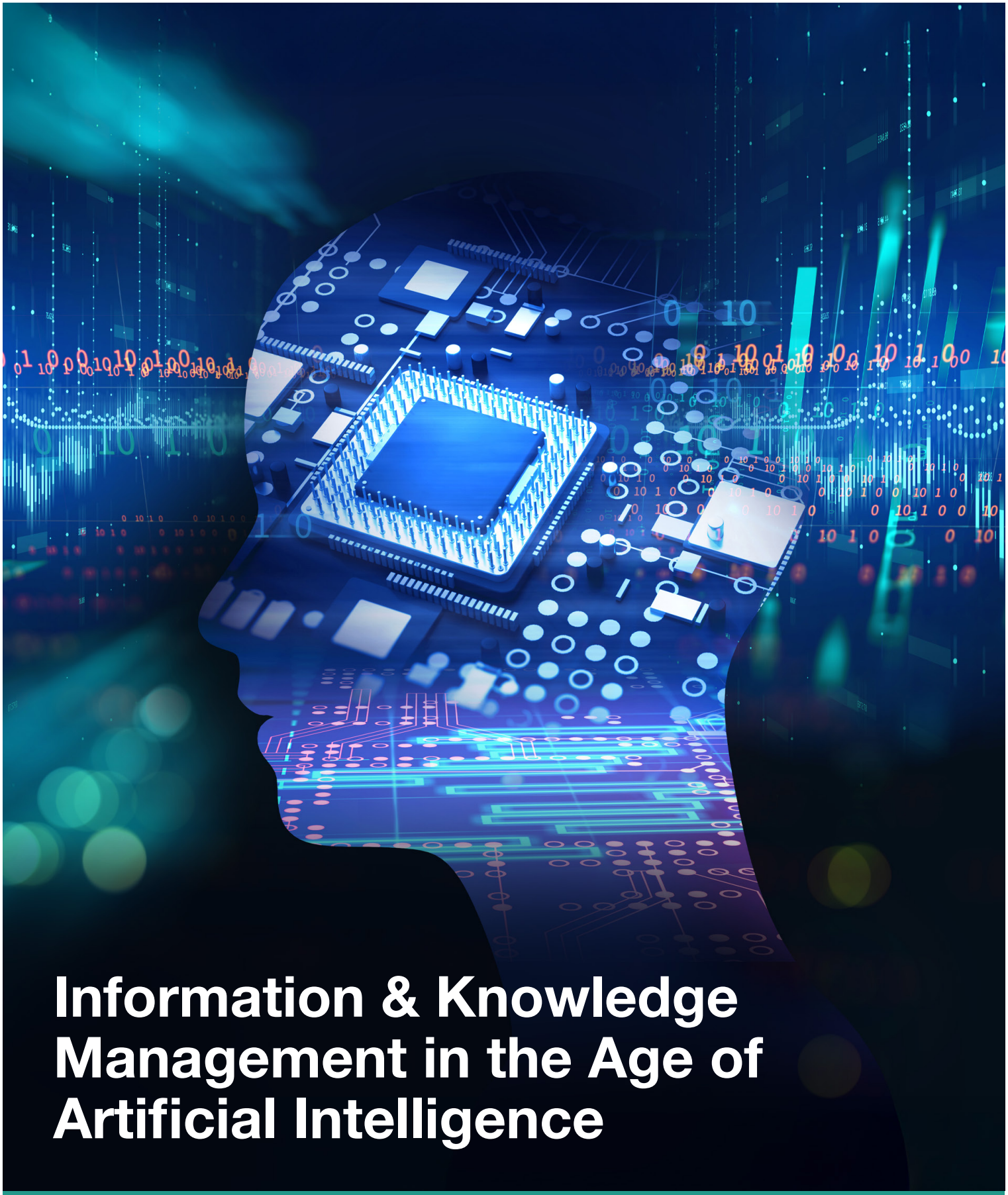


Information & Knowledge Management in the Age of Artificial Intelligence

Whitepaper by

Berty Hooyman, Chief Architect, Mphasis

Suresh Nair, Principal Consultant – Consumer Banking, Mphasis

Mphasis

The Next Applied

Contents

1. Introduction

Every major technological advancement—from fire and agriculture to the Internet and general-purpose AI—has been accompanied by deep cultural and societal changes. These changes are not always recognized when they happen.

In this paper, we focus on a very small subset of changes related to information and knowledge and some of the radical changes we think are being missed.

2. Observation: We are Creating Very Litle New Information – Just Directing the Movement of Information

The speed and ease of much of what we do today stem from the seamless integration and data exchange capabilities of our service providers. For instance, when applying for a loan, all we need to provide is a unique identifier and permission for credit agencies to share our data. Loan approval hinges on granting loan providers and servicing agencies authorization to share data with credit agencies, allowing the process to proceed smoothly.
A more detailed description is provided in the appendix [see: 7.1].

This seamless data exchange has become so fundamental to our lives that when faced with a form containing hundreds of fields, we often question whether it is worth the effort. In the retail industry, organizations maintain descriptions of goods as “Stock Keeping Units” (SKUs) in their retail management systems. In the past, armies of people in back offices manually entered this data. Today, standards like Universal Product Codes (UPC) enable the seamless exchange of information about retail goods across every layer of the supply and retail chain, eliminating the need for manual data entry.

To simplify logistics tracking, manufacturers include RFID tags in packaging or attach them to products.
Internet of Things (IoT) technology can read these tags and track the movement of individual units from the factory to the point of sale. With precise knowledge of unit locations, we can manage inventories more effectively, with systems automatically restocking items when they run low or reducing the inventory of products that are not selling.

2.1 Adapting to/Adopting the Change

Today, the traditional ecosystem between customer and supplier involves IT systems talking to each other, with the mirror of the process implemented on both sides.

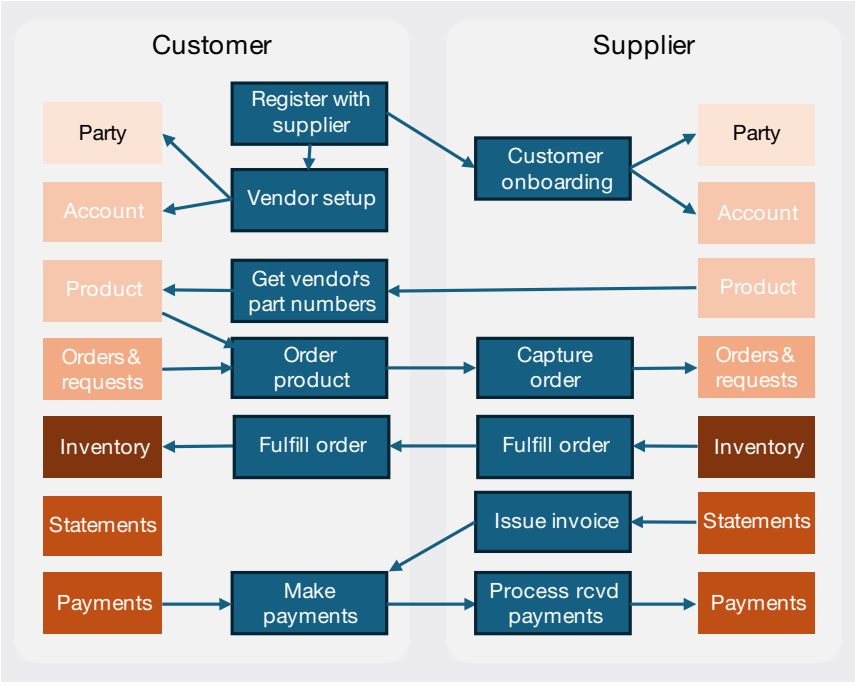


Figure 1: Simplified view of traditional customer/supplier ecosystem

In this model, data is replicated on both sides, with significant operations on both sides to reconcile “inputs” (e.g., orders, statements) against “outputs” (e.g., shipments, payments).

Today, we have a number of “platform” ecosystems that provide an alternative to this model, where shared data is radically simplifying this ecosystem. The following diagram shows what this looks like:

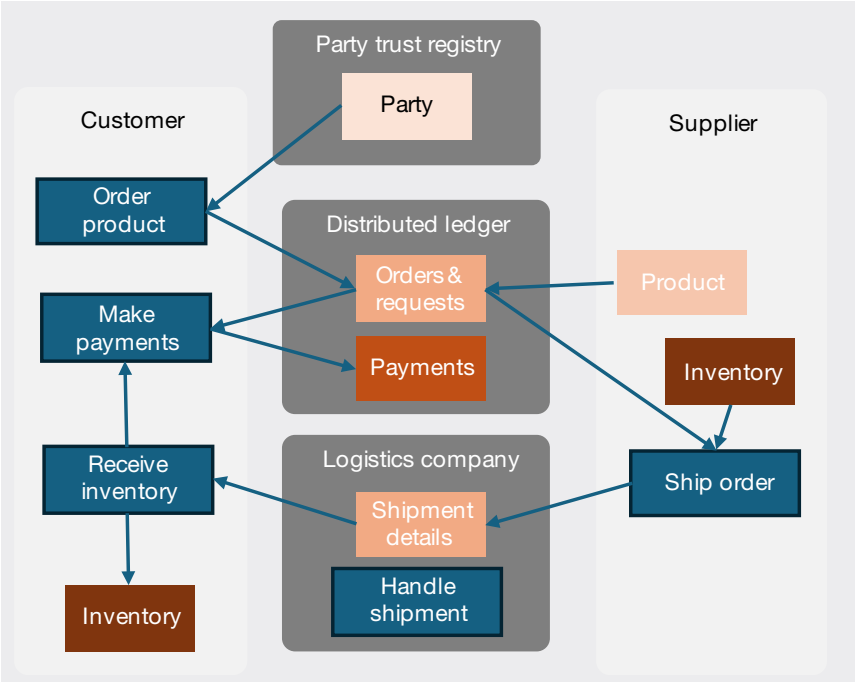


Figure 2: Simplified customer/supplier ecosystem with shared data

This significantly reduces the operations footprint, without loss of control by either party.

The most common versions of this are dependent on B2B platform companies like Alibaba, Amazon Business, eWorldTrade, etc. Web3 (see: https://en.wikipedia.org/wiki/Web3) is aimed at democratizing this and eliminating the need for platform companies to broker the relationships.

The appendix includes a discussion on how this would be achieved within the enterprise. See: 7.2 Shared data within the enterprise.

3. Observation: Why Does Each System Have its Own Language?

If we accept the premise of the previous observation and accept that “no IT application is an island,” it is surprising how little this concept has influenced the actual process of designing and maintaining IT systems. If all the information we handle already exists somewhere in the “ether,” why do we find ourselves modeling it anew each time we build a system?

Organizations that have embraced the digital age extensively use APIs to communicate with their partners, customers and peers. This is crucial for many modern innovations such as just-in-time supply chain management, hyper-personalization and “composable enterprises” like Amazon or Apple, as well as digital businesses like FinTechs. In nearly every industry, formal standards have evolved to make interactions predictable and scalable. The common language used by practitioners has been embedded into these APIs.

However, within organizations, we insist on creating our own “language.” Every organization insists on developing complex internal models with their own layers of governance. Imagine for a moment that we are comfortable using English to talk to our neighbors and friends, but insist that within our homes, we must speak our own made-up language because our family is “special” and does things that no other family does.

Instead of inventing a completely new language, what if we simply added a few words for things that are absolutely unique to us and our shared values, and for everything else, we adopt the common language? Nearly every IT project we work on feels like we are discovering a shared language from scratch in each project. What is truly perplexing about this situation is that the business users in the organization are already comfortable using industry-standard terminology for what they do. The made-up language is one created by the IT teams to capture their understanding of the business language.

3.1 Recommendation: Force IT and IT Systems to Adopt "Public" Languages

Most industries have published standards for APIs that have wide adoption. These standards are managed by industry governance bodies, go through detailed peer reviews and are tested through real-world usage in APIs for interchange.

A few examples:

Standard	Domain	Highlights
ISO 20022	Banking and finance	Addressed mismatches between different standards that had evolved independently. Provides a master list of terms and data structures covering all banking APIs and hand-off standards. Also adopted by BIAN as the data model backbone.
ACORD	Insurance	Covers business processes, product models and data exchange formats that enable standardization and automation across insurers and intermediaries. Used extensively for exchanging insurance policy data, licensing, commissions and transactions.
SIRI, TCIP, GTFS	Public transportation	Exchange real-time transit schedules, vehicle locations and passenger information. Supports communication between transit systems, agencies and the public, enabling interoperability and real-time data sharing.
GS1 Standards	Supply chain	GS1 provides global standards for uniquely identifying products, locations and logistics units using identifiers like GTIN (Global Trade Item Number), GLN (Global Location Number) and SSCC (Serial Shipping Container Code). These standards underpin barcodes and RFID tags, enabling consistent tracking and data sharing across supply chains worldwide.
ICD-10-CM, CPT, HCPCS, LOINC, SNOMED CT	Healthcare	**ICD-10-CM:** An illness and diagnosis classification system developed by the World Health Organization. **CPT:** The American Medical Association (AMA) developed CPT (Current Procedure Terminology) for medical operations and services, which is primarily used in billing. **HCPCS:** Healthcare Common Procedure Coding System is a variant of CPT and is more comprehensive. **LOINC:** For laboratory testing and clinical observations, LOINC (Logical Observation Identifiers Names and Codes) is used. **SNOMED CT:** Systematized Nomenclature of Medicine Clinical Terms codes include symptoms, procedures, diagnoses, family history and more.

One pre-requisite for this future model is to force the adoption of industry standard terminology deep into all internal systems. It means that "secret" languages used between internal systems will be replaced with common languages.

There will be some initial pain as "shorthand" terms are replaced with industry-standard terms. People who learn languages quickly will tell you that you truly learn a language when you use the language in your own head when you think. In the same way, this only works once the systems use the industry standard terms within the systems, not just at the boundaries.

Let's take a simple example from the banking industry.

Account: A legal contract between two legal entities where one legal entity (provider) commits to provide a set of services to another (consumer).

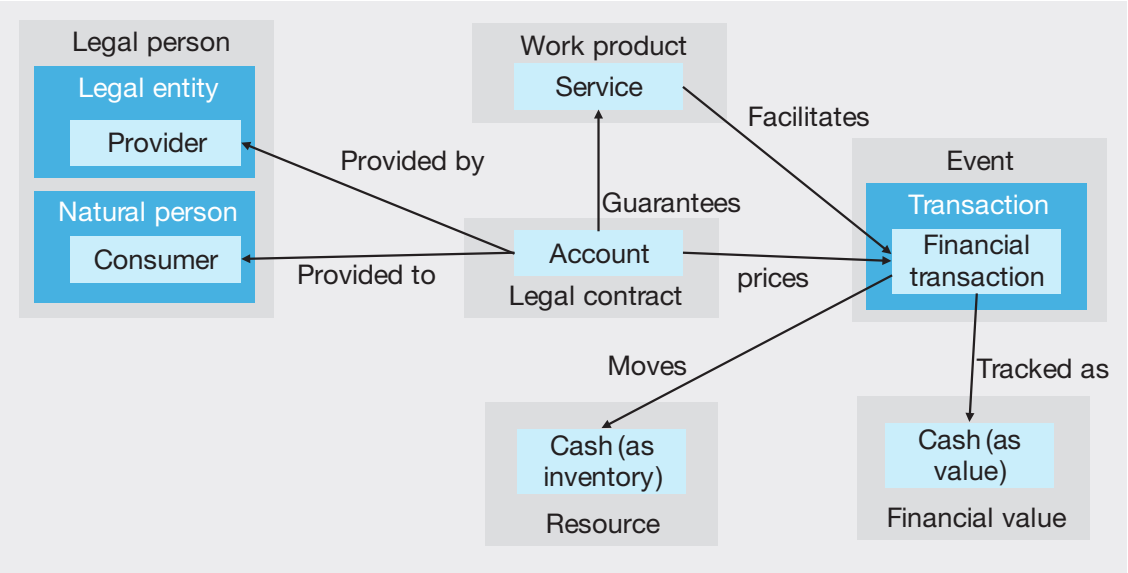


Figure 3: Simplified semantic model of account and transaction

If we follow the formal semantic model, we end up with a very different model of the world than we would see in traditional banking systems.

- There is no such thing as a "customer", just parties involved in contracts
- "Account" tracks the contractual terms between the customer and the bank, with pricing for the transactions facilitated by the services
- Transactions are events that move money between parties, similar to how goods are moved between companies in traditional trading systems
- The "value" of the cash is tracked separately, just as the value of inventory is tracked in traditional systems

Building a system like this eliminates a huge percentage of the problems that banks face today with their IT systems. By focusing on "data structures" and not "data meaning", we have introduced problems that require layers upon layers of additional processing. By reducing systems to this construct, we radically simplify the ecosystem.

4. We have Broken the Data/Information/ Knowledge Pipeline

4.1 The Traditional DIKW Pyramid

Most organizational ecosystems are built on the assumption of the existence – implicitly or explicitly – of a DIKW pyramid (see https://en.wikipedia.org/wiki/DIKW_pyramid).

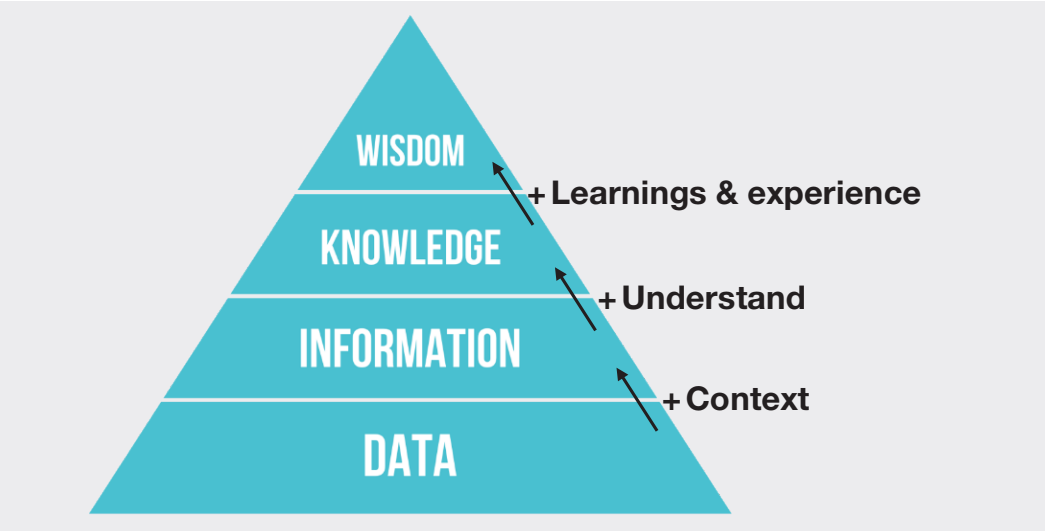


Figure 4: The DIKW pyramid

Data	Raw Values	42	2001-04-12	John Smith
Information	Adding context & definition to values	Number of people in a meeting	Date of birth	Customer name
Knowledge	Understand implications, inferences & deductions	Attendees required to approve project plan	Age is 23 years in Jan 2025 Is legally allowed to vote, drive or be served alcohol	Formal name captured from a government-issued ID through a controlled process
Wisdom	Enriched through experience. Not explicit in data, but arrived at through observation.	We need a voting system to save time. We must have a task tracker to control the process.	90% of wage earners in this age range have limited disposable income, but can be convinced to save 20% of income	Is a very common name in England and some parts of the US and Canada. Additional information is required to uniquely identify.

The DIKW pyramid is the foundation for information management systems that are used extensively today.

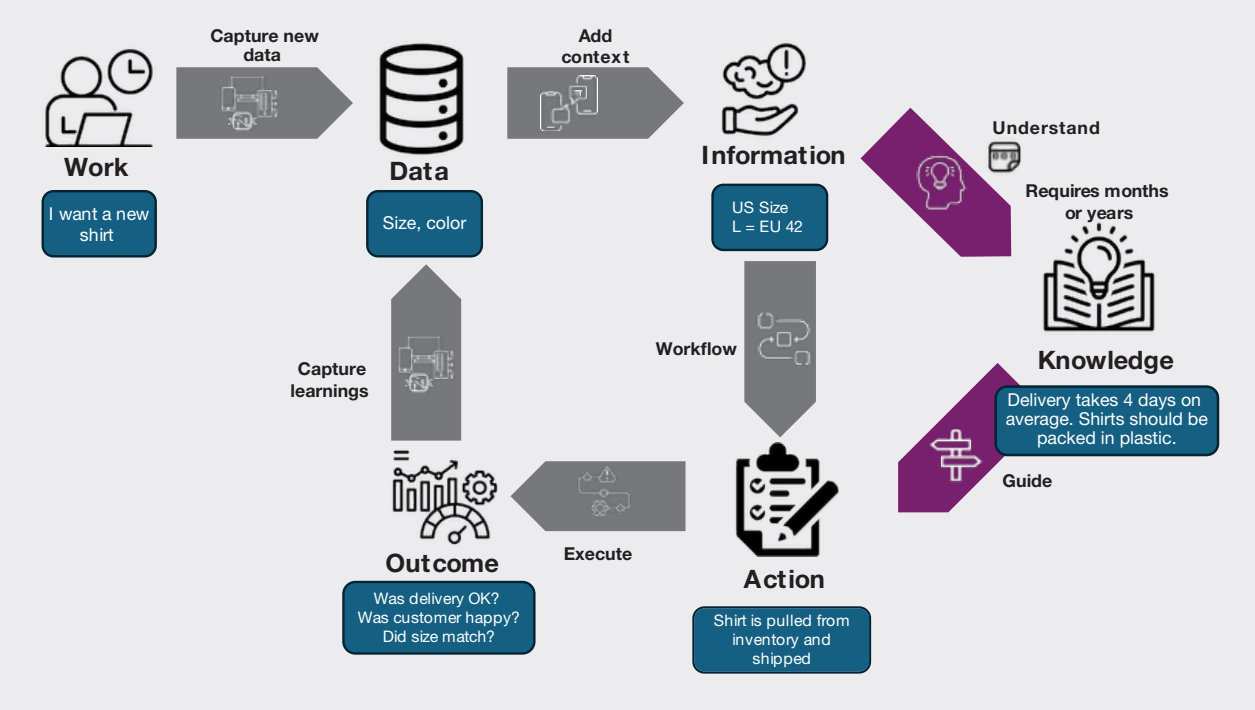


Figure 5: Traditional DIKW-based ecosystem

Data enters the ecosystem through work and is stored in IT systems with context as information. The generation of knowledge from information requires large quantities of data and analysis by experts. Knowledge is required to be justified.

This knowledge drives actions or “best practices”. The outcomes through the following of best practices become learnings, which is data in its own right.

4.2 The Disrupted Pyramid

Over the last 15 years, a new breed of companies appeared that have used technology to deconstruct traditional businesses in a way that enables faster innovation. They have deconstructed the organization into technology powered business capabilities integrated through APIs.

The cost of rapid transformation has been decreasing fairly consistently over the years. In extreme cases, organizations are often driven by intuition – often called the “move fast and break things” culture. In this subset of organizations, success is narrowly defined based on delivered stakeholder value.

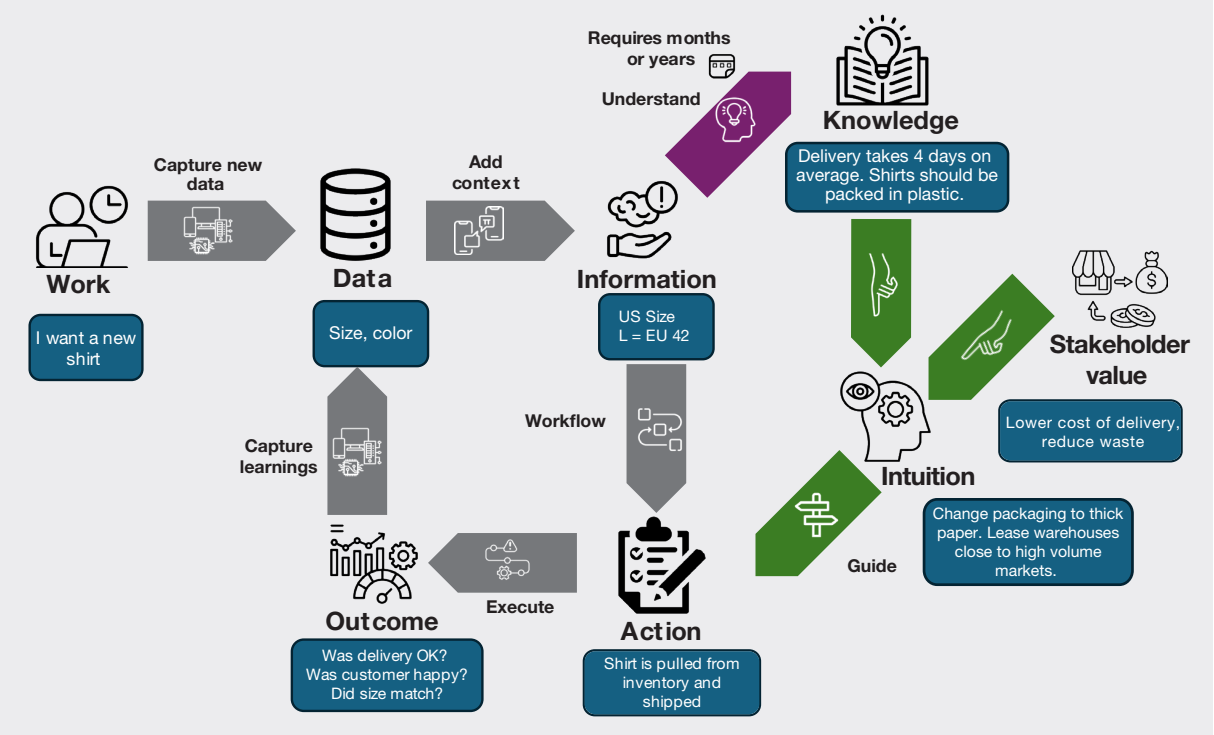


Figure 6: The "move fast and break things" data ecosystem

This has delivered transformational organizations like Google, Amazon, Microsoft, etc. Most large enterprises are actively retooling themselves to adopt this culture; whether explicitly or implicitly.

One of the fundamental tenets of the approach is that past knowledge is not trustworthy. Teams are encouraged to stretch the boundaries of past wisdom and see what happens. This has directly led to a much faster rate of change than any time in human history. Society itself becomes "beta testers" for new technologies. Rather than predicting and preparing for the potential impacts of change, we try out the change and react rapidly to negative impacts as they happen.

Part of what supports this is the huge foundational knowledge base in areas like the sciences, mathematics and economics. Given these rock-solid truths, we have much less fear of the unknown when trying out new things. We know the physical limits of anything we try to do. We have accurate economic models to assess the real value of any real asset and to judge how intangible assets can be valued.

5. Observation: Current IT Practices Actively Contribute to the Loss of Knowledge

IT systems automate what used to be manual tasks. Experts create requirements that tell developers what the systems need to do. Developers build the system to meet these requirements. Developers do not become experts in the domain as they build these systems. As IT ecosystems have grown, and more work has moved to large outsourcing firms, additional layers have appeared that increase the separation between subject matter experts and developers. Developers just think of system as moving data around, without understanding what the data means or does.

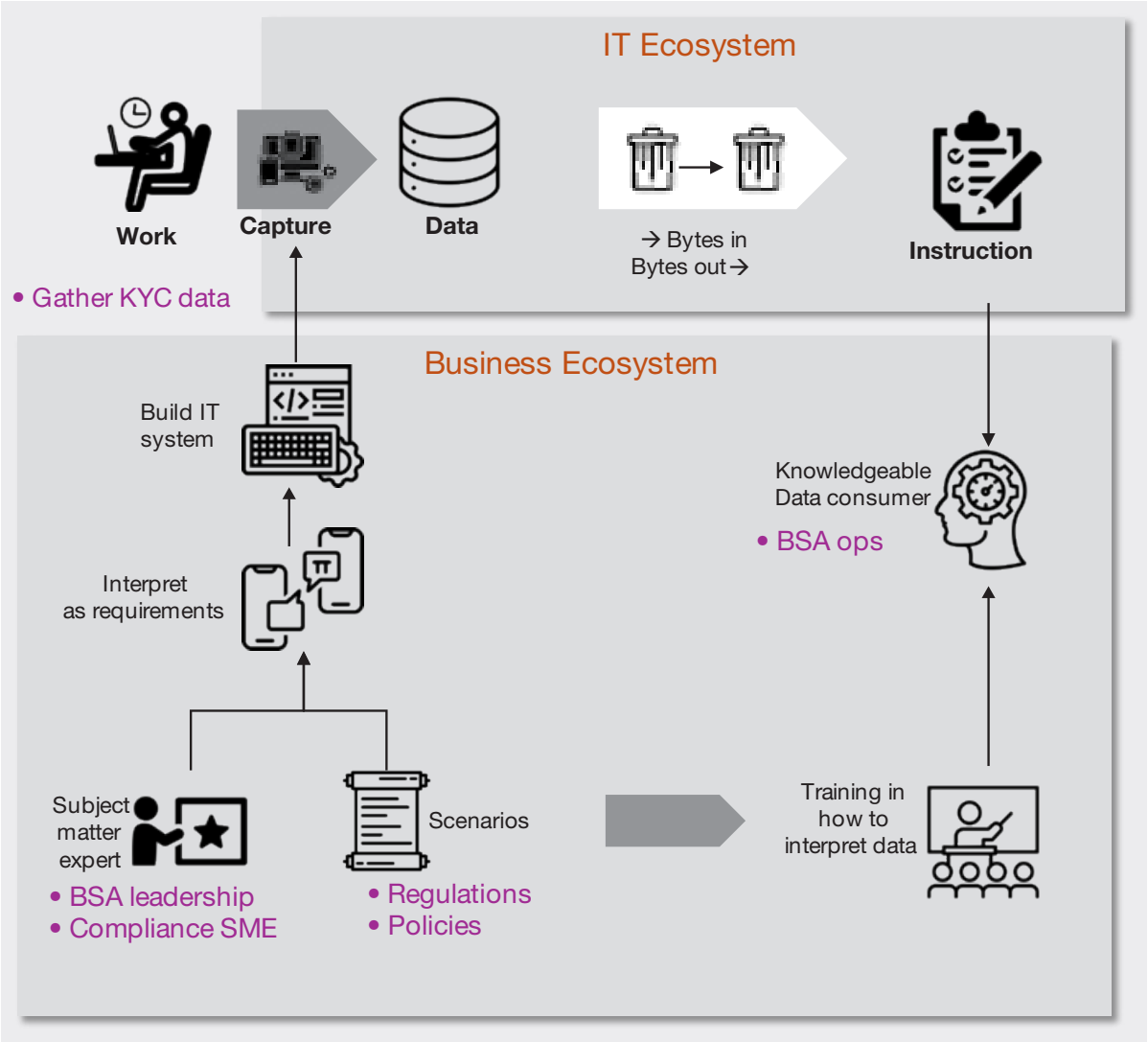


Figure 7: A typical application development ecosystem

5.1 Impact 1: Users Become Dumber

Before IT systems were in place, each participant in the ecosystem was expected to have the knowledge required to convert data to information. Imagine trying to file your taxes on your own before the advent of modern tax systems. As a sample, old US instructions for filing taxes can be found here: https://www.irs.gov/pub/irs-prior/i1040--1969.pdf

Rules for IRA computation of tax— If line 15a is under $5,000 and consisted only of wages subject to withholding and not more than $200 of dividends, interest and nonwithheld wages, and you are not claiming any adjustments on line 15b, you can have IRS figure your tax by omitting lines 16, 17, 18, 20, 21, 22, 23, 24, 25 and 26 (but complete line 19). If you are filing a joint return, show husband's income and wife's income separately in the space to the right of line 15c. Identify husband's income by marking (H) and wife's income by marking (W).

Note: If the IRS figures your tax and surcharge, the law does not permit the IRS to allow you the benefits of: (1) the retirement income credit, (2) head of household or surviving spouse status, and (3) minimum standard deduction, if you are married and filing a separate return. If you are entitled to any of these benefits, it is to your advantage to figure your own tax and surcharge.

Figure 8: Rules for IRS computation of tax (source: From 1040 Instructions, 1969)

Today, most people using automation systems to file their taxes have not read the code. The tax code has grown incredibly complex. The same IRS guideline today runs to 100s of pages, with a wide variety of special cases described and handled.

However, the vast majority of people filing their taxes do not understand the tax code, and most have never read it. They rely on the system to ask them questions that they can understand.

The democratizing of tax filing has also resulted in a "dumbing" of the average person about taxes.

5.2 Impact 2: IT Teams Bleed Business Knowledge

A bigger problem is that the IT systems are not designed to manage the knowledge that was captured from domain SMEs, and is definitely NOT captured as knowledge in the IT systems.

Business rules are converted into standard code constructs like IF statements, FOR loops or more complex algorithms. Code is designed to drive consistent processing of data, not to retain knowledge.

As systems evolve over time, the knowledge that the Subject Matter Experts (SMEs) put in at the start becomes dated, and eventually outdated and irrelevant. Every change made to the system reduces the value of the original documentation. In many cases, the documentation was not preserved, making matters worse.

Besides business knowledge, IT systems also require a lot of technology knowledge. Systems work on data in databases, present data in user interfaces, exchange data with other systems through files and APIs, etc.

The modern enterprise system typically involves hundreds of different technologies, even if they use a single language like Java, C# (.Net) or Python. Enterprises provide developers with frameworks that take care of making these technologies easy for developers to use. However, they fall into the same trap, resulting in "dumber users" – i.e., developers do not understand the technologies they are manipulating through the frameworks and "loss of knowledge", as the developers of the frameworks move on to other projects.

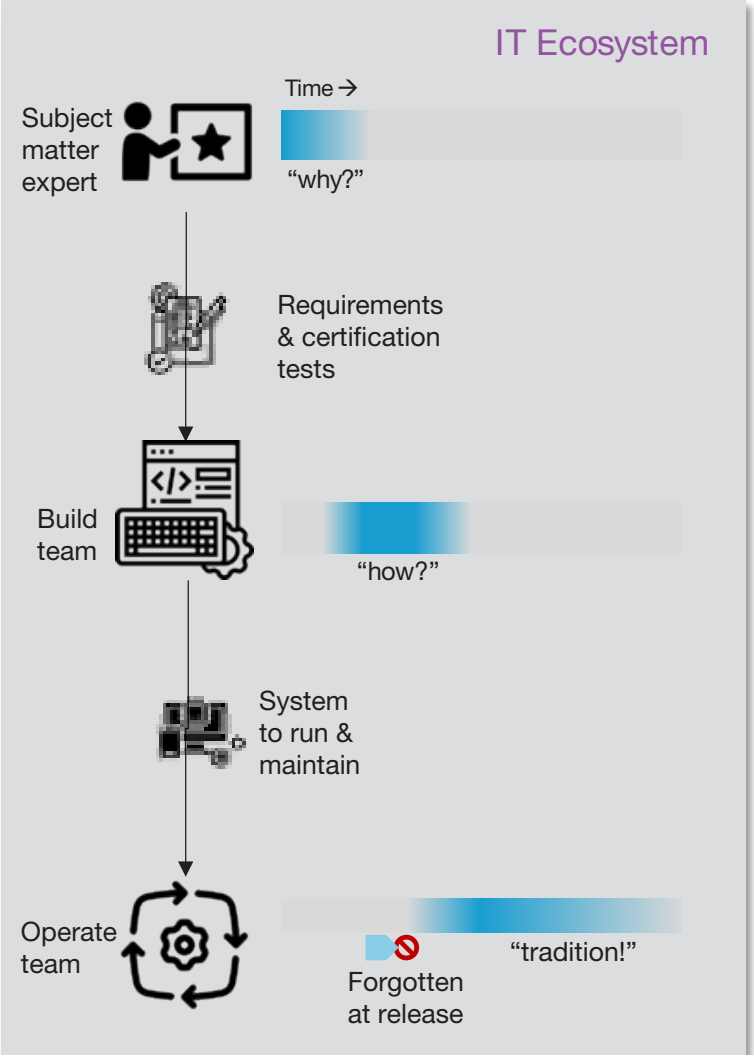


Figure 9: Most IT ecosystems lose knowledge

5.3 How do We Fix This?

We need to embed knowledge into the IT systems that we are developing. This means creating dedicated knowledge repositories and forcing them to become part of the development pipeline. The following shows a model that we have put in place for a number of projects.

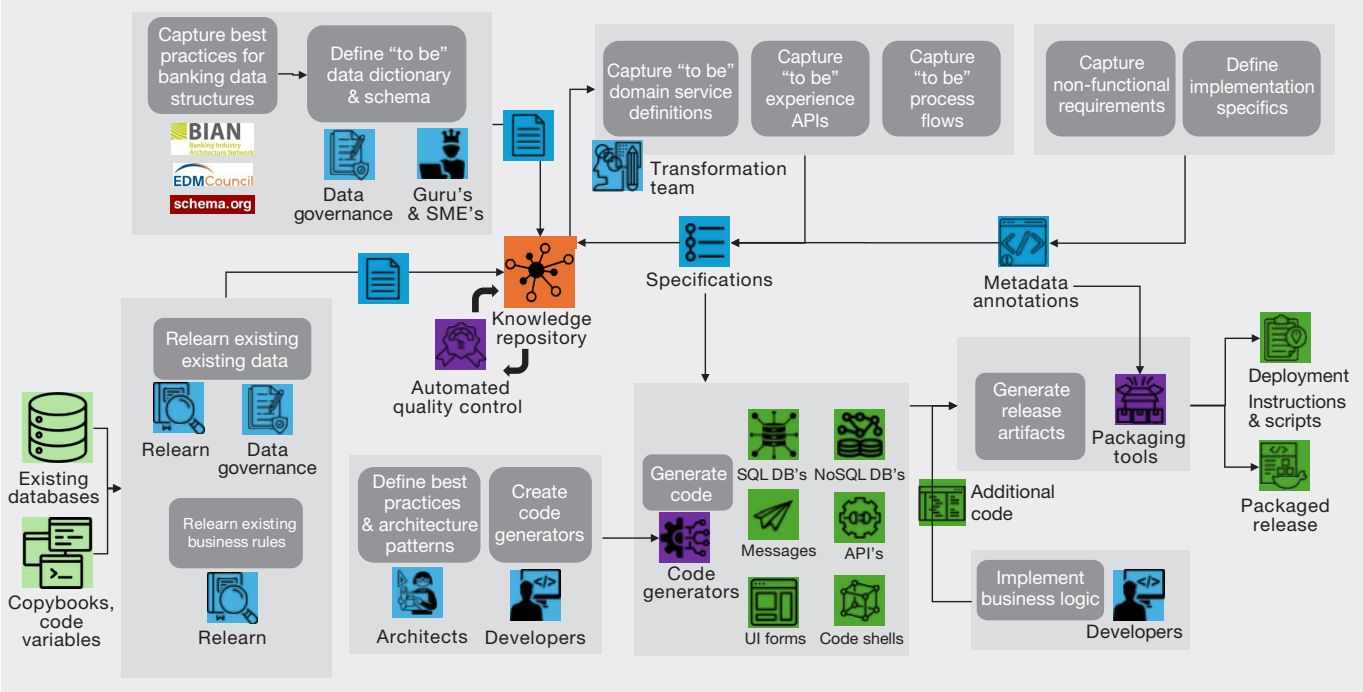


Figure 10: Making a knowledge repository the core for IT system generation

We seed the knowledge repository using standard industry models. A huge percentage of functionality in systems can be fully described by knowledge that is in the public domain.

Another source of knowledge is legacy source code. AI tooling is used to relearn business rules and data structures from legacy systems.

The functionality and behavior of the system; and non-functional requirements (volumes, SLA guarantees, usability mandates) are also captured and encoded.

Coding automation and modern prompt engineering tools can be used to generate a large percentage of code. Only a small percentage needs to be developed by hand.

In this ecosystem, since knowledge drives the code, any change must start from updating the knowledge model. The knowledge model is fed into AI agents that then generate the new code. This ensures that & mandates the knowledge model always remains current.

6. Observation: The Pool of Traditional Experts is Shrinking

Along with the erosion of the traditional data/information/knowledge/wisdom pyramid, we are also seeing the pool of what we would traditionally call experts dwindling.

A generally accepted definition of an expert is someone who has internalized years of accumulated human knowledge and has synthesized their own perspective. They do not just repeat past learning, but are able to synthesize new solutions from their knowledge,; which in turn increases their knowledge – becoming wisdom.

There are a number of well-known reasons for the reduction in the expert pool:

Problem 1: Knowledge barrier is too high, and growing
As systems have become more complex, individuals need to amass large amounts of knowledge to become even basic experts in a field.

Think of auto mechanics today; who are required to have deep engineering expertise to handle complex engines; software engineers to handle the complex control systems, electricians to handle the complex wiring handling all the sensors and automation; and chemists to handle the unique bonding agents and adhesives used in the different kinds of seals and joints.

Problem 2: Partially outsourced brains
Another major shift has come from the way that people learn and use knowledge. Ubiquitous access to smart phones and the Internet has changed the way in which people find and use knowledge. Rather than remembering things, people have "outsourced" part of their memory to the internet. They have the confidence that they can find very topical knowledge when they want it, exactly at the time they need it,; at next to no cost.

Today, think of how many people repair things after watching YouTube videos. It is really scary when we realize that the same behavior we use to fix a leaking toilet is being used to building transaction processing systems.

Problem 3: End of "Seek, and you shall understand"
The act of searching for the right knowledge for a task helps the consumer understand the knowledge better.
One learns to ask the right questions, and through questioning, get better at understanding the results of their search.

AI agents automate the search and aggregation of knowledge based on its best guess on what we are looking for.
AI agents use their knowledge of the context, skill level of the user and knowledge of what similar searches are happening around the world to fill in the gaps in the questions being asked.

The user becomes the embodiment of the sentiment - I do not know what I do not know.

It may be that knowledge has truly become democratic and available to everyone. The traditional expert may no longer have a true role. It may take a generation or more for the long-term impact of this loss of traditional experts to be fully understood.

6.1 Observation: TThe New Knowledge Worker: Conductor of an Orchestra of Agents

What is evolving instead is a new type of expert, one who is solely an expert in stitching tools together. They live in a symbiotic relationship with AI agents. They have a clear understanding of what they want to achieve and leverage combinations of AI agents to achieve it. This is often worded as "AI Will Not Replace Humans, AI Will Replace Humans Who Don't Use It" (see: https://www.outlookbusiness.com/corporate/ai-will-not-replace-humans-ai-will-replace-humans-who-dont-use-it-says-mphasis-ceo-nitin-rakesh).

Just as we see people outsourcing part of their intellect to smartphones, we predict the new experts will use AI as force multipliers, allowing them to become experts in real-time, in dozens of distinct domains, working together in real time.

Imagine Augmented Reality devices such as the Microsoft HoloLens or Apple VisionPro tapping into the huge platform ecosystems these organizations have access to, along with dozens of intelligent agents that perceive all aspects of the user's world through the device's capabilities. Sight, sound and touch are encoded in real-time. The digital world streams in and out through multiple APIs.

If the user has an army of agents working with them in real-time and ingesting all this input in real-time, and helping the user absorb every aspect of the inputs, and responding digitally to the user's will, the results may appear almost as magic to fellow humans who are not able to adapt to this kind of new world and ways of working.

7. Recommendation

7.1 A New IT Ecosystem Paradigm

The most important recommendation we are making is a completely new paradigm for IT systems. The paradigm is summarized in this diagram.

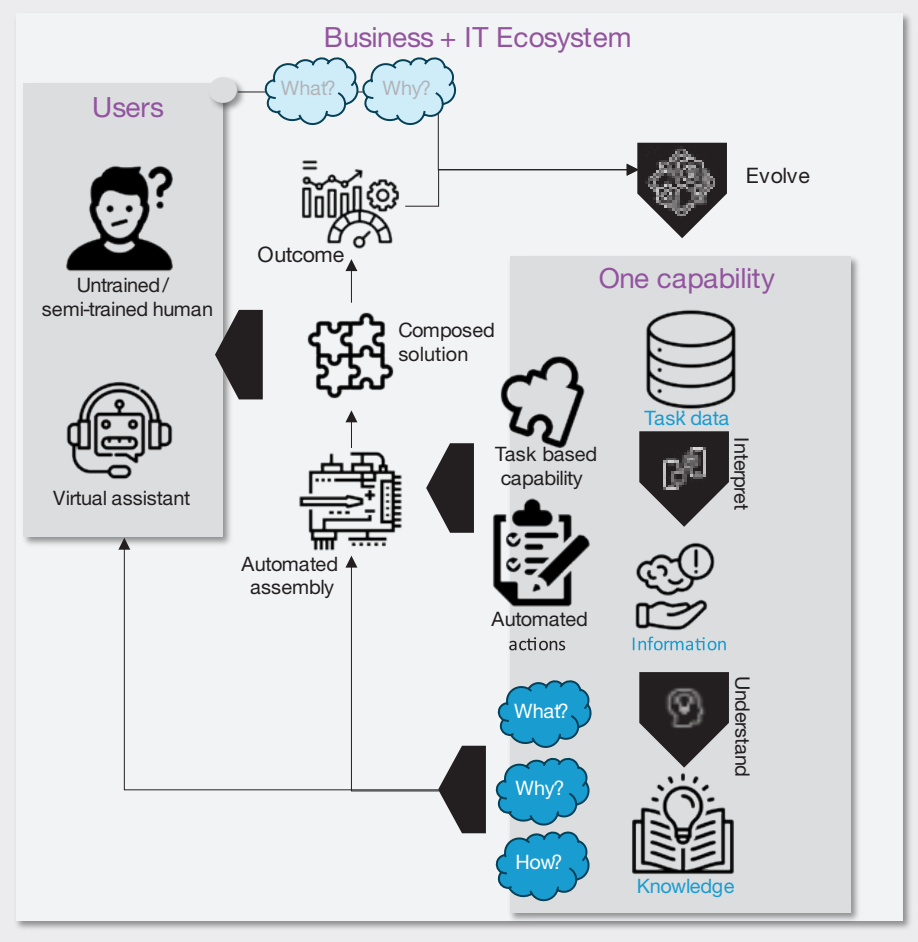


Figure 11: The new knowledge worker ecosystem

User: Untrained human	The changes mentioned earlier with users losing domain knowledge are irreversible. We also want to ensure that everyone benefits from automation. To this end, systems must be able to cater to any kind of user, whether they know the domain or not. It needs to cater to gaps in knowledge and protect the user from their own lack of knowledge.
User: Virtual assistant	Increasingly, humans will use virtual assistants to get work done. Rather than interacting with systems themselves, humans will interact with virtual assistants, which in turn will interpret the ask, and instruct the applications to get work done.
Composed solution / automated assembly	A critical change required is to get past the idea of “fixed” collections of applications, and allow assemblies of capabilities to be created in real-time to handle a task. The two examples in the next section show this at work.
One capability	To allow for the dynamic assembly of solutions, instead of applications, with their own user interfaces, we need to think in terms of individual capabilities that can be assembled in different ways. Capabilities perform single tasks and own their own data.
Capability API	Functionality and data are private to the capability and can be accessed only through APIs provided to the outside world.
Capability level knowledge	Knowledge needs to be encapsulated within capabilities and should be accessible just like APIs. However, knowledge cannot be updated via APIs, only by the capability owners. To understand how the capability is used (i.e., to capture context), the capability owners will require additional information in the API layer to capture context.
Outcome tracking and evolution	To provide for continuous improvement, the system needs to understand whether the outcomes required by the users have been met. This will help continuously evolve both the assembly of solutions and the capabilities themselves.

The examples below show these in use.

8. Example 1: The Everything Calendar & Time Management System

Let's imagine I want to have a tool that provides me with 100% of everything needed to manage my time, ensuring that I get all my chores done, never miss a meeting, a flight or a child's music recital.

This is much more than a calendar – it is a living system that helps track every single thing I need to do.

Let's take 2 sample scenarios:

Scenario 1: Grocery side trip

- I get a call to pick up milk on the way home, and that needs to be added to the calendar, including travel time to the store, time for shopping and time to get home.
- An optimizer should kick in, and add anything else that I need to pick up from the store – even things that I might not need immediately.
- Since my electric car was not at full charge in the morning, the optimizer should pick up that I need to charge, and pick a supermarket with charging available outside the store.

Scenario 2: Annual budget forecast

- I have never had to submit a budget before, but after a recent promotion to unit head, I am required to prepare the budget for my group for the coming year.
- I am told today (Tuesday) to have the budget ready for review by the department head by Friday.
- My agent figures out the tasks involved in creating a budget for a unit like mine, adds the tasks to my calendar with dependencies, and estimated times.
- The agent flags dependencies on things I need from my colleagues, and adds time to request it from them + for me to review and incorporate it into my budget.
- The system checks calendars for my colleagues to create invites aligned to my calendar. It finds that one of the key people in the department is off till Thursday. The calendar immediately puts the budget deliverable at risk, and suggests work arounds. This involves my spending additional time to gather the inputs myself.
- Lower priority tasks in the period are postponed to after the budget submission date to make space for the new tasks.

Here's one possible ecosystem that would support this, following the approach suggested above.

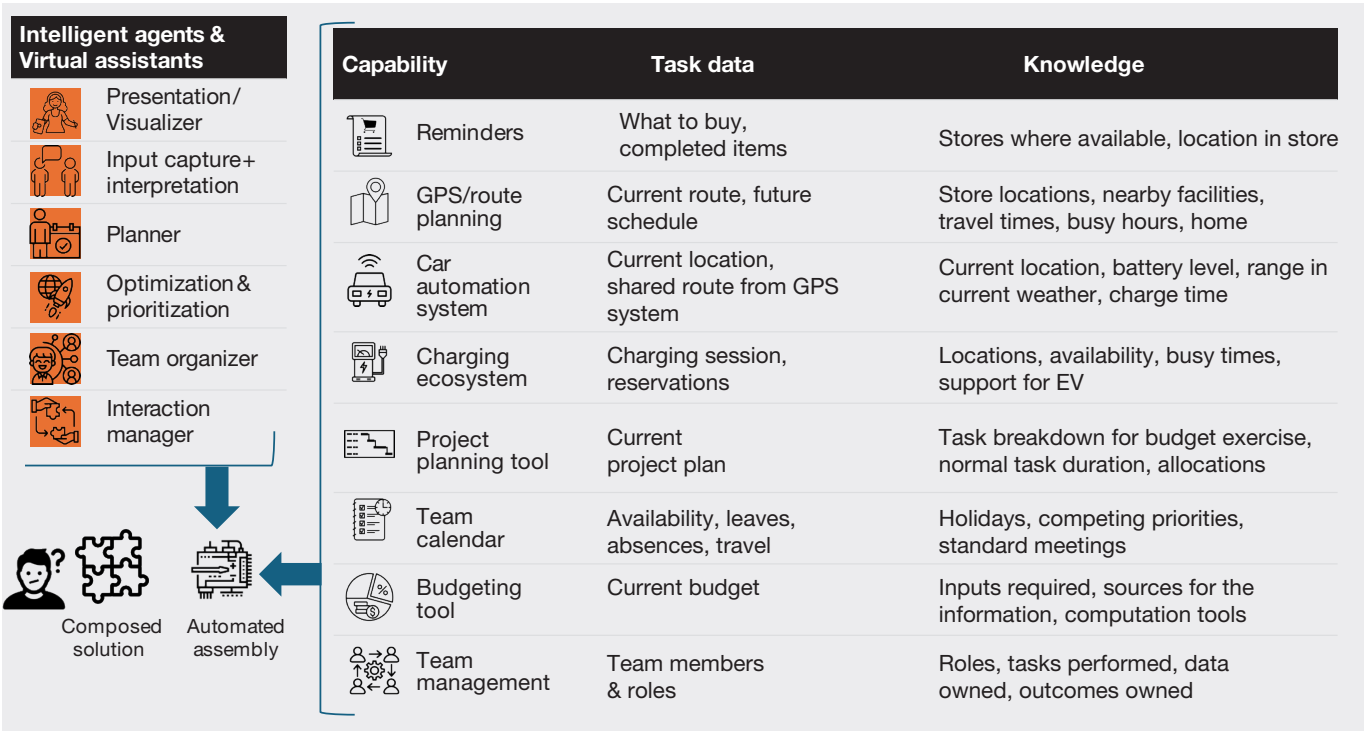
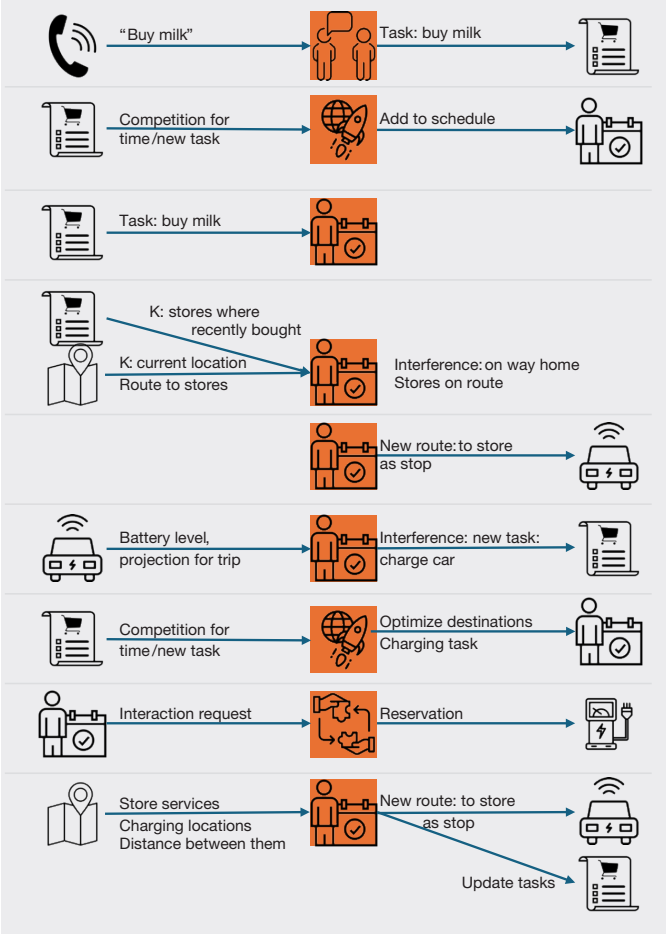
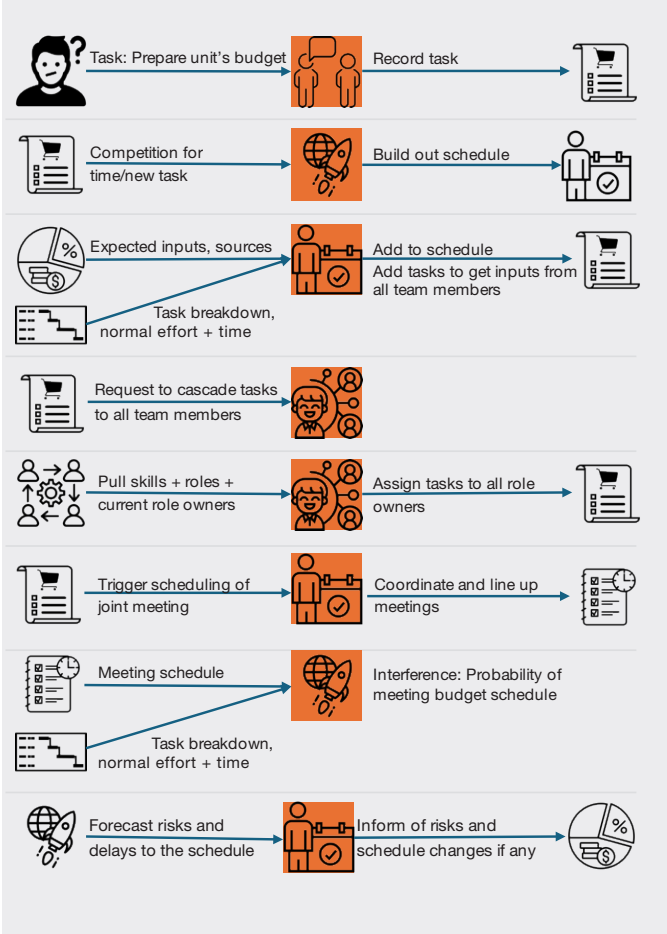


Figure 12: Agents and capabilities in the calendar ecosystem

The following is an extremely simplified view of how these agents and capabilities would interact to achieve the required outcome.



Scenario 1 (Calendar): Interactions between agents & capabilities



Scenario 2 (Budget): Interactions between agents & capabilities

8.1 How and Why this Addresses Observed Issues

This approach recognizes and addresses many of the problems observed in traditional systems:

- **We are not creating new information**: The approach inherently recognizes that we are moving data around. It avoids duplication or re-capture of data. Each system owns its own data and knowledge. We rely on the transient "memory" of the agents that are stitching the tasks together to pull the data together the instant we need it, then forget it once it is done. What needs to be remembered is remembered in each system that owns the related task data.
- **The DIKW pyramid is gone**: Each domain captures knowledge relevant to one area, irrespective of usage scenarios. This allows for much deeper capture of specialized knowledge. The agents that assemble capabilities together rely on this knowledge rather than attempting to create deep knowledge models internally.
- **Data is fragmented, in good way**: Rather than thinking of data fragmentation as a problem, we are creating an ecosystem that assembles fragments of information when we need them. The intelligent agents create assemblies of data where and when they need it.
- **Breaking out of the GIGO paradigm**: The ecosystem is deconstructed into smaller units each of which understands "what" it does. When assembling capabilities together, the system clearly understands exactly what it is trying to accomplish.
- **Business knowledge is retained**: Since capabilities are built around "atomic" business capabilities, the APIs expose data in business terminologies and knowledge retention is built into the capabilities, business knowledge is retained explicitly.

- **Traditional experts replaced with knowledge sources**: The agents pulls data from multiple sources, many that the user would not of thought of at all. Rather than expecting the user to "ask" for information,the system is asking "is your knowledge useful for this task".
- **Support for the new knowledge worker**: The example shows how the user has a clear force multiplier experience, where many factors are considered and balanced seamlessly. The individual agents are working towards helping the user achieving their goals, without making them think through each integration and task.

9. Example 2: Consumer Bank Back Office

It can be argued that the first example is already available to some extent in the Google/Android ecosystem and is evolving in the Apple/Siri ecosystem. In the next example, let us consider the same kind of thinking within the enterprise in the back office of a consumer bank.

In this example, we want to focus on the everyday stuff that happens in the background of daily operations of a bank.

For the purpose of this conversation, let's use this simplified model of a consumer bank:

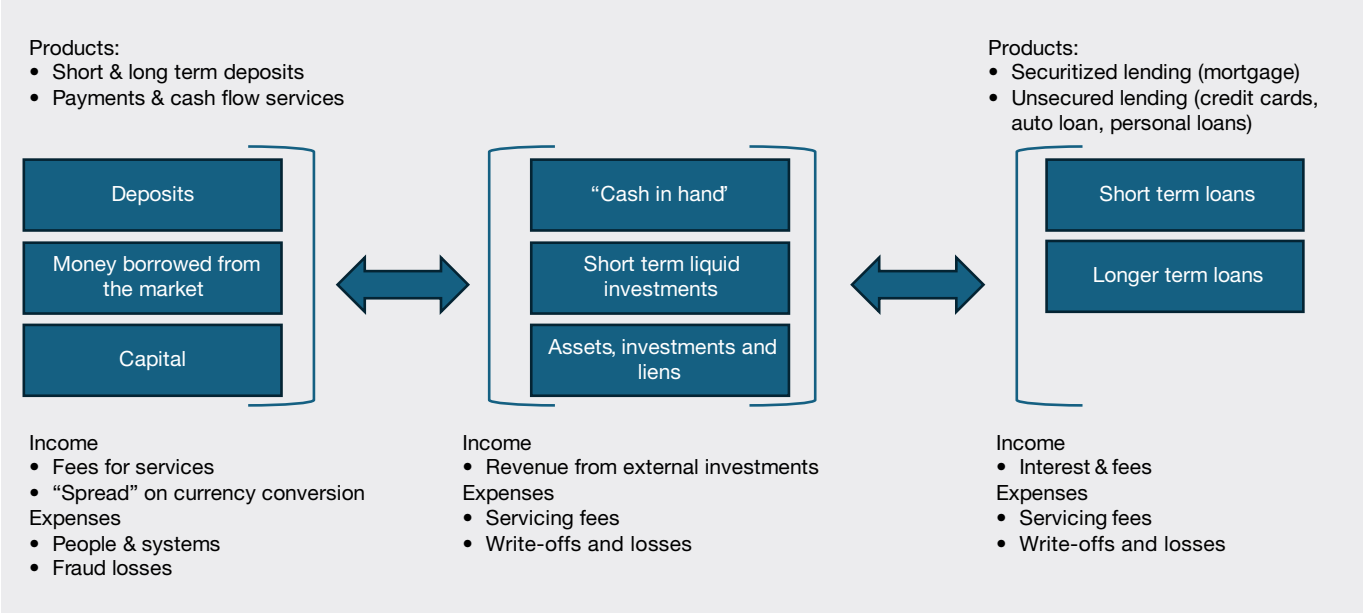


Figure 13: Simplified supply chain model of a bank

9.1 Limited, but Complex Scope for Discussion

For the purpose of this discussion, let's consider a few critical business tasks of the back office (IMPORTANT: this is not everything that happens, but focuses on a few of the most critical items).

Traditionally, banks have been built around the key books and records that the bank maintains.

Customers are offered financial services such as checking accounts to manage cash and payments; credit cards for short term credit lending, or payment ecosystems that help customers make or receive payments.

- **Back-office tasks**: Handling secondary tasks resulting from any front office/self-service operation that could not be completed completely automatically. This does not imply that it requires human intervention – just that additional steps are required that could not be completed in real-time.
  - E.g., A customer applies for a mortgage online. We need to decide if a home inspection is required, and if so, schedule it.

- **Record financial impact of events**: Capture the impact on the bank's own ledger, due from/due to partner banks & the impact on the customer's account balances.
  - E.g., When a new credit card is issued, record the potential exposure/risk that the bank is taking on through the issuance of a credit line. In the card account, set the "available credit line" to the approved limit.
- **Monitor and manage risk**: Use traditional risk factors such as total income vs. fixed expenses, past behavior and recent life events to forecast the customer's potential future behavior. Keep track of the total risk in the bank's portfolio, allowing the bank to take on a few higher-risk customers if they can be offset by safer customers, or where the potential upside is significant.
  - E.g., Look at the lifetime potential of college students when offering credit.
- **Discretionary pricing**: Where the product supports custom pricing for clients based on a comprehensive view of relationship.
  - E.g., When establishing pricing for a small business owner, factor in revenue from the small business;, providing preferential pricing to the customer.

9.1.1 Green Field, Not "Brown Field"/Current, Not Future

The consumer banking industry has been one of the earliest and widest adopters of computerized automation, and the thinking in banks is driven by 60+ years of real-world experience. Let us use our new paradigm, throw everything out and start again.

We are setting a few boundaries for this conversation:

- For the purpose of this discussion, we are not going to worry about backward compatibility or phased conversion. It is not as hard to do this as it might appear on the surface, but is beyond the scope of this document.
- We are intentionally limiting the approach to technologies that are considered mainstream and stable today (March 2025).

9.2 Agents and Capabilities in the Banking Ecosystem

To set up this ecosystem, we are NOT going to create a set of interconnected applications that serve different functions. Instead, we are going to tell our agent empowered people what they need to achieve. They will work on the tasks, trusting their intelligent agents to assemble themselves dynamically to complete them.

This is going to take a huge leap in thinking. It is ingrained in us that we need to build pre-integrated, stable ecosystems for anything involving financial transactions. The worst thing that can happen with the calendar app is that you miss a deadline. If a transaction is not recorded, or if we approve a transaction that is not allowed as per law, our bank will face significant regulatory and legal challenges.

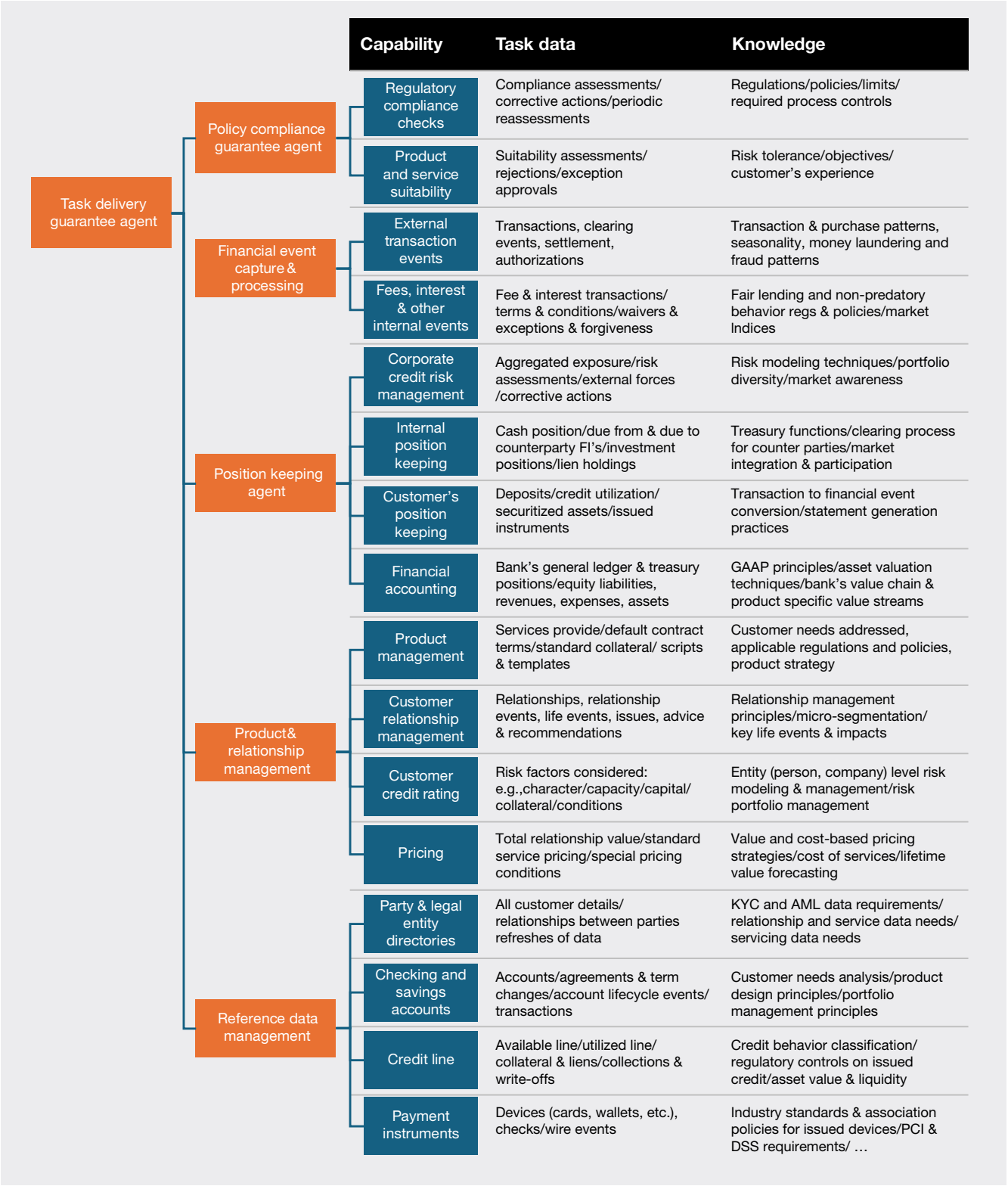


Figure 14: Agents and capabilities in the consumer bank ecosystem

For people in the know, at the first glance, this list looks exactly like the list of service domains we would implement if we followed a microservices standard like BIAN (Banking Industry Architecture Network. See: https://bian.org/servicelandscape-12-0-0/ & go to the Matrix View). However, the way they work in the agent-enabled model is completely different from traditional banking systems.

Firstly, we are going to eliminate the idea of pre-building the ecosystem. Instead, we are going to have individual SME's talk to their own personal agents to create the capabilities

Let's take an example to show how this would work:

9.3 Party & Legal Entity Directory

A party database maintains a listing of people and companies, as well as critical facts about them.

There is nothing really banking centric about a party database. Nor is it specific to information about customers, be they individuals or companies.

In our banking example, the "parties" stored in the database could be customers of the bank, employees of the bank, other financial institutions the bank does business with and vendors the bank uses to complete its inventory of capabilities.

The key features we are looking at are:

- Knowing what kind of information needs to be stored about parties. The kind of information we need to store is based on their role and the relationship between the company maintaining the party directory. The kind of information we need about customers is different from what we need to store about bank employees. What we need to store for a checking account holder is different from what we need to store about credit card holders. The same party could have different roles. E.g., a company could be a customer of a bank, as well as a vendor.
- Keeping the data safe, following all data privacy, retention and protection laws. E.g., Sensitive data must be stored encrypted. It must only be provided when there is a legitimate need for the data.
- Based on an understanding on the nature of data, decide what data needs to be refreshed or reviewed periodically. For example, date of birth or government-issued IDs do not need to be refreshed, but contact details, employment details and salaries would need to be.
- Lineage of the data is understood. The system needs to know where the data came from and the controls used to ensure the quality of the data. Clearly call out less trustworthy data, and do not provide it when asked for data used for decision making.

In the design, we want to focus on how we can address drawbacks identified in the "observations" section: breaking out of the GIGO paradigm/business knowledge is retained.

9.3.1 Setup of the Capability

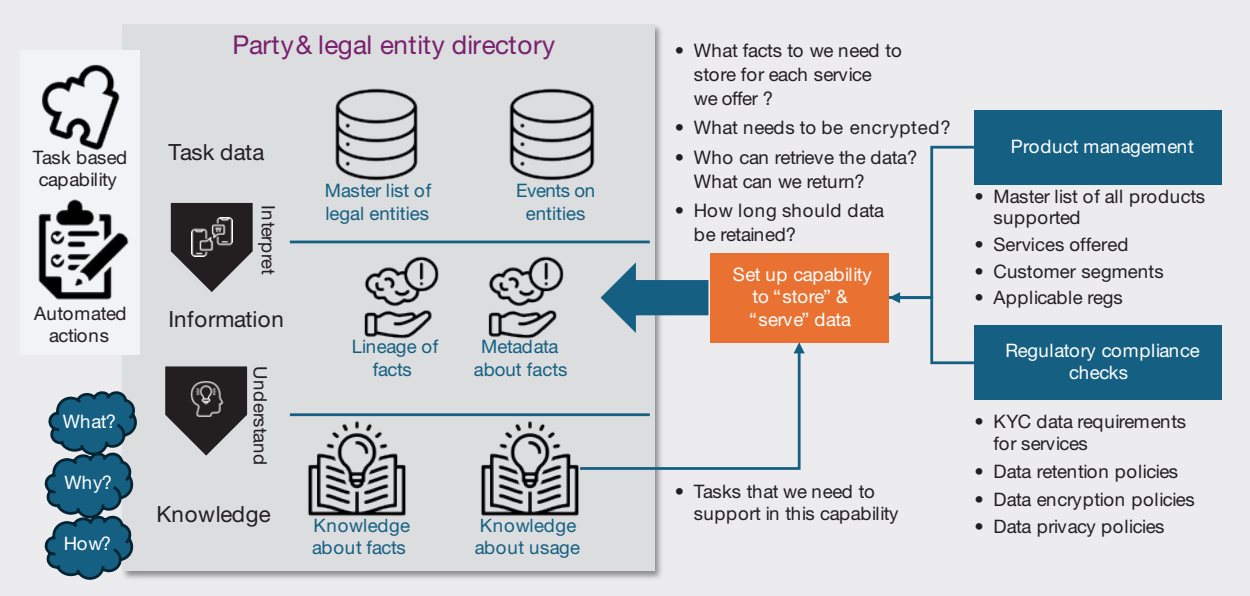


Figure 15: Set up using "Knowledge" from other capabilities

Here, we have implemented an agent to do what groups of business and technology SME's would normally work on. The agent asks the questions that we need answered and sets up the capability based on the answers.

If regulations change or if a new product is created, we just need to run the agent again, and it will revise the capability setup.

9.3.2 Usage Scenario

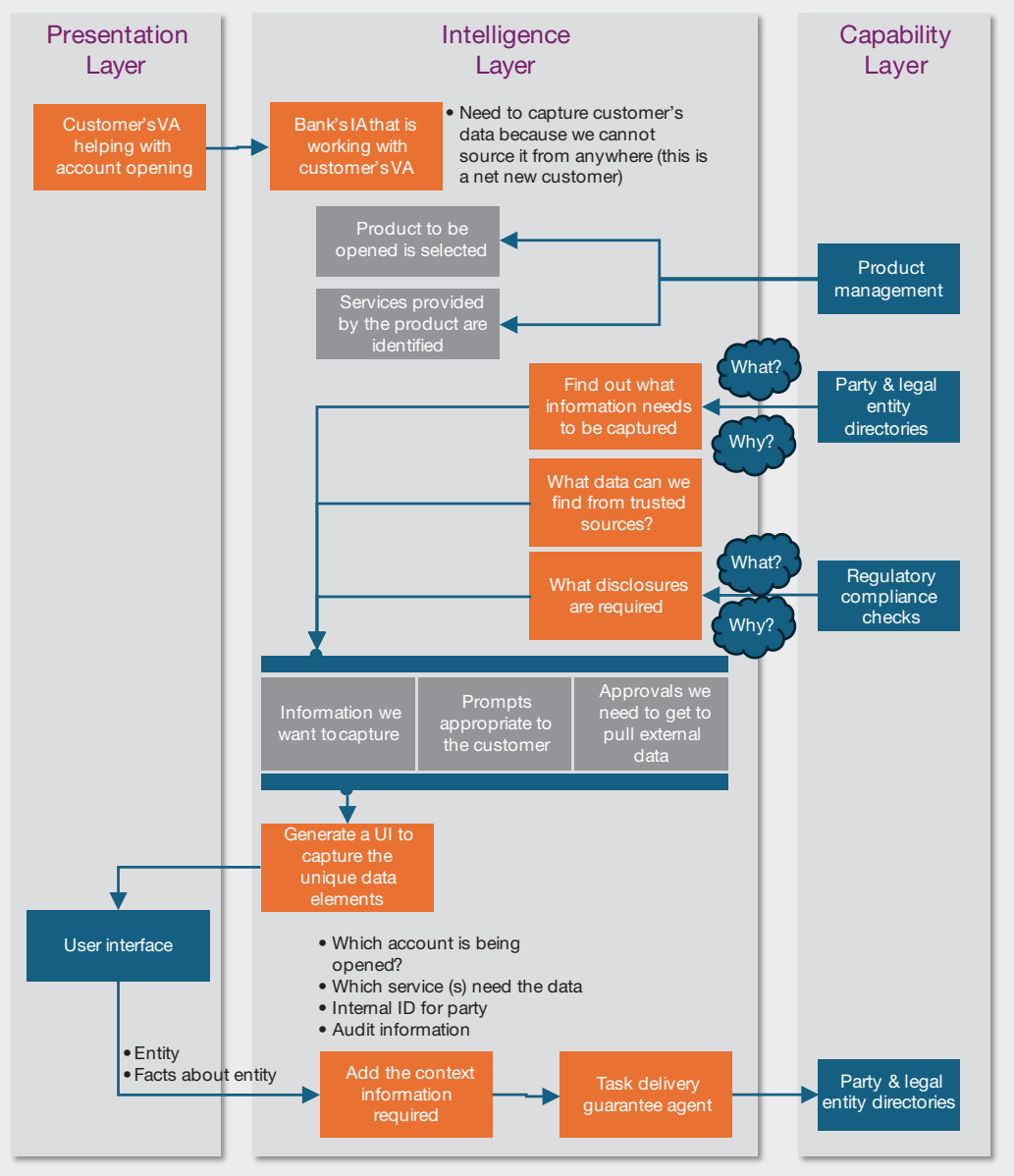


Figure 16: Interactions with the service domain

The intelligence layer plays a key role in making this work. The agents in this layer are responsible for talking to the other service domains and performing the tasks that would normally involve a human talking to another human. Because of the speed at which agents can run, the ecosystem can perform in real-time tasks that normally take months of development, with armies of teams talking to each other.

It is important to note that the UI generation is not meant to be very advanced or onerous to build. The superset of data that can be captured would be known ahead of time, and the developers can cater to all possible fields and data types. The agent can then switch fields on or off as required. Similarly, placeholders can be provided for disclosures and educational prompts.

10. In Conclusion

In this paper, we hope we have highlighted the transformative impact of AI on information and knowledge management.

These technologies need to build on the still ongoing transformations driven by APIification of applications, Internet of Things and the drop in cost of information and knowledge. AI used correctly can help reverse the loss of knowledge and traditional experts.

We already know that the next generation of applications will embrace Intelligent Agents and Virtual Assistants to increase productivity. We believe if used as described in this article, they can directly address some of the shortcomings of our existing ecosystems.

11. Appendix

11.1 Loan Application – How Permission to Share Data is Handled

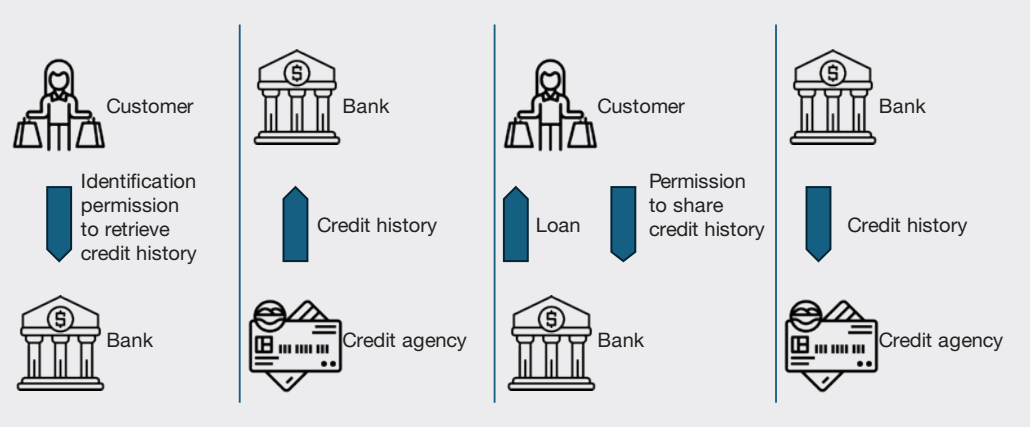


Figure 17: Interactions in a loan application

11.2 Shared Data Within the Enterprise

Legacy "core" systems typically encapsulated both data and functionality. The current best practice is to implement "microservices" that are focused on "bounded domain contexts" – i.e., a single business capability. Instead, imagine a world where we hold data without any associated functionality. We wrap the data in a service that manages the "persistence" of data. [See appendix for details on the persistence layer elements: 7.2 Persistence store types]

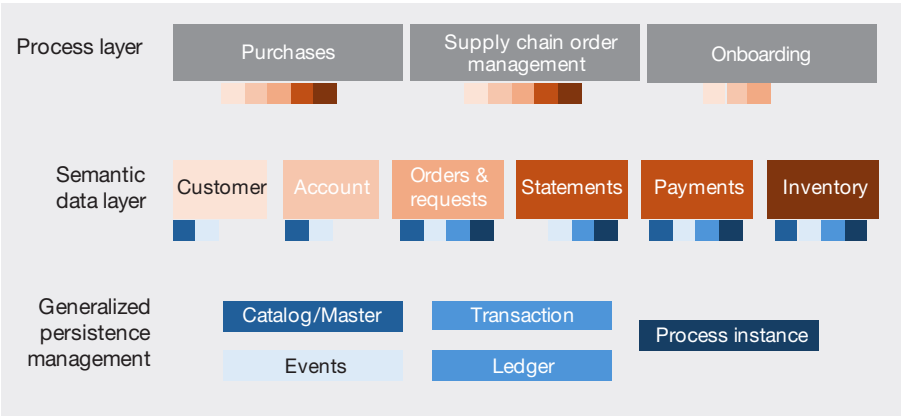
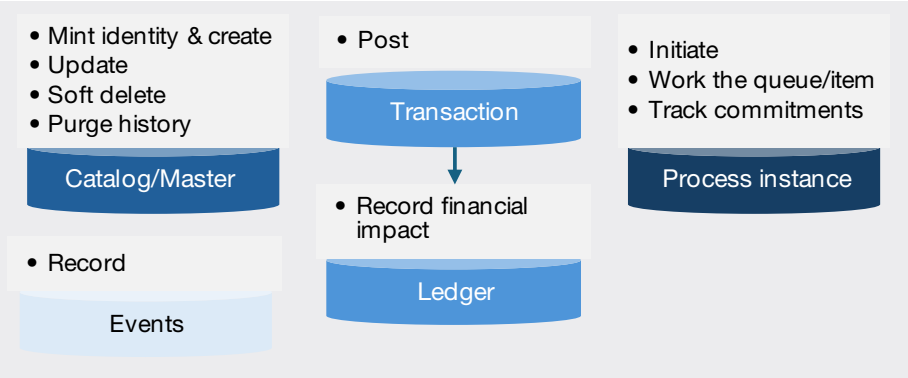


Figure 18: Externalized data within the enterprise

11.3 Persistence Store Types

Nearly all of the data that we handle within enterprise applications can be thought of as being persisted using one of 5 different patterns. Each pattern has its own wrapper used to manage how data is created or maintained.



Catalog:

As Catalog/master: E.g.: list of customers, list of accounts

Events: e.g., customer service request, end of business day, hiring of an employee

Transaction: e.g., purchase of raw material, reservation of goods & eventual sale

Ledger: e.g., general ledger of the company, asset inventory

Process instance: e.g., purchase order, opening of a new account

Each source of data requires its own semantic model. We need combinations of persistence stores for each kind of data.

E.g., Customer: we have a catalog of customer, and event stores of customer-related events such as maintenance activities, sales, account creation, etc.

We can now create processes by implementing logic that interacts with these different data stores.

For more information, contact: marketinginfo.m@mphasis.com

USA
Mphasis Corporation
41 Madison Avenue
35th Floor, New York
New York 10010, USA
Tel: +1 (212) 686 6655

UK
Mphasis UK Limited
1 Ropemaker Street, London
EC2Y 9HT, United Kingdom
T : +44 020 7153 1327

INDIA
Mphasis Limited
Bagmane World Technology Center
Marathahalli Ring Road
Doddanakundhi Village, Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

VAS 30/06/25 US LETTER BASIL 9572

www.mphasis.com