The Future of Software Development

From code to knowledge: the new architecture on which enterprise software will be built

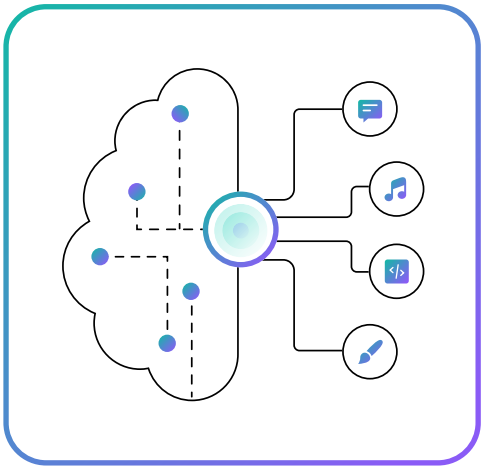
We are living through a double technological revolution.

We are living through a double technological revolution. Two simultaneous forces are amplifying each other and redefining the way enterprise software is built, operated, and conceived.

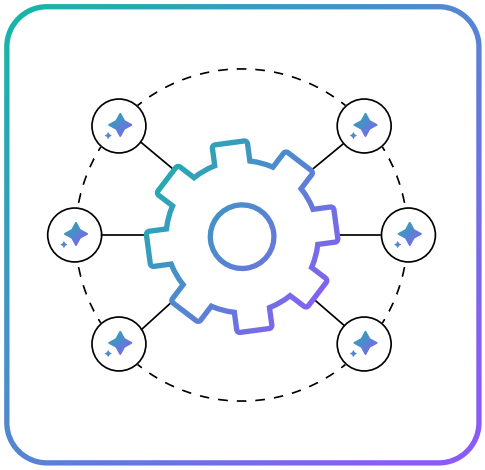
The first revolution is generative artificial intelligence. Since the launch of ChatGPT in November 2022, the ability to generate code, process natural language, and communicate with systems through human language stopped being experimental and became the new normal. For the first time in the history of software, it is possible to program systems at scale by declaring intent instead of writing instructions.

The second revolution is the agentic one, and it is the one that receives the least attention in the press despite being equally or even more significant than the first. The systems being built today include agents that access knowledge, reason about it, and act autonomously. This revolution builds on the foundations of the first and defines a new way of operating organizations.

Nicolás Jodal, CEO of GeneXus, points out that this is the first time in history that two technological revolutions are happening simultaneously.



GenAI Revolution



Agentic Revolution

The last major transformations in the industry arrived at different times and were felt in layers: the smartphone and mobile connectivity redefined the relationship between people and technology starting in 2007, creating entire industries that did not exist before and destroying others that seemed immovable. Its impact was immediate and visible to anyone. Then, as organizations absorbed that change, the cloud arrived: **Amazon Web Services, Google Cloud, and Microsoft Azure** radically transformed how enterprise software is built and operated, moving from costly, hard to scale in house infrastructure to an on demand, accessible, and elastic service. Its impact was deeper but slower, and it dominated the enterprise agenda for more than a decade.

In each of these cases, organizations had time to understand the change, reorganize, and adapt before the next one arrived. Today that margin does not exist. The generative revolution and the agentic revolution do not follow one another, they overlap. And what makes that overlap qualitatively different from any previous change is that each one amplifies the other: agents are more powerful because generative AI exists, and generative AI is more useful because agents exist to orchestrate it. The result is the multiplication of both changes. That is the magnitude of what is happening.

*In this whitepaper we explore that transformation through the perspective of Gastón Milano, **CTO of GeneXus** and of **Glob.AI OS at Globant**, who analyzes the changes taking place in enterprise software development, the roles that are emerging, the assets that are becoming strategic, and the decisions organizations must make today to position themselves correctly for this revolution.*

What software should be built?

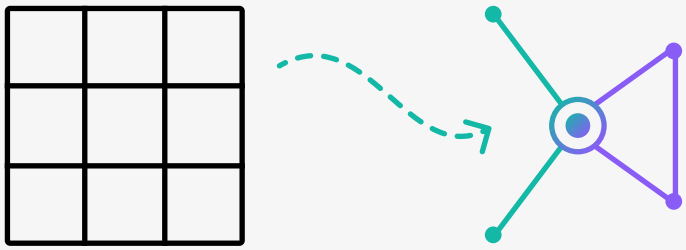
Before talking about speed, agents, code generation, and how to build software in this new era, it is important to understand what type of software should be built. This distinction is, according to Gaston Milano, one of the most common mistakes in the market, because until the "what" is answered, any conversation about the "how" lacks foundation.

What is coming is much more challenging, as global leaders themselves affirm:

Satya Nadella (Chairman and CEO of Microsoft)

He described a future where the business logic that today exists as static code will come to be operated by artificial intelligence agents, with only a few systems of record remaining, while everything currently known as SaaS and service software undergoes a profound reinvention.

"When leaders of the global technology transformation make disruptive statements, the initial reaction is usually skepticism. But history shows that as the changes materialize, it becomes clear they were not wrong," adds Milano.



The agentic paradigm: a new economy

There is an entire infrastructure of protocols and standards, none of which is generative artificial intelligence but rather pure engineering, that is maturing and defining what it means to build agentic software.

"Today it is possible to verify whether what you built is truly agentic or not. This reflects that the concept of agentic already has a concrete, checkable, and auditable technical definition. This maturing of protocols reminds us of the mid 1990s, when the internet began consolidating its protocol stack. In the same way, the infrastructure on which the agentic world will operate is maturing today, and that consolidation is what turns the concept of agentic from an aspiration into something technically verifiable and auditable."

These protocols include:

Agent discovery protocols
Mechanisms for one agent to find another available agent in the ecosystem.

Buying and selling protocols
New forms of economic transaction between agents and systems.

Tool use protocols
Standards for agents to access and use external capabilities in a structured way.

Web navigation protocols
Mechanisms for agents to interact autonomously with web interfaces.

Output format standards
Definitions that allow greater interoperability and dynamism between systems.

The collapse of the traditional software paradigm

Whether we agree or not, several of the pillars of software development as we know it are becoming obsolete.

An application is, in essence, a prediction. Someone decided in advance what most users would need and fixed it onto a screen. A dashboard is a bet on which visualizations will be useful. A web page is a design meant for the average user. None of them adapts to the person actually using it.

"Software engineering solved this the only way it could: building for the average. It worked, but it was always a compromise solution. True personalization, the kind the industry pursued for decades, was never achievable with that model because preparing something different for each person was unfeasible. Generative conversational systems break that logic. They do not have one fixed answer for everyone: they generate a different response for each person, based on the specific context they have about that person. Two people asking the same question can get completely different answers, and both can be correct for each of them," explains Milano.

When generation speed becomes indistinguishable from that of a human conversation, the pre built prediction loses its advantage. At that point, personalizing costs no more than standardizing, and the average stops being the only possible path.

Where are we headed?

The user experience is migrating toward interfaces where the user simply asks an agent for what they need, and that agent renders it in the appropriate format, audio, video, text, book, in real time.

The following table summarizes the most significant shifts in what is losing and gaining relevance in this new paradigm:

WHAT IS LOSING RELEVANCE	WHAT IS GAINING RELEVANCE
Multiple applications	Conversational agents as the single point of access
Form based interfaces and static screens	Fully dynamic interfaces rendered at runtime
Predefined dashboards and visualizations	Visualizations generated on demand based on user context
Statically coded business logic	Business logic operated by agents over knowledge bases
CRUD and fixed screens	Agents accessing knowledge in real time
Pull requests and line by line review	Architectural oversight and after the fact validation
Traditional IDEs	Orchestrators for fleets of agents

Gaston Milano identifies concrete signals of where this transformation is heading:

1. "The model that is coming has no forms, no fixed screens, no CRUD. There are agents that access knowledge and generate exactly what each person needs in real time, as if Netflix rendered the content for you the moment you asked for it, instead of having it pre recorded."

2. "For that to work well, those agents need to be given more context, more declared knowledge about how they should behave, what rules to follow, and how to best serve each customer. That includes policies, processes, roles, decision rules. Everything that today lives in people's heads, in scattered documents, or in unwritten conventions must be codified so agents can act on it."

3. "When agents can act with that knowledge, the company gains new properties: everything that happens becomes versionable, auditable, able to branch into different paths, operable autonomously, and at the same time understandable to the humans who need to oversee it."

4. "This is not an isolated idea. Researchers such as **Andrej Karpathy** (former Director of AI at Tesla and founding member of OpenAI) and others are converging on the same direction: the companies of the future are not programmed, they are documented. Organizational knowledge stops being implicit and becomes an explicit, living, executable asset."

Companies as Code:
The new organizational paradigm

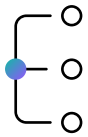
For Gaston Milano, it is essential to treat companies the way software is treated: just as a system's code can be versioned, deployed, and executed, an organization's knowledge can be codified and turned into the engine on which artificial intelligence agents operate.

Milano calls this vision **Companies as Code**: a way of understanding the company where all of its organizational knowledge, its policies, processes, roles, and decisions, is documented, versioned, and available for agents to operate on.

Together, this represents a shift in era: moving from companies that operate with implicit knowledge, scattered in employees' heads or buried in documents, to companies whose knowledge is explicit, living, and executable. When that knowledge is properly codified, properties are obtained that were previously impossible to have all at once:

Versionable
Every change to a policy or process is recorded. You can know what it said before, what it says now, and why it changed.

Auditable
Everything that happens in the company is traceable. No decisions occur outside the system.



Derivable
From that base knowledge, alternative paths can be explored, scenarios simulated, or different decisions made with grounding.

Operable by agents
Agents do not just consult that knowledge, they execute it. They can act on the company's behalf following its own documented rules.

Understandable to humans
None of this is useful if only machines understand it. The system has to be explainable and able to be overseen by people.

What needs to be done?

The shift toward an agentic architecture does not happen in a single move. Milano identifies three concrete steps organizations must take to position themselves correctly in this new era.

Step 1: Codify the knowledge

A company's most valuable knowledge does not live in relational databases or in code. It lives in a wiki: a set of ideas connected to each other through an ontology, that is, a structure that defines how concepts relate to one another. When that knowledge is codified in that format, it becomes operable. Agents can navigate it, interpret it, and act on it. People can work alongside those agents on the same foundation. The company stops depending on knowledge living in someone's head and starts having it as a real asset.

"**GeneXus** is a good example of this, because it operates this way. Its internal wiki contains everything from how to give a good presentation to how to answer difficult questions, how to run meetings, how to take care of your health. In this new era, those assets become consumable, expandable, and scalable by agents, in ways that were not possible before."

For those who work in the GeneXus ecosystem this is nothing new: the Knowledge Base (KB) has always been a central asset. What changes is that this asset can now connect directly to the execution layer. The company's knowledge is not just documented, it is executed.

"The vast majority of the industry is starting from zero. It has no codified policies, no versioned processes, no organizational knowledge in a format agents can consume. Whoever already has a GeneXus KB has something the rest of the market still does not know how to build. That gap is not small: it is the difference between already having the strategic asset of the next technology cycle and just starting to understand what it is."

In this context, KBs have characteristics that make them especially valuable in the agentic world:

They are versionable
Like code, their historical changes can be tracked.

They are consumable by agents
With GeneXus for Agents, KBs can be accessed and used directly by AI agents.

They are human guided
Humans provide the intent, the judgment, and the organizational identity; the agents execute.

> *"Developing software was never the central goal of an organization, but rather the means to build the digital assets that allow it to operate and serve its customers. The difference is that today that means has to be framed within the same logic that governs the rest of the organization: a knowledge base as the source of truth and agents that operate on it. When that happens, software development stops being an isolated technical activity and becomes just another operation of the company, as manageable, versionable, and auditable as any other."*
>
> **Gastón Milano**

Step 2: Model Knowledge Bases + Agents

The architecture of future software, according to Milano, is built around two fundamental elements: a **Knowledge Base (KB)** that codifies all of the organization's knowledge, its processes, policies, data, business rules, and context; and **AI agents that operate on that KB** to execute tasks, make decisions, and generate value.

The model redefines the roles of each actor in the organization:

Role	Function	Tool
Humans	Drive intent, ideation, and company identity. Make strategic decisions.	Knowledge base plus human judgment
Generative AI	Executes tasks, generates content, processes data.	LLMs plus engineering harness
Agents	Operate complete processes autonomously, consume the KB.	Agentic protocols plus KB plus tools
AI Architect	Designs the stack, orchestrates agents, defines levels of autonomy.	Orchestrators plus multi agent platforms

> Human time is no longer spent writing code; it is spent defining the what, building the knowledge base, and orchestrating the agents that execute. Gaston Milano illustrates this with his own practice: since November 2025 he has not written a single line of code.
> His job is to orchestrate.

Step 3: The token economy

One of the practical aspects Milano highlights is the need to efficiently manage token costs. Large companies do not have the same subsidies as individual users; the cost of operating fleets of agents at enterprise scale can be very significant and becomes a strategic factor in its own right.

For this reason the concept of tokenomics emerges: the economy of tokens. Skills and precisely codified knowledge make it possible to drastically reduce the number of tokens consumed per operation, lowering costs and increasing efficiency. This turns the quality of the knowledge base into a critical economic factor. A well structured KB does not just make agents work better: it makes them work more cheaply. And at scale, that difference defines the viability of the model.

The problem of vibe coding and the need for oversight

Vibe coding is the term coined by **Andrej Karpathy** in February 2025 to describe a way of developing software where the programmer describes what they want to an AI in natural language, accepts the generated code without reviewing it in detail, pastes errors directly back to the model for it to fix, and lets the system grow organically, often beyond what the developer themselves fully understands.

Milano describes this problem with a visual metaphor: "Vibe coding can be seen as Swiss cheese: each slice has holes. When many slices are combined, the holes accumulate and errors grow exponentially. This can work for quick prototypes or personal projects, but in mission critical enterprise systems, where code has to be predictable, maintainable, and auditable, vibe coding without a solid knowledge base behind it produces unreliable results."

The solution to this problem has several layers:

Reducing holes with better tools
The arrival of tools like Claude Code, Codex, Gemini CLI, and other harnesses has significantly reduced the number of errors in automated code generation.

Real and precise context
The only way to further reduce errors is to provide agents with correct and complete context, which comes from a well structured knowledge base.

Human architectural oversight
Humans must keep designing the right architectures and overseeing results to keep software predictable and functional.

Vibe coding without context is the symptom. The knowledge base is the solution. And the AI Architect is the one who brings both together.

AI Architect: The new strategic role

The AI Architect is the role that emerges once code generation stops being the bottleneck. This is the person who defines the what and the why, and who orchestrates the agents that handle the rest.

The emergence of the AI Architect is not a coincidence: it is the direct consequence of three factors that converged at the same time:

1. Models reached a threshold of competence sufficient for well defined tasks.

2. The engineering around the model matured, with tools like Claude Code, Codex, and Gemini CLI that solved how to build the agentic loop, how to give agents the right tools, how to compress memory, and how to manage the file system.

3. And agents work in parallel around the clock, producing volumes of code that no human team can match.

The result is that the bottleneck stopped being code generation and became defining what to build and with what knowledge to build it. That is where the AI Architect operates.

Their core responsibilities are:

- **Designing the level of autonomy of each system component**
  Determining which parts require active human oversight, which can operate with suggestions to the human, and which can function fully autonomously. This decision, which seems technical, is actually strategic: it defines how much control the organization retains and where it delegates to agents.

- **Orchestrating fleets of agents**
  Defining how dozens or hundreds of agents working in parallel are coordinated, how tasks are assigned to them, and how their results are managed. Gaston Milano regularly operates with 30, 40, or 50 agents running simultaneously overnight. Orchestrating that fleet is his main job.

- **Designing the full technology stack**
  Choosing which LLMs to use, which orchestration platforms, which databases, and how to combine them. The AI Architect can combine Claude, Codex, Gemini, or any other model depending on what each situation requires, without being tied to a single provider.

- **Ensuring quality without line by line review**
  It is impossible to review 100,000 lines of code generated over a weekend by agents. The AI Architect designs systemic validation mechanisms and after the fact oversight that replace traditional review workflows.

- **Maintaining an end to end vision**
  Preserving a holistic view of the solution, without artificially fragmenting it into data, UI, and security as siloed compartments that toss specifications over the wall at each stage.

In addition, the AI Architect also defines the different levels of automation that coexist within the same system:

- **Direct human intervention**
  Tasks that still require a person to act without agent assistance: infrastructure deployments, security key configuration, IoT device installation, physical robotics.

- **Assistance with suggestions**
  The agent proposes, the human decides. The system amplifies human capability without replacing judgment.

- **Human as occasional auditor**
  The agent acts autonomously and the human reviews exceptions and results, not every step of the process.

- **Full autonomy**
  The agent operates without human intervention in the normal cycle. A concrete example: when GeneXus generates the code for a database reorganization, no one reviews it manually. The system is trusted to work, because the context and rules are correctly defined.

Implications for the GeneXus Community

For Milano, in today's market, all GeneXus users are naturally "AI Architects." Why? Because the GeneXus user already has the right mindset built in:

- They know that a knowledge base must be built first.
- They know that the correct data must be in place before building.
- They understand the concept of generating from specifications.
- They think end to end instead of being fragmented into technology silos.

In addition, GeneXus users never reviewed pull requests or followed GitHub workflows. What might seem like a limitation in another context turns out to be exactly the right mindset for this moment, because those traditional review workflows are dying out.

"There is no way to review 100,000 lines of code generated over a weekend by agents, and the entire industry is looking for a way to solve that. GeneXus users already operated without that dependency. However, there is something the community still needs to adopt, specifically data Impact Analysis. There is now a tendency to throw things directly at Claude Code without that prior analysis, and that is a mistake that comes at a high cost. Instilling that practice in the community is still pending work."

Milano anticipates that there will be vertical, industry specific knowledge bases. This means that partners with sector expertise will have a unique opportunity to build deep, differentiated knowledge assets for their industries, which will then become competitive advantages that are hard to replicate.

GeneXus is already working on ensuring the "future proofing" of its clients' existing knowledge bases. The goal is for those KBs to be able to evolve toward the agentic world without losing accumulated value. The strategy includes:

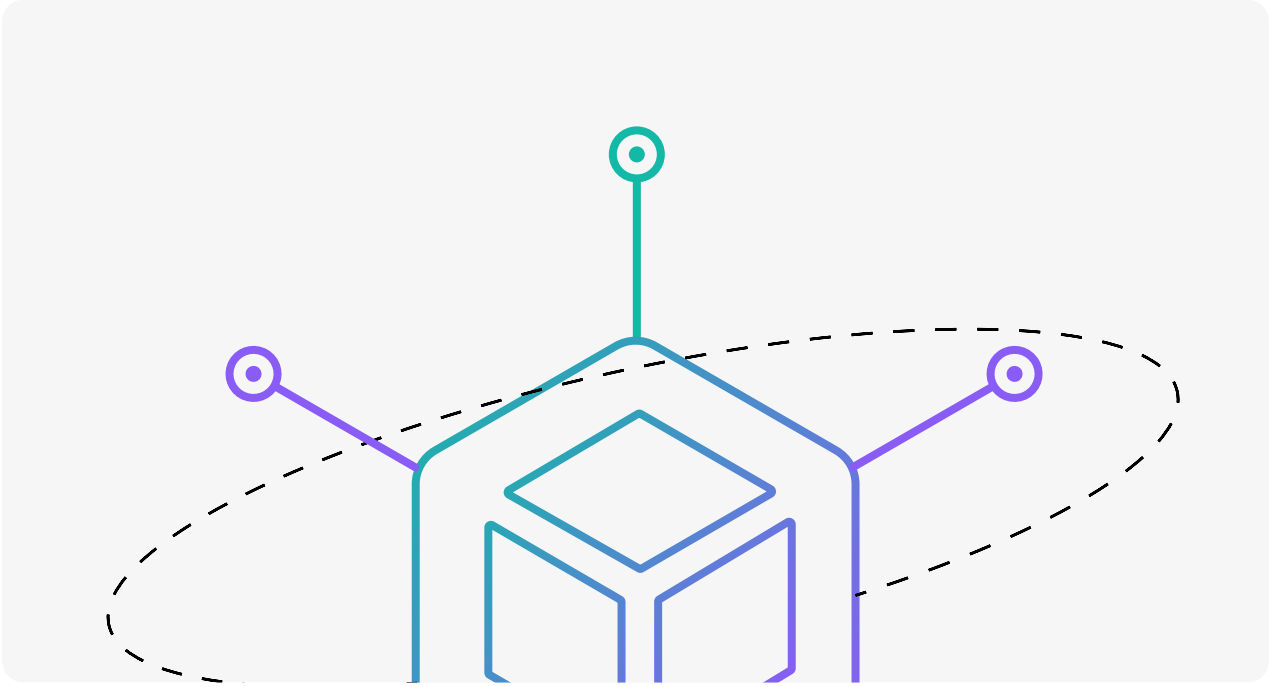
- GeneXus for Agents: a tool that makes KBs consumable by external agents.
- A second iteration toward full agentic capability: an evolved version of KBs that lets agents work directly with clients' existing knowledge.
- Orchestration tools that allow GeneXus to be combined with other generators and agents in the market.

Back to the future

GeneXus was founded in Uruguay in 1988 with an idea that seemed radical at the time: that business knowledge should be at the center of software development, and that systems should be generated from that knowledge. While the rest of the industry was building code, GeneXus was building knowledge bases. While others optimized how to write faster, GeneXus worked on eliminating manual code writing by generating systems from declared knowledge.

Today, at the moment when the global industry is converging toward exactly what GeneXus has been practicing for years, the model evolves once again: platform plus services, with vertical industry specific knowledge bases, implementation, governance, and continuous operation. A recurring model where codified knowledge is the central asset and agents are the execution engine.

What for years was a particular way of working turns out today to be the right architecture for the agentic era.



The time to act is now

The world is maturing at an extraordinary pace. Agentic protocols are consolidating. Tools are evolving. And the gap between organizations that are already building their knowledge bases and those still deciding whether to do so is widening every day.

The organizations that will lead the next era of enterprise software are the ones that understand, ahead of others, that the strategic asset of this era is codified knowledge, and that have the right guidance to build it and operate it with agents.

That is exactly the work we do. From Globant and GeneXus we support organizations through that transition: from building their first knowledge base to designing complete agentic architectures, selecting the technology stack, and training the teams that will operate in this new paradigm.

If your organization is looking to take that step, the time to start the conversation is now.

Contact us