
INTENT
ARCHITECT
EMPOWERING DEVELOPERS

The real reasons software codebases deteriorate

Software systems that continue to change and grow, share the same fate…

PROGRESSIVE DETERIORATION - of the team's velocity, ability to make changes, retention of talent, and ability to hire. Changes take longer and longer, until eventually a rewrite is the only option – another legacy system.

Affecting the entire industry, the success rates of software projects by size*:

Amplified by the size of the codebase, the fundamental forces that cause this:

* Statistics from 2015 CHOAS Report by The Standish Group, which assess project outcomes from a broad database of software projects. Success in this context means on-time, on-budget, and with satisfactory results.

TECHNICAL DEBT

Inconsistency, design-erosion, technical debt that grows with each change in the codebase, tending toward the big ball of spaghetti.

ARCHITECTURAL RIGIDITY

Complexity grows exponentially in a codebase as features are added, paralyzing developers' ability to make changes.

FALLING BEHIND TECHNOLOGY

Becoming stuck in outdated technologies or versions, unable to achieve new non-functional requirements or remain supported.

LOSS OF SYSTEM DESIGN VISIBILITY

Systems decline rapidly when design visibility is lost (e.g. key developers moving on), making it harder to make changes correctly and avoid design erosion.

Preventing codebase deterioration

In an **ideal world**: code has no weight, architecture is without cost, systems self-document their design blueprint, and technologies are interchangeable. The following **mechanisms**, when combined, help achieve this:

PATTERN REUSE

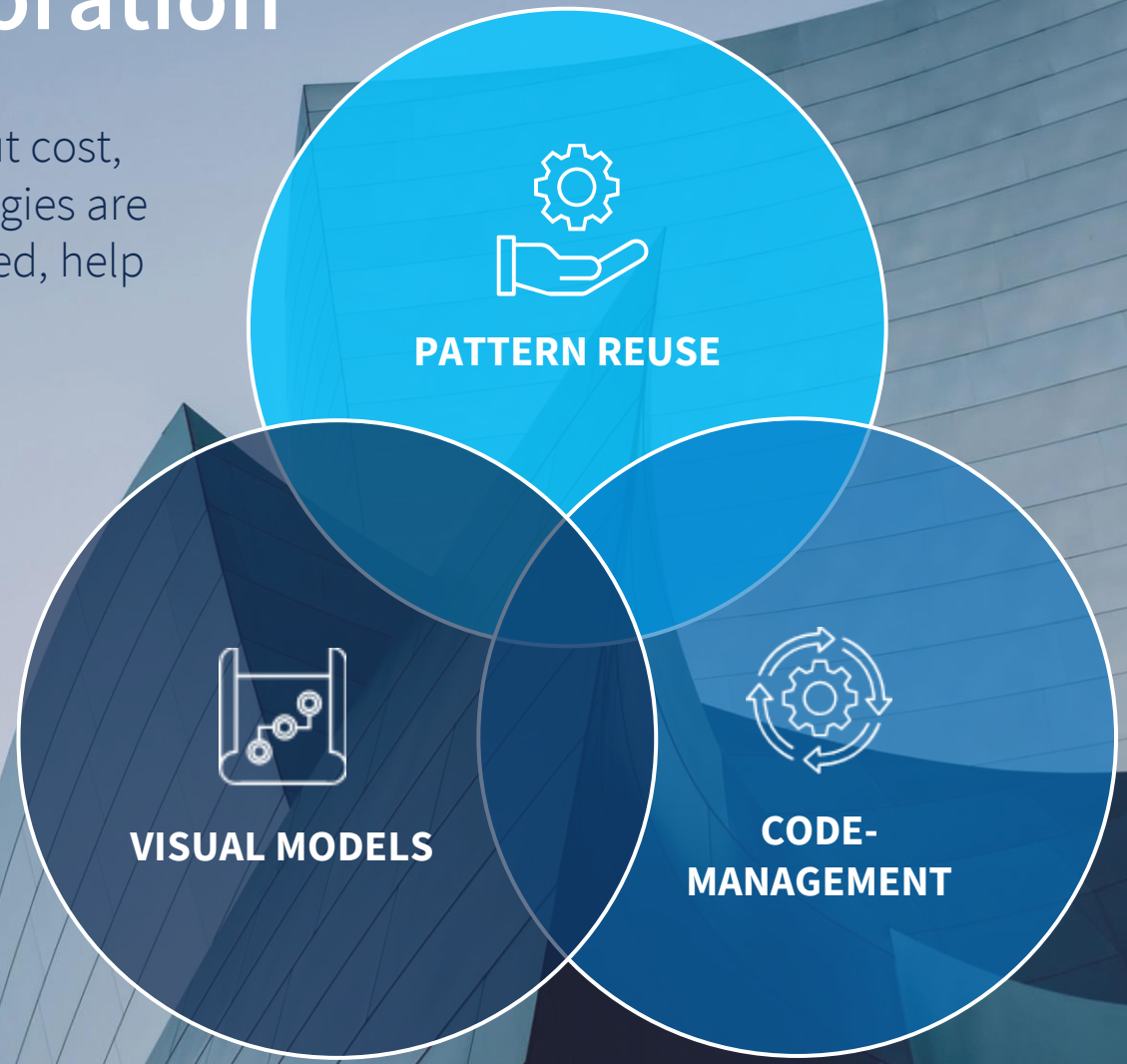
Software systems are made up of patterns – each instance is different, but the patterns are the same. Pattern reuse is the ability to turn these patterns into artifacts that can be reused within projects and across organizations.

CODE-MANAGEMENT

If developers can only write several lines of code at a time, then code-automation is the ability to write thousands in a second. Code-management is a unique approach to this which avoid the common pitfalls which historically have made it impractical.

VISUAL MODELS

Using visual models to describe software design contracts can be incredibly powerful. By compressing information into visual formats, the human mind can quickly interpret and digest it. If synchronized with the code, it offers a blueprint of the system.



What is Intent Architect?

Intent Architect is a *next-generation software development tool* for developers and solution architects.

TOOL FOR DEVELOPERS

Installed as a tool on the developer's machine and used alongside their favourite IDE

MODULES

Modules are Intent Architect's artifacts for *pattern reuse*. They encapsulate code patterns which can be installed into applications.

VISUAL MODELS

Powerful and extensible designers used to describes the system's design in visual models

NON-PRESCRIPTIVE

Allows developers to automate the architectures and patterns of their project, in exactly the way that they want.

CODE-MANAGEMENT

A unique approach to code-automation which employs intelligent merging algorithms - enabling developers and the automation systems to work seamlessly in the same code files.

NO LOCK-IN

Intent Architect does not introduce runtime dependencies or lock-in.

ADDITIONAL INFORMATION

Intent Architect is a tool that allows developers to design their applications and automate the structures, patterns & architectural code that make that design a reality.

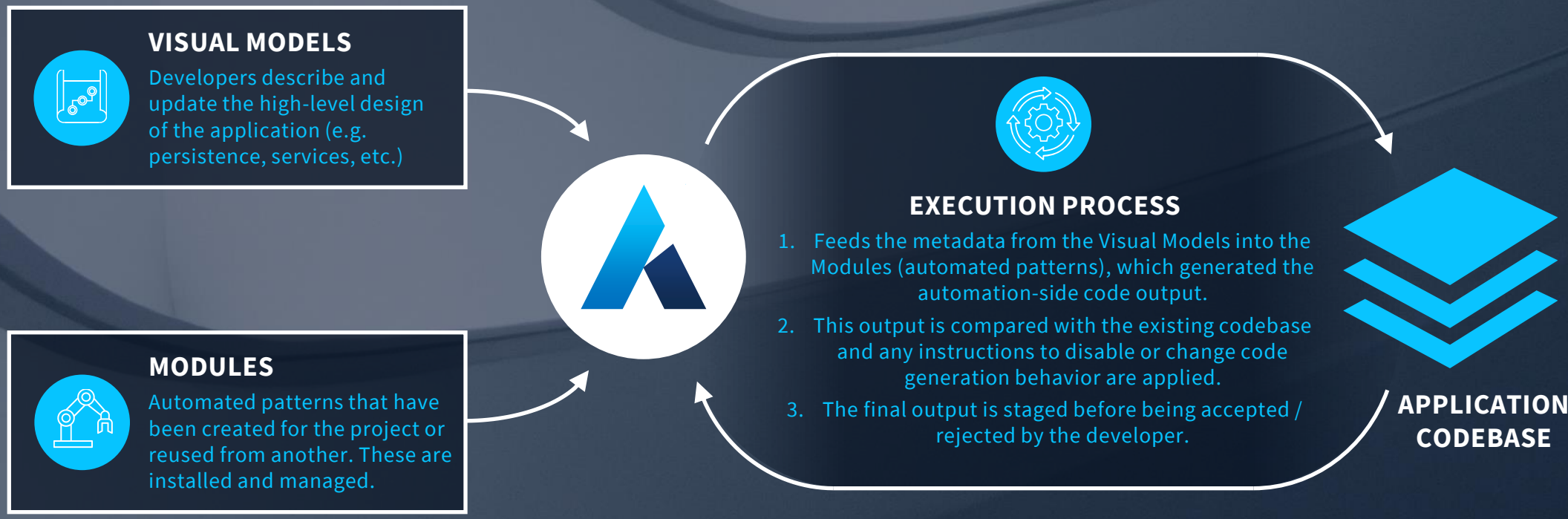
It helps tech-enabled companies of all sizes to develop their software faster, more efficiently, and at higher levels of quality and consistency. It does this without being prescriptive, or forcing the team to change technologies, architecture, or coding standards.

Intent Architect is programming language agnostic, with real-world usages in C#, Typescript, Java, HTML, SQL and others.

How it works

Intent Architect works with the developer to create and update files in the codebase.

- ✓ Code is managed based on the developer's design (Visual Models) and which Modules they've installed.
- ✓ Non-automated tasks & business logic are managed by the developer.
- ✓ Changes are determined each time the developer runs the *Execution Process*, as illustrated below.



How it helps

Intent Architect directly tackles the fundamental forces that cause software to deteriorate. Teams can expect to deliver high-quality, consistent code – that flawlessly follows their architecture – in a fraction of the time it would've normally taken them.

CONSISTENCY & STANDARDIZATION

Codifying and automating patterns means that anything managed by Intent Architect is exactly according to the team's standards and can be updated in one place.

LIGHTNING-FAST DELIVERY

Code-management enables immense speed, without any compromise in quality - the code is generated correctly, as the team intended, and without human error or bugs.

VISIBILITY & DOCUMENTATION

Visual models represent the system's design / blueprint, which is aligned with the underlying code – an up to date and accurate document of the system's design.

KNOWLEDGE RETENTION

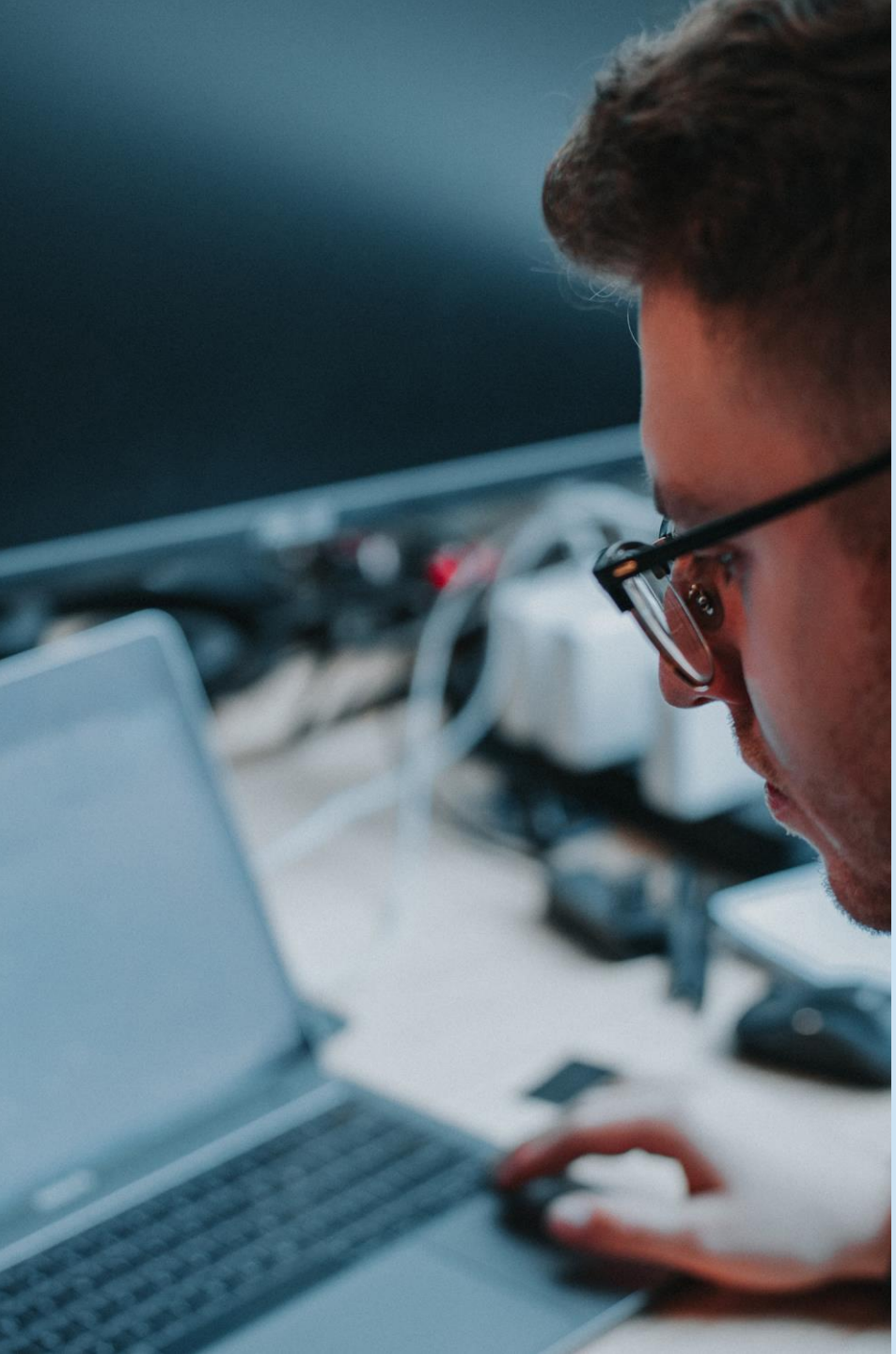
Improves shared knowledge, reduces key-man dependencies and fast-tracks the learning curve for new joiners.

AGILITY & FLEXIBILITY

Change designs, upgradable patterns, architecture and even technologies across the entire system, in one place. Avoid becoming a "legacy" system.

BUILD IP & PATTERN REUSE

Automated patterns, when reused between teams and across projects, forms IP, "free" architectures, and even a competitive advantage for an organization.



Who we are

We are a founding team of software architects, with collectively over 50+ years of real-world experience.

Our mission is to empower development teams with superior ways to build software systems.

We do that by creating **powerful**, **practical** and **innovative** tools.

Excellence

We believe in holding excellence at the core of every product and solution we create.

Value

We aim to ensure there is tangible and profound value in everything we do.

Pragmatism

We make sure that every solution we create ***actually works*** – in the real world.

For developers. By developers.

Common use cases

Intent Architect is ideal for new projects, microservices, modernizations, rewrites, and large refactors. It can also be progressively introduced into existing codebases.

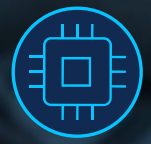
Typically, Intent Architect is used to create and manage the code that supports:

MICROSERVICES
Bootstrapping, integrating, package configuration, etc.

PERSISTENCE
ORM Mappings, Repositories, Specification patterns, etc.

SERVICES
RESTful APIs, SOAP APIs, DTOs, Proxies, etc.

WORKFLOWS
Flow configuration, logic hook-points, etc.



EVENTING
ORM Messages, Bus configuration, Publishers, Subscribers, etc.

FRONT-END
Technology boilerplate, standard UI patterns, etc.

CASE STUDY
INOXICO

How Inoxico used Intent Architect to build high-quality SaaS products for their users at a pace that their competitors could never match.

OVERVIEW

Inoxico adopted Intent Architect in 2015 as a strategic software development tool to help them overcome their rigid legacy systems and to build, enhance and maintain their envisioned SaaS products.

With a small team of software engineers using Intent Architect to automate the majority of their boilerplate coding tasks, they were able to focus on design, customer experience, and getting the most out their technology stack (while being able to continuously change direction as the business evolved!).

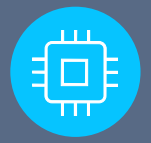
Within 9 months they were able to rewrite over 3 years of legacy systems, helping them leap ahead of the competition and set them on a course to later become the de facto industry leader – their advantage: the ability to bring high-quality SaaS products to the market at several times the speed of their competitors.

MICROSERVICES

PERSISTENCE INFRASTRUCTURE

SERVICE INFRASTRUCTURE

WORKFLOWS



EVENTING INFRASTRUCTURE

* Ways that Intent Architect was used

OUR APPROACH

Standardized Architecture
Used Intent Architect to ensure separation of concerns and to keep infrastructure code 100% consistent and standardized

Microservices
Enabling a microservice architecture by making the setup of a new microservice a 5 minute task, complete with service, persistence and eventing infrastructure.

Domain Driven Design
Combined Intent Architect's modelling & automation capabilities to follow DDD principles and patterns (typically very time-consuming) at zero to the team.

Customized Workflows
Realizing they could capitalize on a gap in the market for customized workflows, Inoxico used Intent Architect to reduce up to 90% of the development time & cost.

RESULTS

3.16x
FASTER DEVELOPMENT

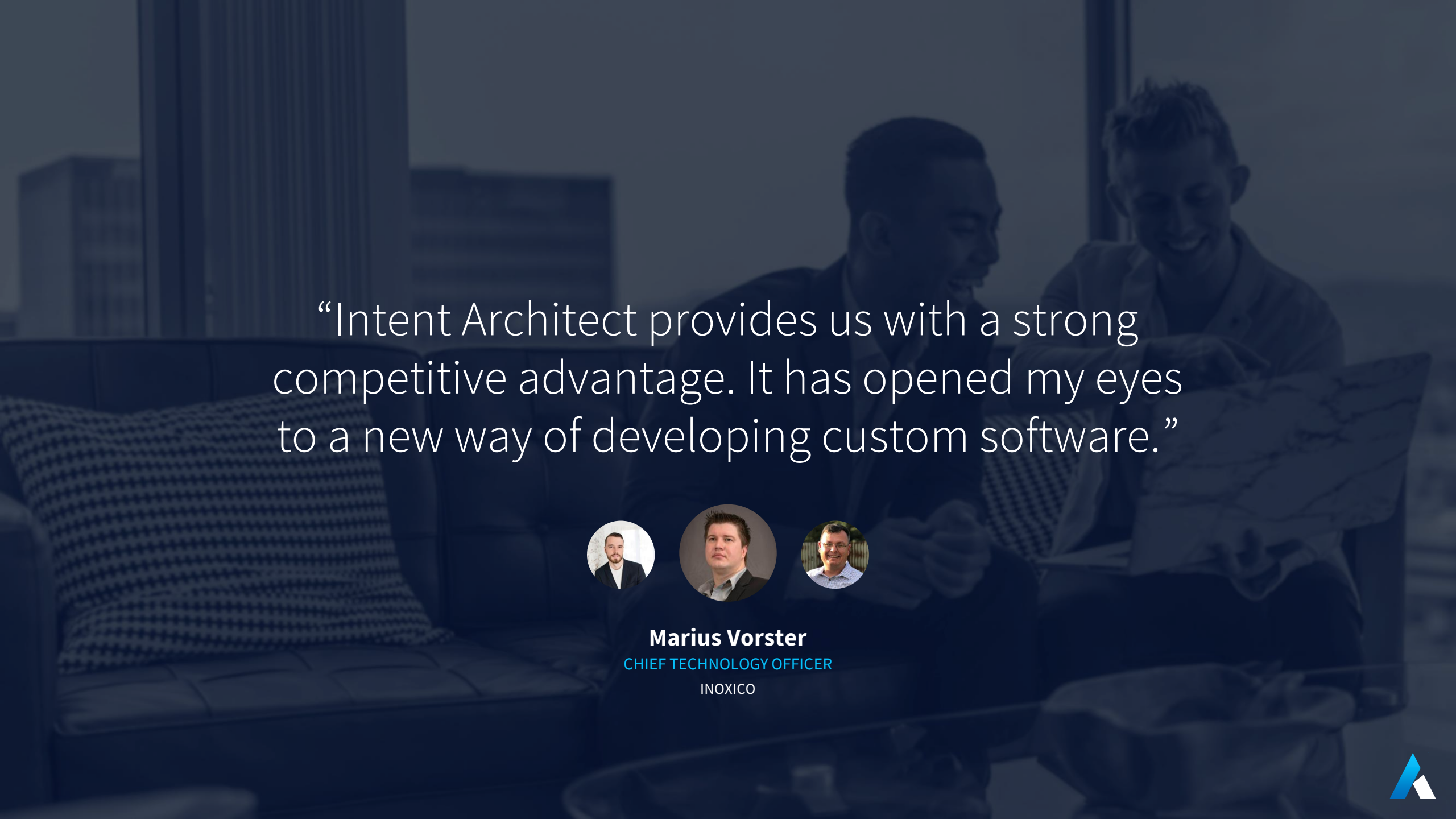
71%
CODE MANAGED

67%
LESS BUGS

CONCLUSION

Since being adopted, the effect of Intent Architect has been dramatic – it shifted the way that Inoxico competes in the market. It allowing them to build (and change) their products at lightning speed, in a way that their larger (and better resourced) competitors couldn't equal.

While most businesses expect a trade-off between cost, time and quality, Inoxico is a perfect example of how Intent Architect can enable increased quality & flexibility without increasing cost and time – but instead reducing it!



“Intent Architect provides us with a strong competitive advantage. It has opened my eyes to a new way of developing custom software.”

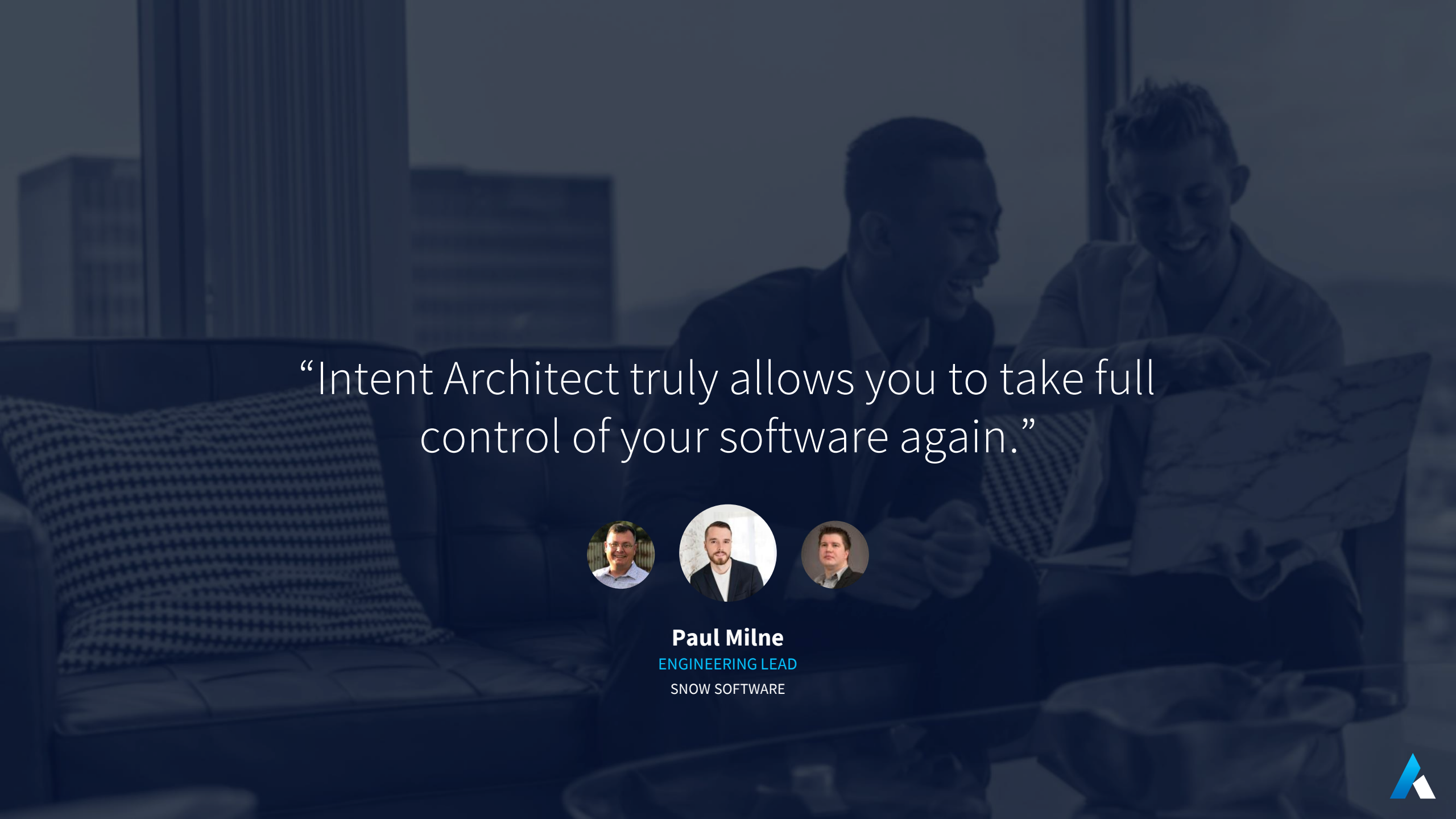
Marius Vorster

CHIEF TECHNOLOGY OFFICER

INOXICO

“As a FinTech organisation, adaptability is key to being an industry leader. Intent Architect helps set us apart.”

Johan van Rhyn
SOLUTIONS ARCHITECT
BRIGHTROCK



“Intent Architect truly allows you to take full control of your software again.”

Paul Milne

ENGINEERING LEAD

SNOW SOFTWARE

info@intentarchitect.com / www.intentarchitect.com