

CASTLE RIDGE
SOFTWARE

IDE Architectures

Then and Now

Thomas Mäder
Castle Ridge Software GmbH
TheiaCon 2026

Why IDEs

"An IDE is intended to enhance productivity
by providing development features
with a consistent user experience
as opposed to using separate tools,
such as vi, GDB, GCC, and make."

— Wikipedia

© 2026 Castle Ridge Software

Why IDEs

"An IDE is intended to enhance productivity
by providing development features
with a consistent user experience
as opposed to using separate tools,
such as vi, GDB, GCC, and make."

— Wikipedia

Make programming easier for humans!

© 2026 Castle Ridge Software

What do IDEs integrate?

© 2026 Castle Ridge Software

What do IDEs integrate?

- Software construction
  - Editor
  - Compiler
  - Build systems
  - Code navigation tools and X-ref
  - Code manipulation (refactoring, code assist, quick fixes)

© 2026 Castle Ridge Software

What do IDEs integrate?

- Software construction
- Change management
  - Issue tracker
  - Review
  - CI
  - Version control

© 2026 Castle Ridge Software

What do IDEs integrate?

- Software construction
- Change management
- Operations
  - Deployment
  - Monitoring

© 2026 Castle Ridge Software

Inner loop vs. outer loop

- **Inner loop:** edit / compile / run (test)
- **Outer loop:** full build, review, integrate, deploy / publish

© 2026 Castle Ridge Software

Before IDEs

- Punch cards
- Teletype
- → batch processing, non-interactive

© 2026 Castle Ridge Software

Before IDEs

- Punch cards
- Teletype
- → batch processing, non-interactive

Integrated Development Environment = Interactive Development Environment

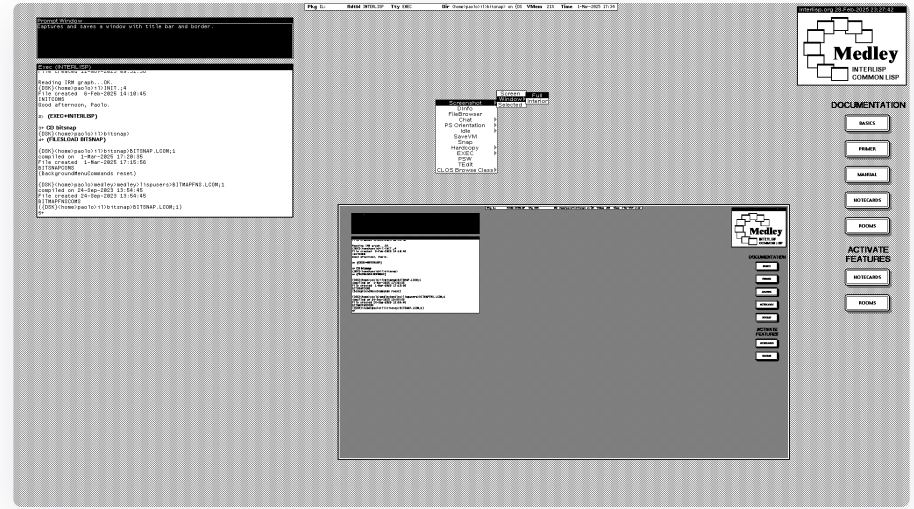
© 2026 Castle Ridge Software

IDEs of the past

© 2026 Castle Ridge Software

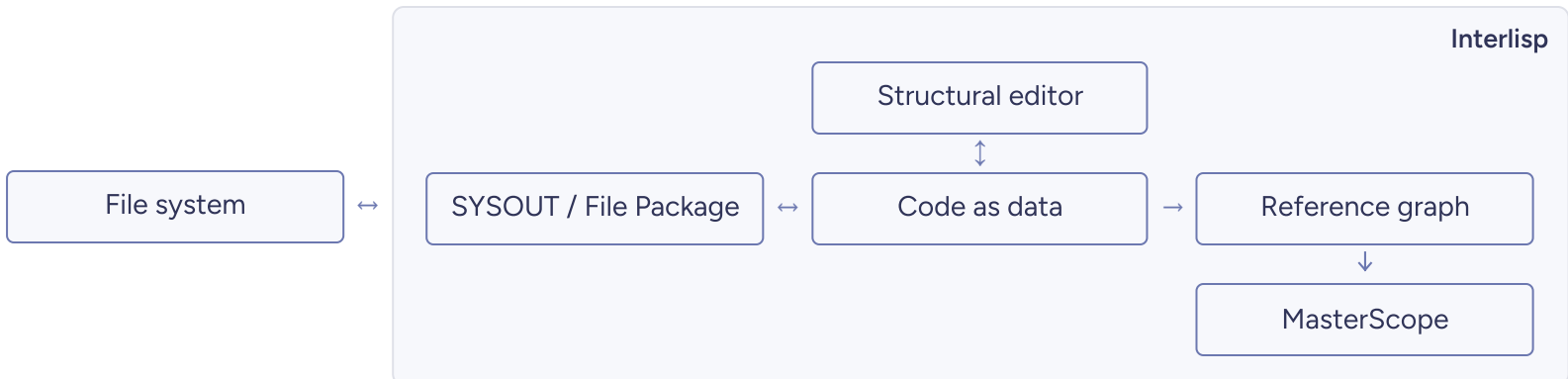
Interlisp (ca 1974)

- Code as regular data structures
- Edit functions and expressions
- Cross-referencing (MasterScope)
- Full reference graph in memory
- Update reference graph lazily or on request
- SYSOUT image / file package
- Do What I Mean (“quickfix”)
- Integrated debugger with hot code replacement



© 2026 Castle Ridge Software

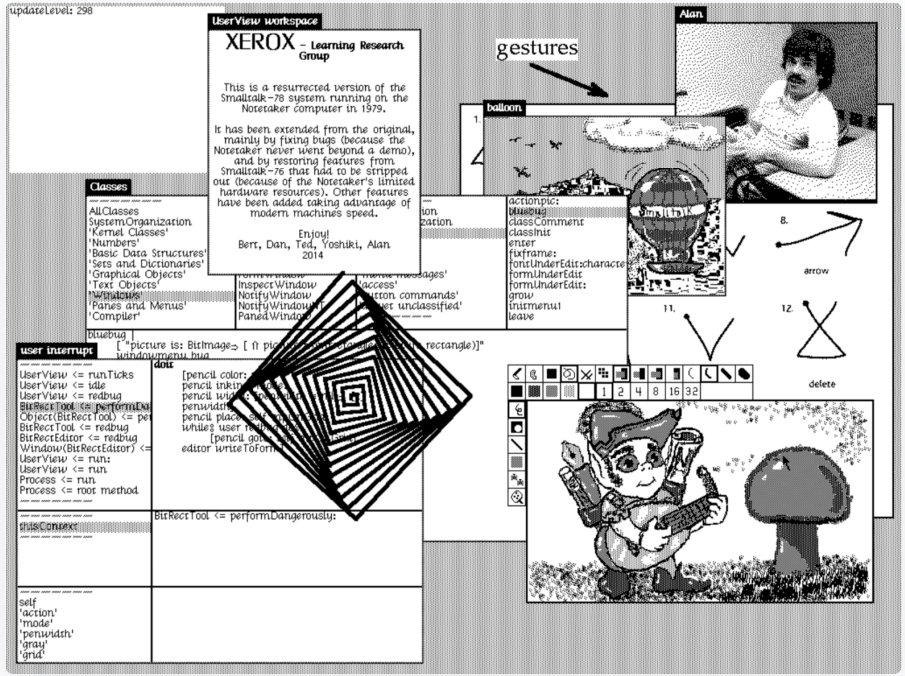
Interlisp — architecture



© 2026 Castle Ridge Software

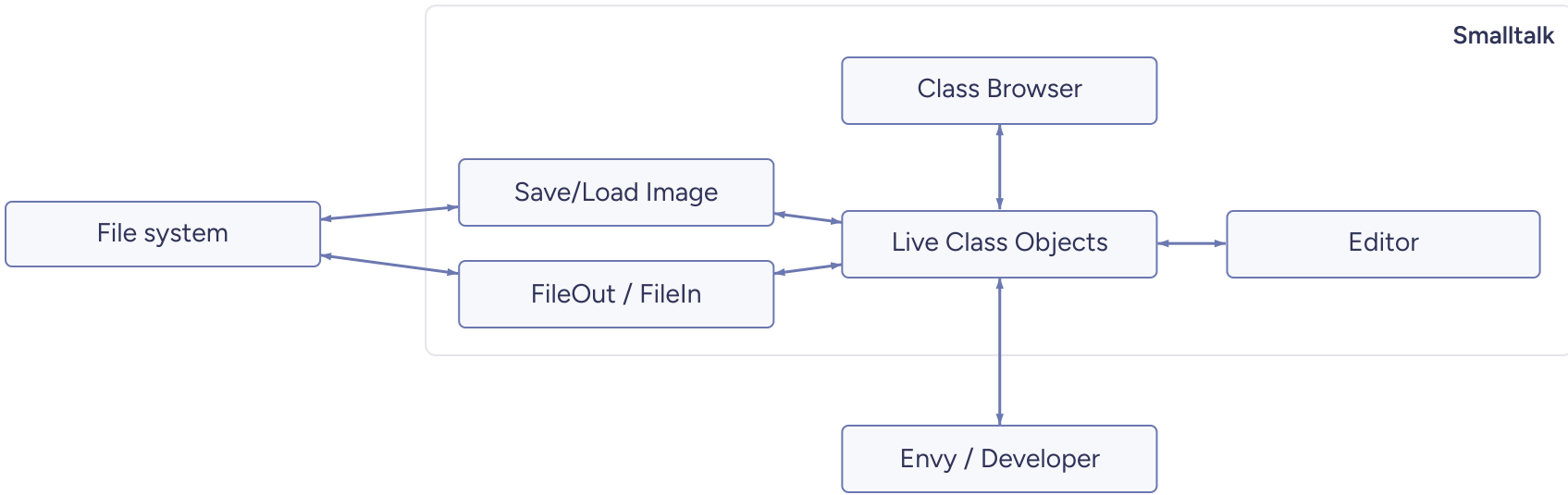
Smalltalk (80)

- Code as regular data structures
- Edit live class objects
- Interactive debugger
- FileOut / FileIn
- Later: Envy/Developer
- Refactoring Browser



© 2026 Castle Ridge Software

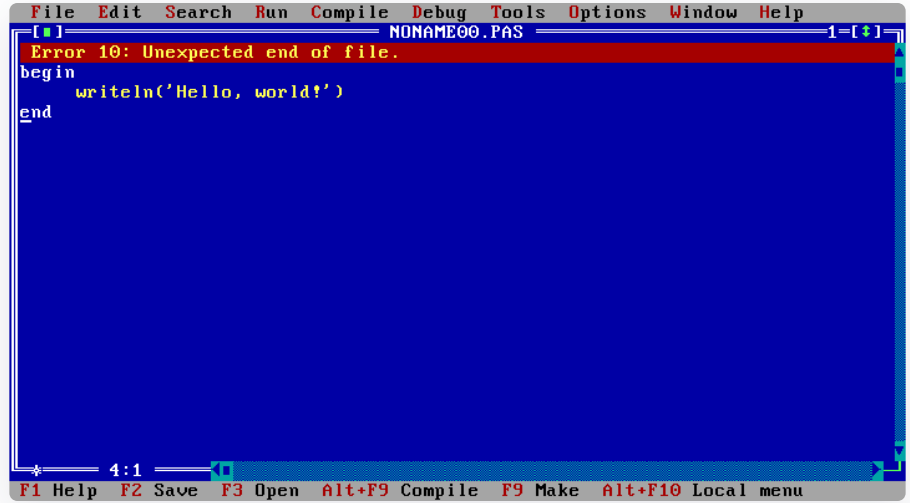
Smalltalk — architecture



© 2026 Castle Ridge Software

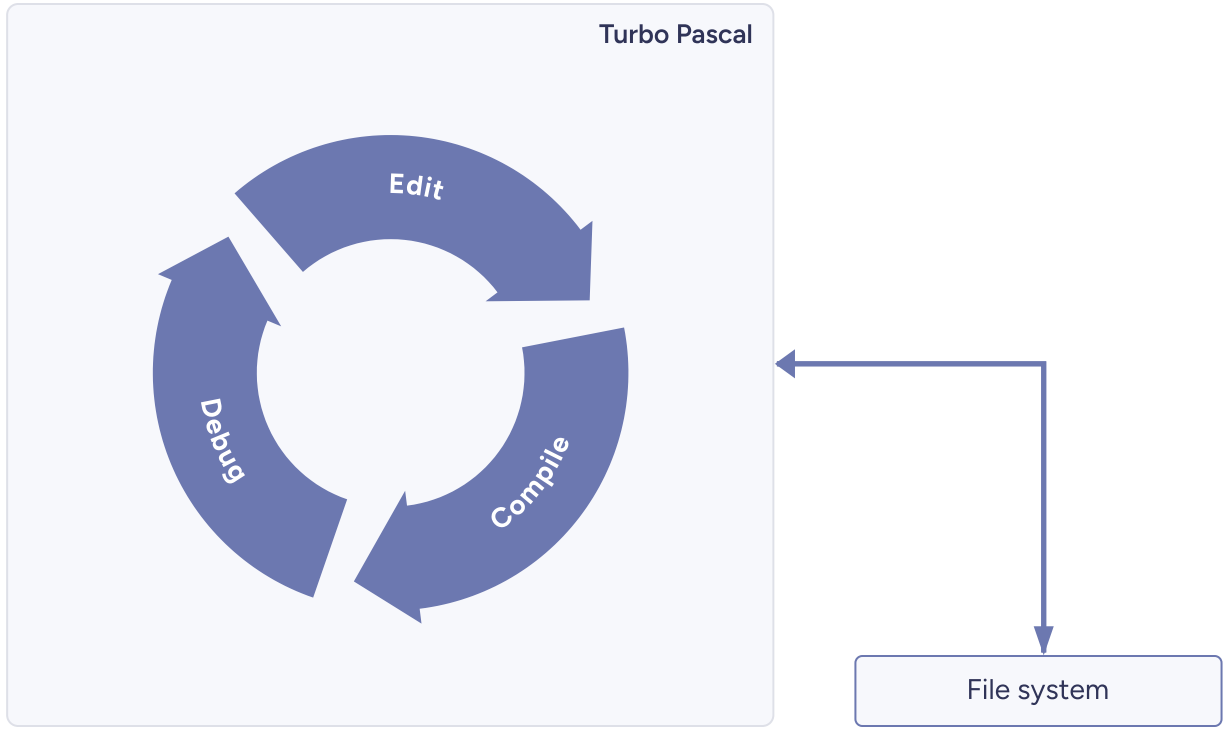
Turbo Pascal (1983)

- Editor
- Compiler
- Debugger
- Code search
- IDE for the masses — speed and price



© 2026 Castle Ridge Software

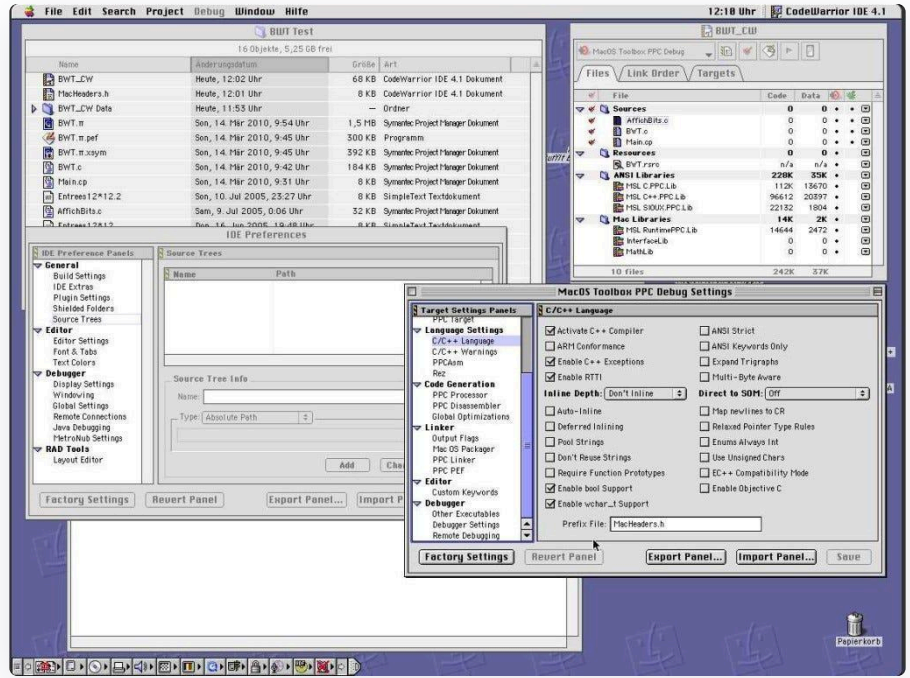
Turbo Pascal — architecture



© 2026 Castle Ridge Software

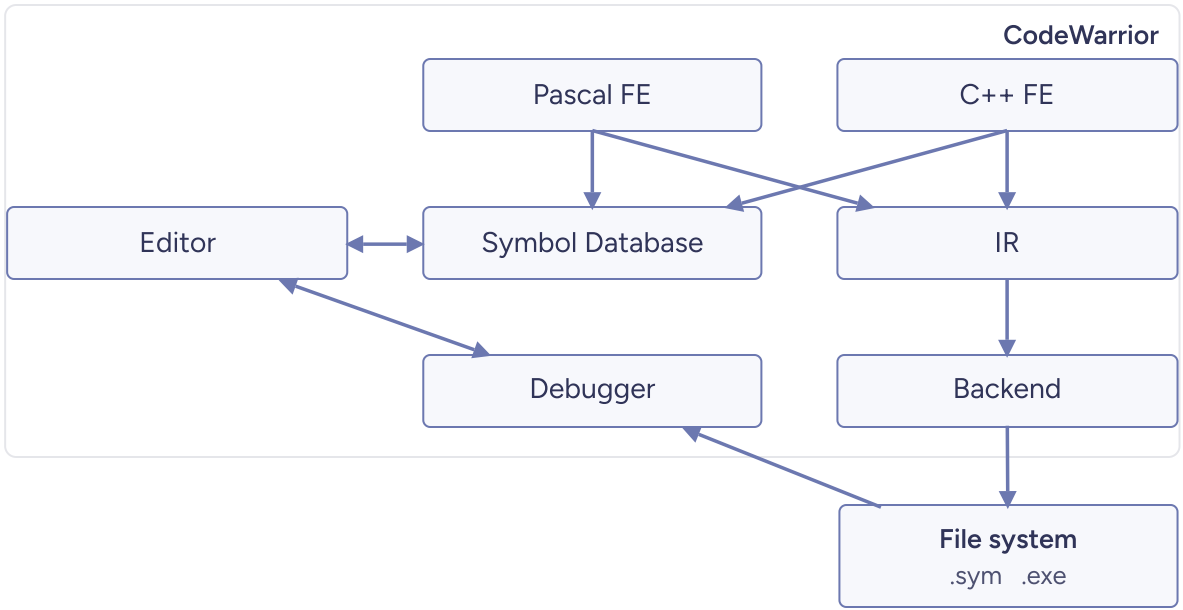
Metrowerks CodeWarrior (1993)

- Multi-language
- Cross-language navigation (Class Browser) and debugging
- Pluggable compiler frontends
- Powerful build system



© 2026 Castle Ridge Software

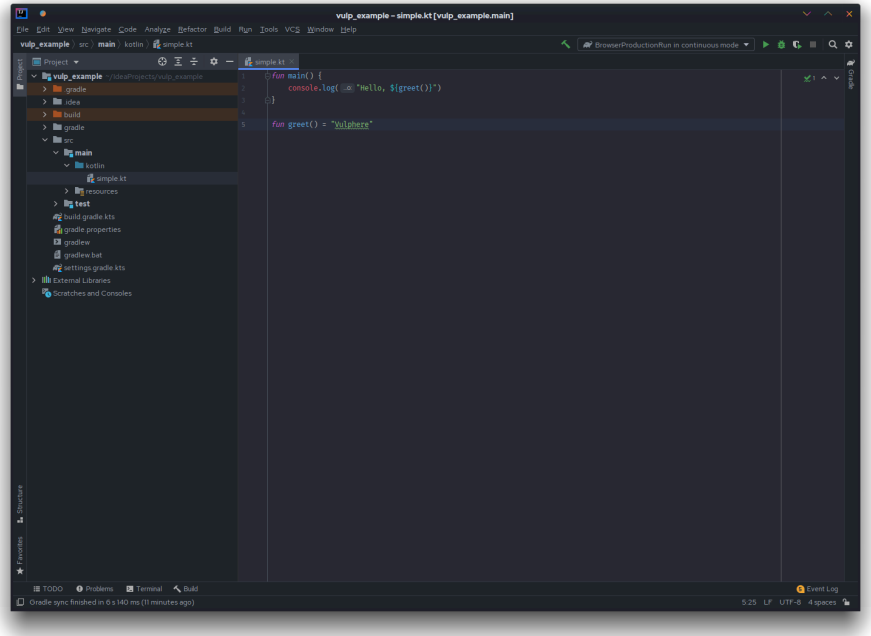
CodeWarrior — architecture



© 2026 Castle Ridge Software

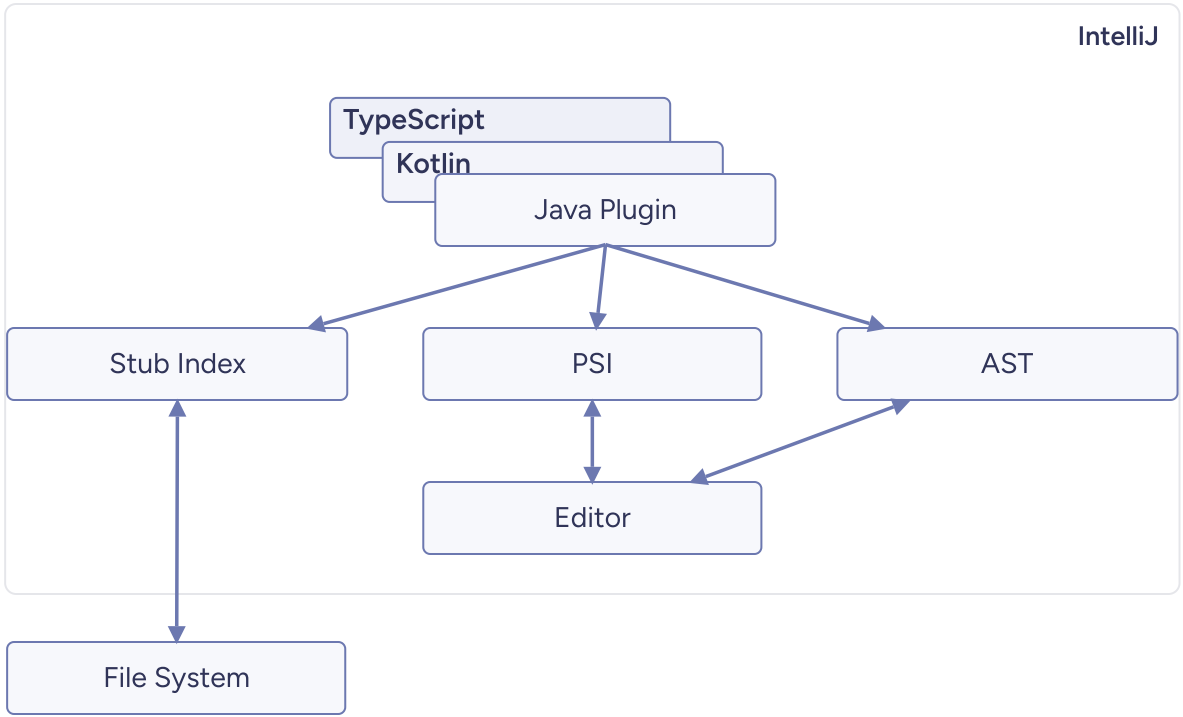
IntelliJ (2001)

- Initially Java only
- Java refactoring
- Generic language integration layer (PSI)
- Language plugins



© 2026 Castle Ridge Software

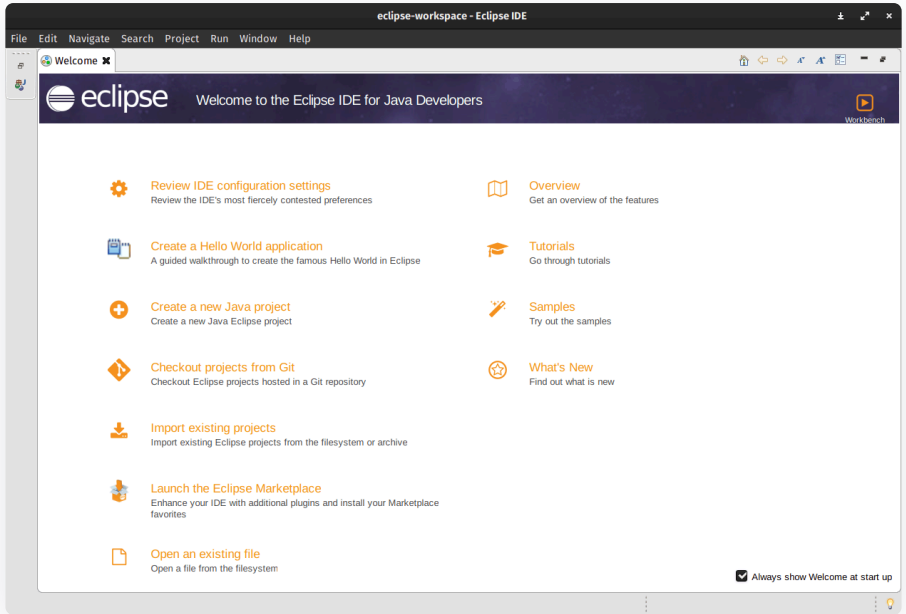
IntelliJ — architecture



© 2026 Castle Ridge Software

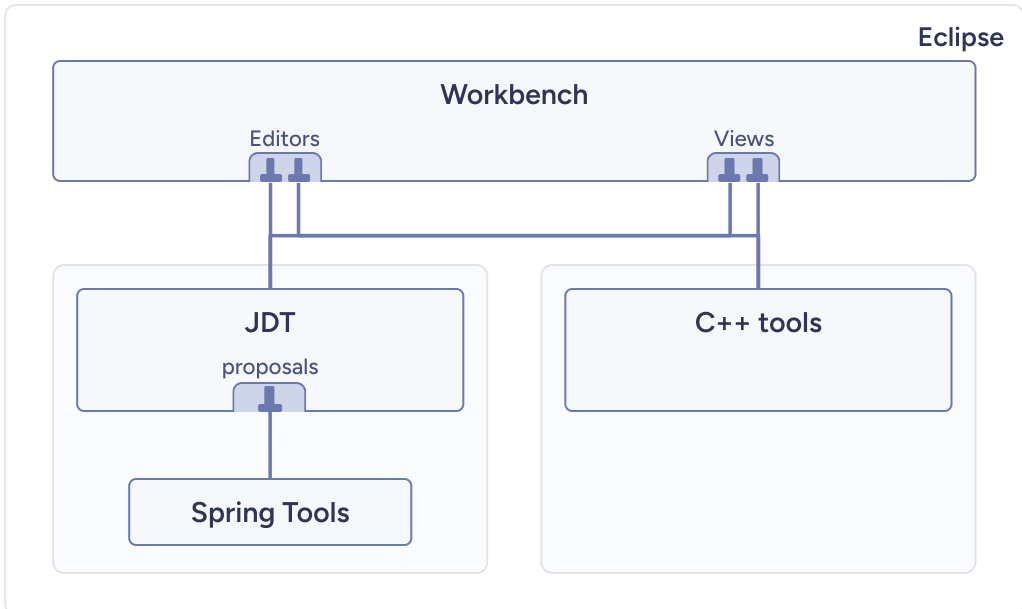
Eclipse (2001)

- Plugins all the way down (no privileged actor)
- “Image” functionality for Java (autobuild, hot code replacement with J9)
- Language features on generic editor / view extensions — not language-semantic extensions
  - No common language infrastructure
- In-process extensions



© 2026 Castle Ridge Software

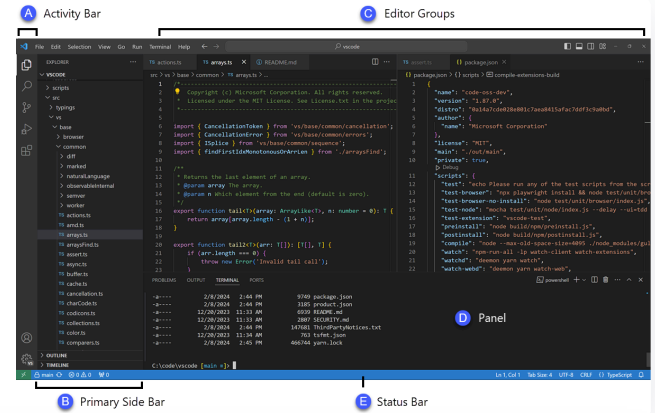
Eclipse — architecture

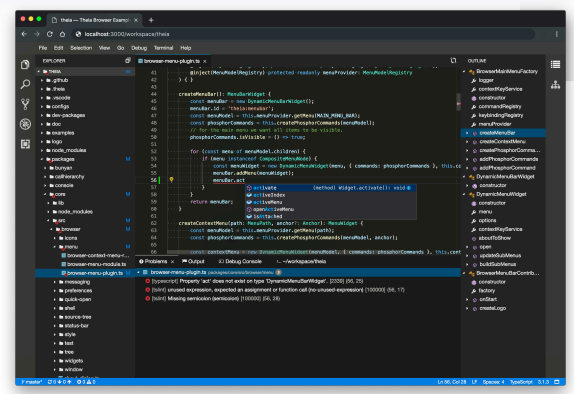


© 2026 Castle Ridge Software

VS Code (2015) / Eclipse Theia

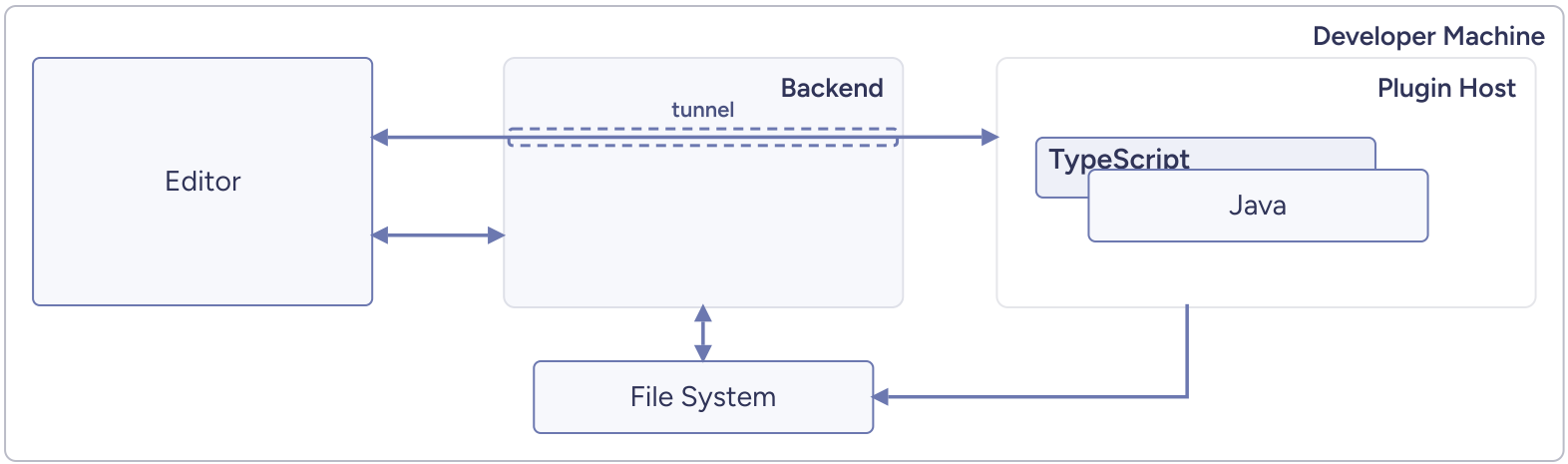
- LSP, DAP
- Out-of-process extensions (coarse-grained) to the IDE UI
- Interaction-based (not semantic)
- Back end can be remote





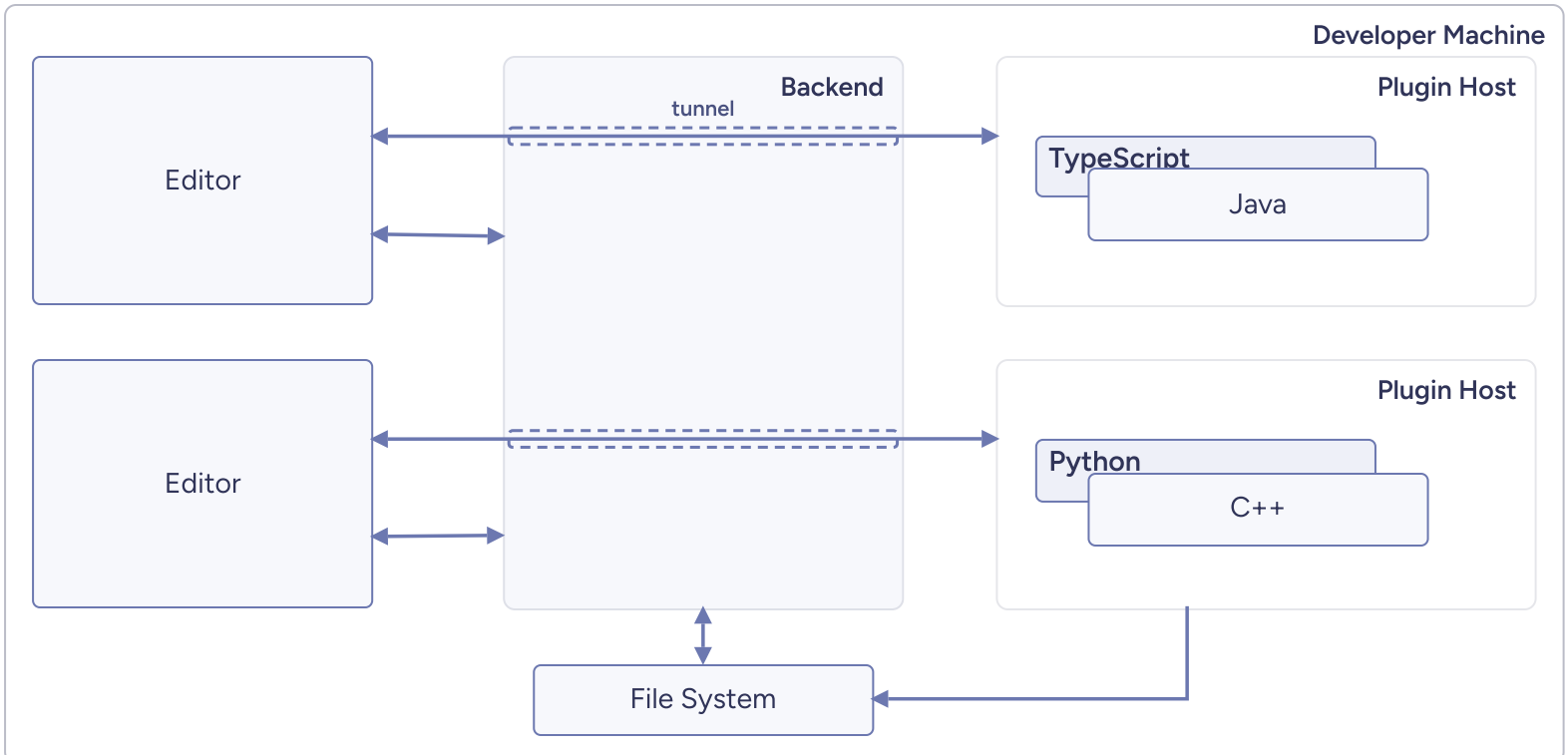
© 2026 Castle Ridge Software

VS Code / Theia — architecture



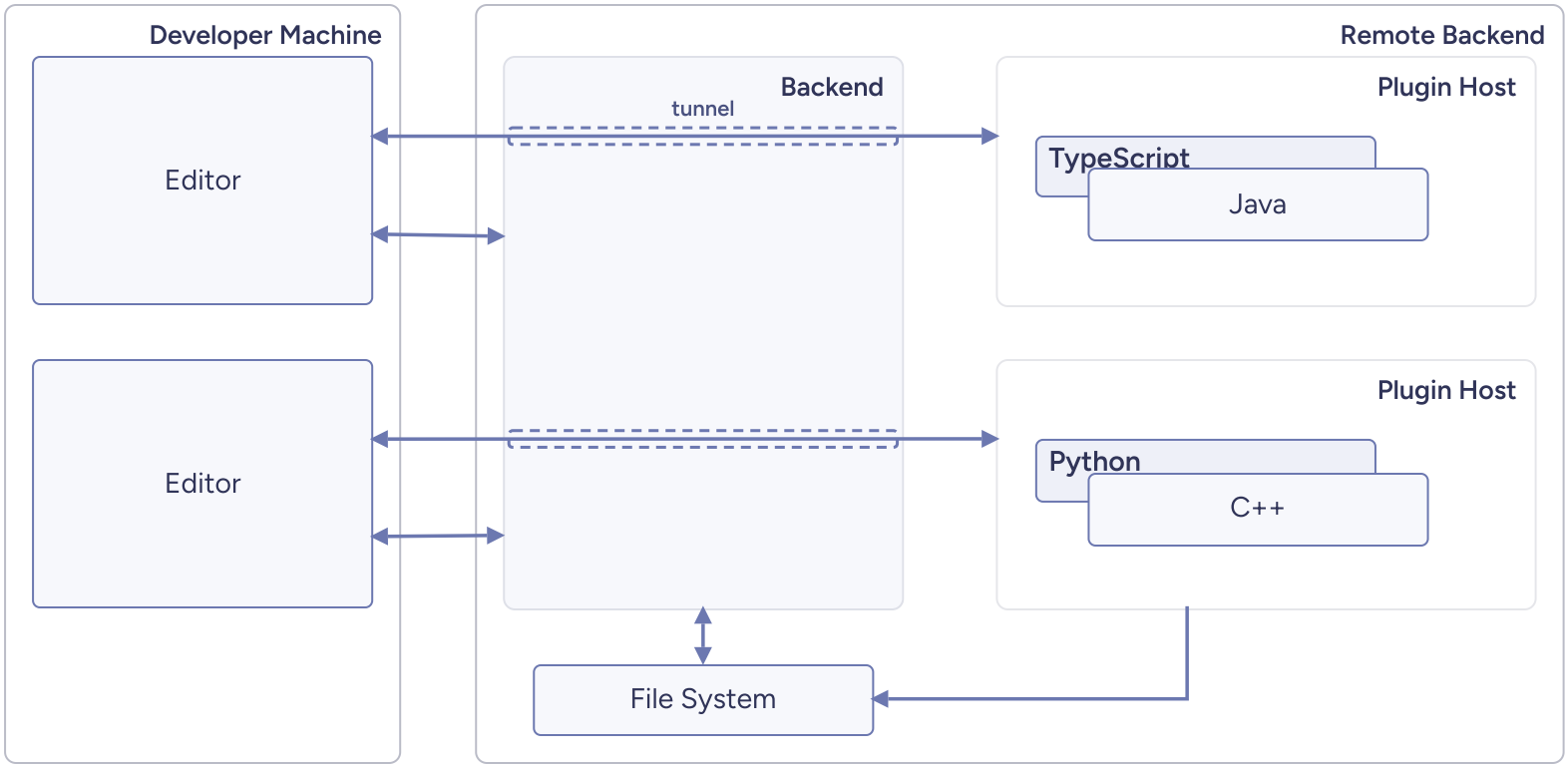
© 2026 Castle Ridge Software

VS Code / Theia — architecture



© 2026 Castle Ridge Software

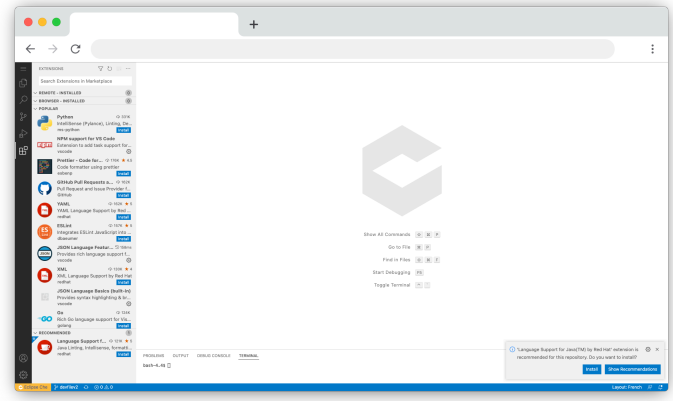
VS Code / Theia — architecture

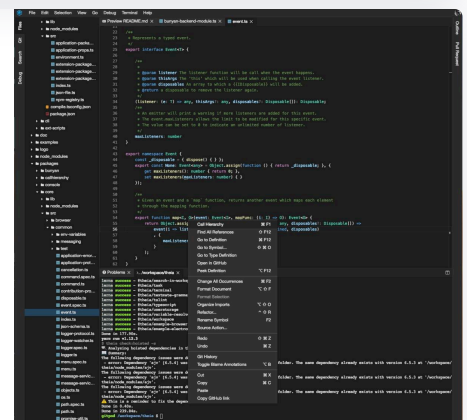


© 2026 Castle Ridge Software

Eclipse Che

- Workspace manager runs a local dev environment in a container / VM
- Containers hold tools, IDE backend, and plugins
- Browser connects as directed by the workspace manager
- "Local development with remote UI"





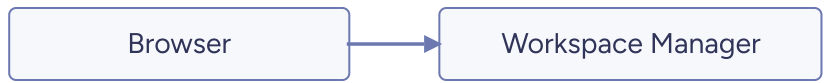
© 2026 Castle Ridge Software

Che — architecture



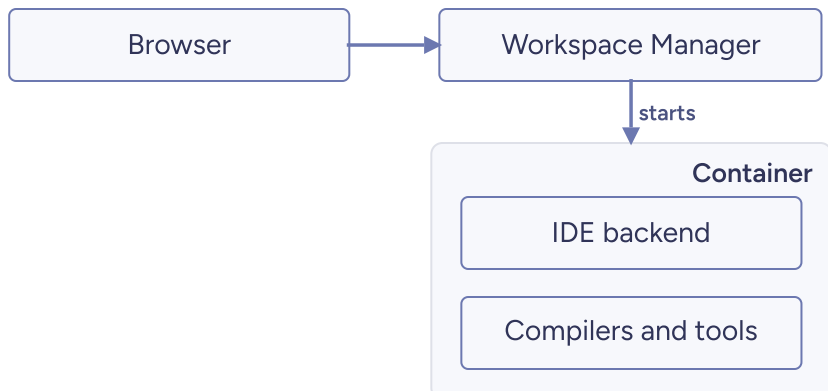
© 2026 Castle Ridge Software

Che — architecture



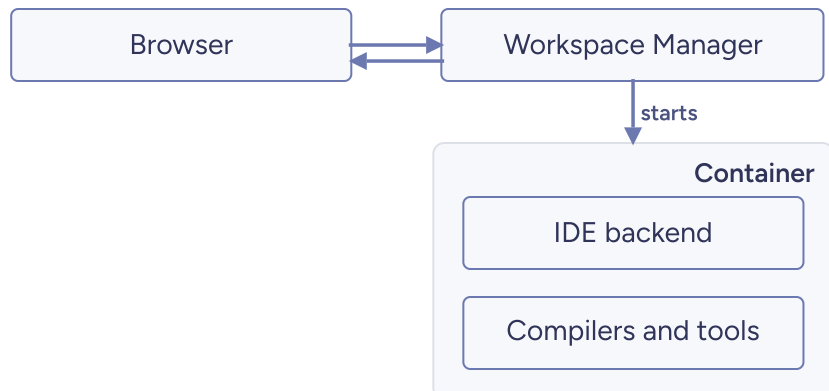
© 2026 Castle Ridge Software

Che — architecture



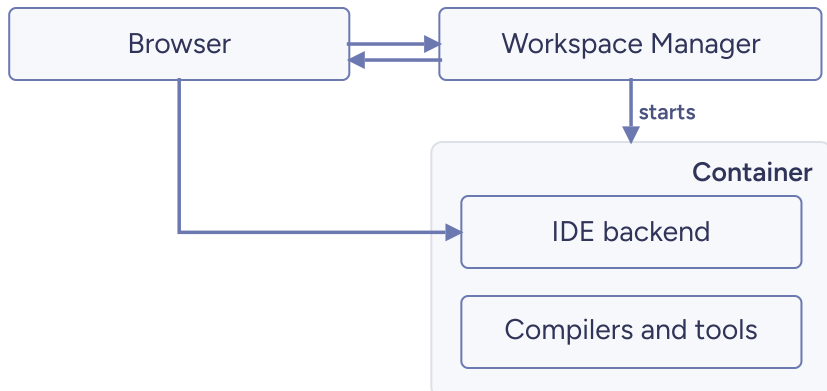
© 2026 Castle Ridge Software

Che — architecture



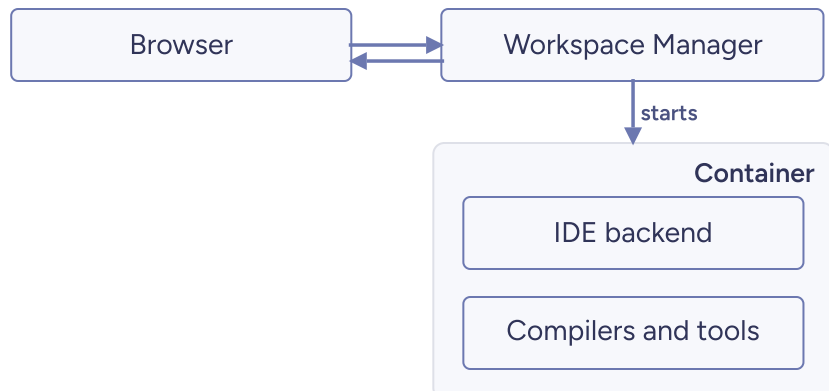
© 2026 Castle Ridge Software

Che — architecture



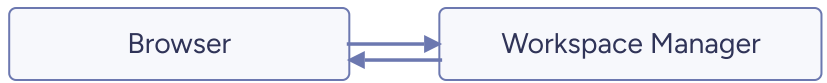
© 2026 Castle Ridge Software

Che — architecture



© 2026 Castle Ridge Software

Che — architecture



© 2026 Castle Ridge Software

So what changes with AI?

© 2026 Castle Ridge Software

Programming Tasks

Humans

- Write
- Understand
- Debug
- Test
- Deploy
- Plan
- Coordinate
- Review

© 2026 Castle Ridge Software

Programming Tasks

Humans

- Write
- Understand
- Debug
- Test
- Deploy
- Plan
- Coordinate
- Review

AI

- Write
- Understand
- Debug
- Test
- Deploy
- Plan
- Coordinate
- Review

© 2026 Castle Ridge Software

Characteristics

Humans

- Input: visual / spatial / auditory
- Working memory: ~5–10 tokens
- Bad at parsing text (reading is high mental load)
- Memory
- Limited high-level knowledge
- Good at abstraction

© 2026 Castle Ridge Software

Characteristics

Humans

- Input: visual / spatial / auditory
- Working memory: ~5–10 tokens
- Bad at parsing text (reading is high mental load)
- Memory
- Limited high-level knowledge
- Good at abstraction

IDEs make programming simpler by reducing cognitive load

© 2026 Castle Ridge Software

Characteristics

Humans

- Input: visual / spatial / auditory
- Working memory: ~5–10 tokens
- Bad at parsing text (reading is high mental load)
- Memory
- Limited high-level knowledge
- Good at abstraction

LLMs

- Input: token sequences
- Working memory: ~200k tokens
- Good at parsing text (it's all they do)
- No memory
- Encyclopedic, detailed knowledge
- Unreliable abstraction

IDEs make programming simpler by reducing cognitive load

© 2026 Castle Ridge Software

Characteristics

Humans

- Input: visual / spatial / auditory
- Working memory: ~5–10 tokens
- Bad at parsing text (reading is high mental load)
- Memory
- Limited high-level knowledge
- Good at abstraction

IDEs make programming simpler by reducing cognitive load

LLMs

- Input: token sequences
- Working memory: ~200k tokens
- Good at parsing text (it's all they do)
- No memory
- Encyclopedic, detailed knowledge
- Unreliable abstraction

Tools for AI should manage the context window

© 2026 Castle Ridge Software

Characteristics

Humans

- Input: visual / spatial / auditory
- Working memory: ~5–10 tokens
- Bad at parsing text (reading is high mental load)
- Memory
- Limited high-level knowledge
- Good at abstraction

IDEs make programming simpler by reducing cognitive load

LLMs

- Input: token sequences
- Working memory: ~200k tokens
- Good at parsing text (it's all they do)
- No memory
- Encyclopedic, detailed knowledge
- Unreliable abstraction

Tools for AI should manage the context window

Tools for AI reduce token usage

© 2026 Castle Ridge Software

LLMs don’t need the “integrated” part

© 2026 Castle Ridge Software

“IDEs are dead”

Replit on X

© 2026 Castle Ridge Software

“IDEs are dead”

Replit on X

IDEs are for humans

© 2026 Castle Ridge Software

Do we still need IDEs?

- Will we still read the code?
- Is LLM programming and abstraction like C vs assembler?
- Is it a leaky abstraction?

© 2026 Castle Ridge Software

Yes, we do

- Decide what to build
- AI needs guidance
  - Design is deciding
  - AI often falls into local minima
- Trust / responsibility
  - Example: NASA Flight Readiness Review
  - Unreliable results from AI

© 2026 Castle Ridge Software

What will we be doing?



© 2026 Castle Ridge Software

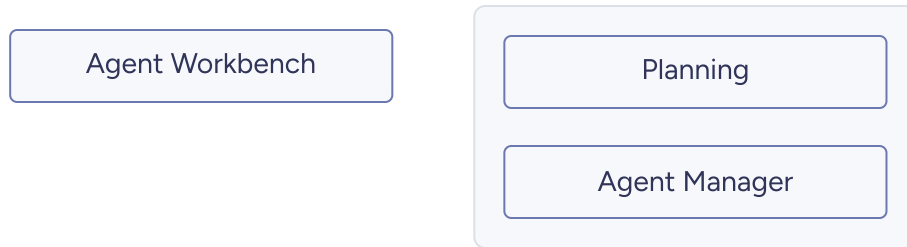
Agent IDE building blocks

- Agent independent of user
  - Lifecycle is different from IDE
  - Agent has its own work environment
- Agent needs IDE services[1]
  - LS
  - Debugger
- The agent needs to be contained
- The user interacts briefly with many agents
  - Startup cost is a blocker

JetBrains study

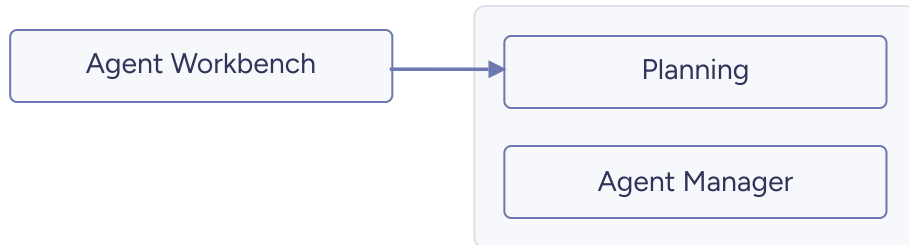
© 2026 Castle Ridge Software

AI architecture



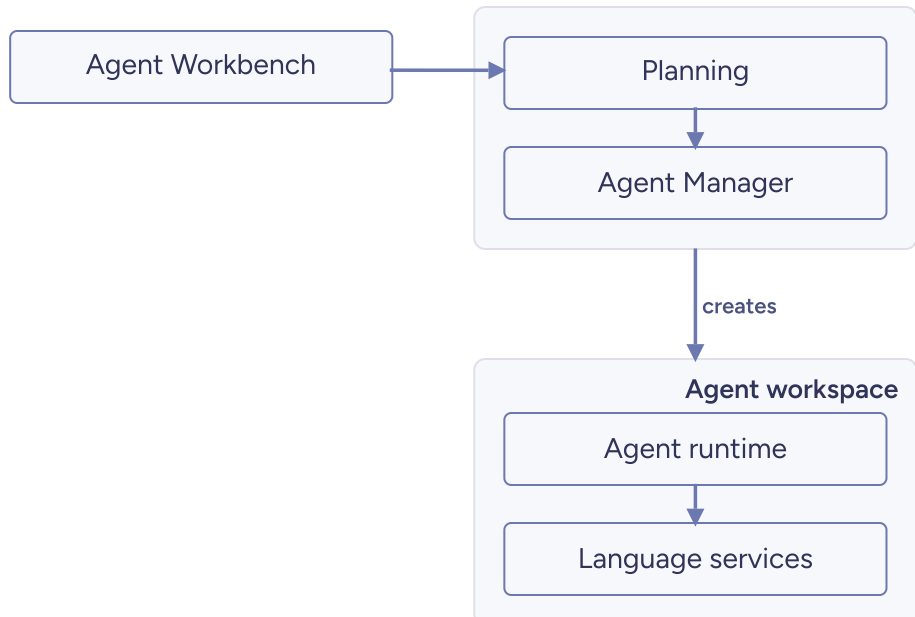
© 2026 Castle Ridge Software

AI architecture



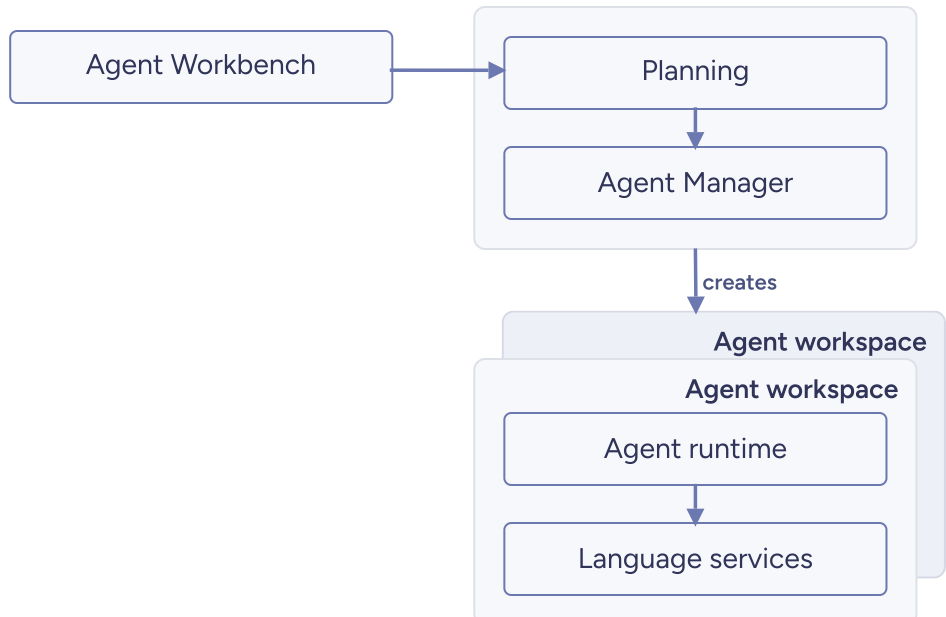
© 2026 Castle Ridge Software

AI architecture



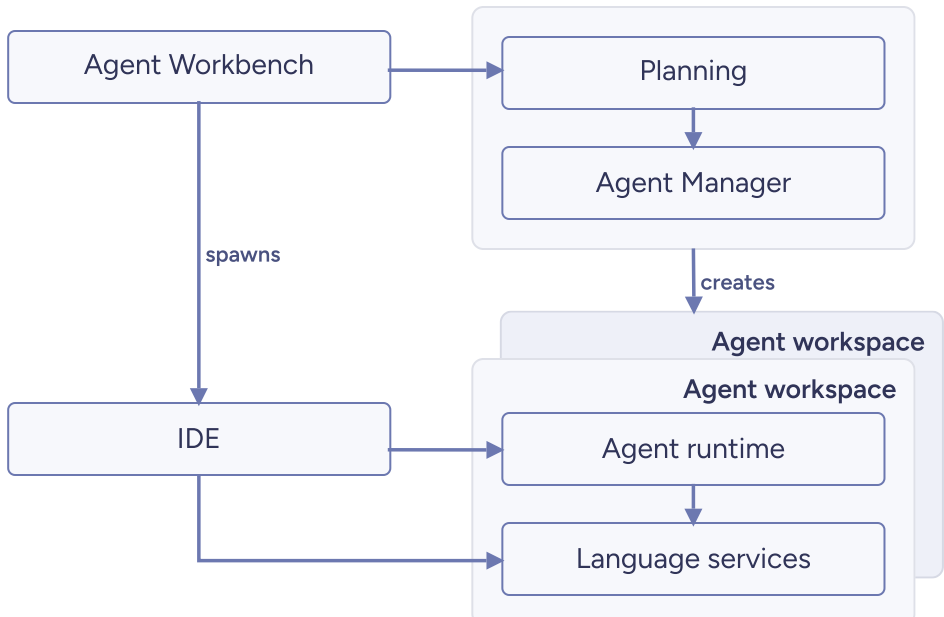
© 2026 Castle Ridge Software

AI architecture



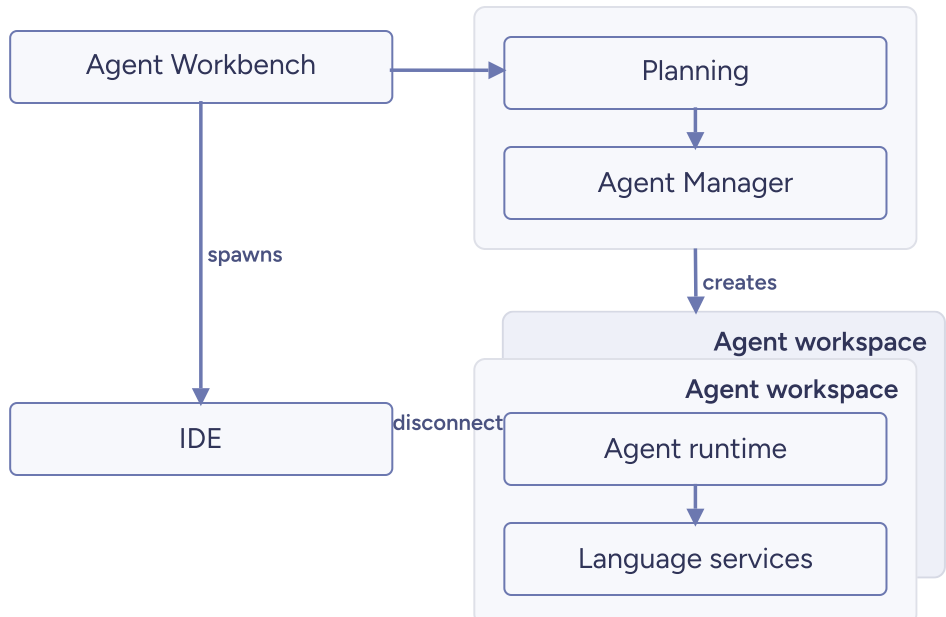
© 2026 Castle Ridge Software

AI architecture



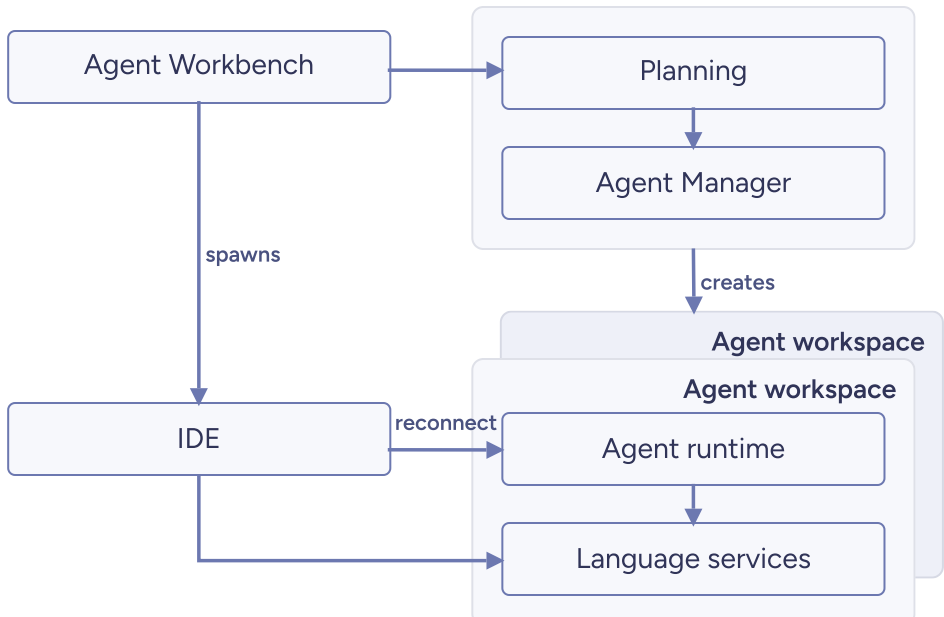
© 2026 Castle Ridge Software

AI architecture



© 2026 Castle Ridge Software

AI architecture



© 2026 Castle Ridge Software

Current architecture disconnect

- Language services plugin into the UI
  - ⇒ Separate headless protocol for language services
  - ⇒ IDE plugins need to be re-layered
- IDE controls the session
  - ⇒ Agent controls lifecycle
  - ⇒ Agent provides configuration
- Protocols (LSP, DAP) are single-client
  - ⇒ Change protocols
  - ⇒ Wrap LSP servers

© 2026 Castle Ridge Software

Why is Theia a good base?

- We have all the parts
  - Theia.cloud
  - Remote IDE support
  - Flexible service architecture
    - backend plugins, preferences
    - API namespaces
    - DI scopes (connect, etc.)
  - Capable and general AI agent framework (Theia AI)
- Vendor neutral
  - Tokens become a commodity
  - No switching costs
  - No rug-pulling
- Framework

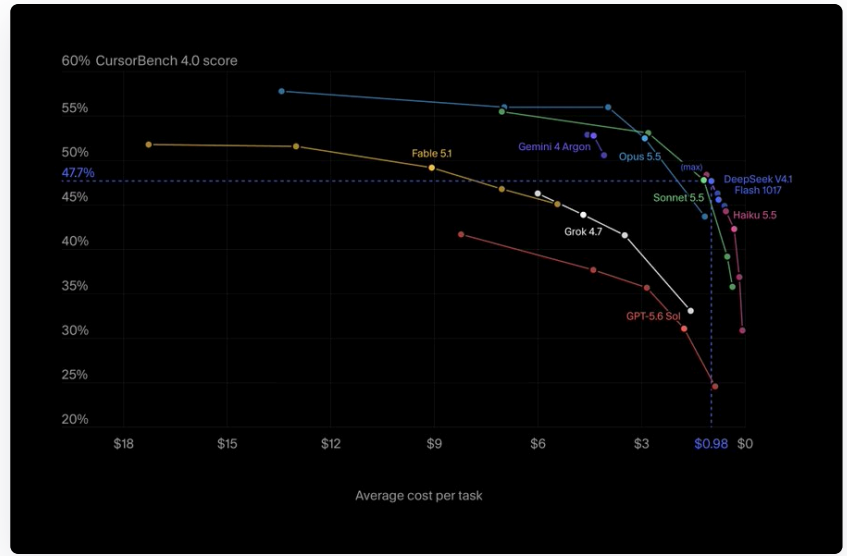
© 2026 Castle Ridge Software

The market view

© 2026 Castle Ridge Software

The market view

- Models are commodity
  - customers are cost sensitive
  - cheap models are almost as good
  - the "best" changes frequently



© 2026 Castle Ridge Software

The market view

- Models are commodity
- The harness is the moat
  - Claude Code
  - OpenAI buys Ona (Gitpod)

© 2026 Castle Ridge Software

The market view

- Models are commodity
- The harness is the moat
- Oligopoly is a nightmare
  - token cost
  - lock-in

© 2026 Castle Ridge Software

We need an open alternative that works for us

© 2026 Castle Ridge Software

We need an open alternative that works for us

Let's "Eclipse" the AI giants

© 2026 Castle Ridge Software



CASTLE RIDGE
SOFTWARE

Let's talk

thomas@castleridge.ch

tsmaeder

tmaeder