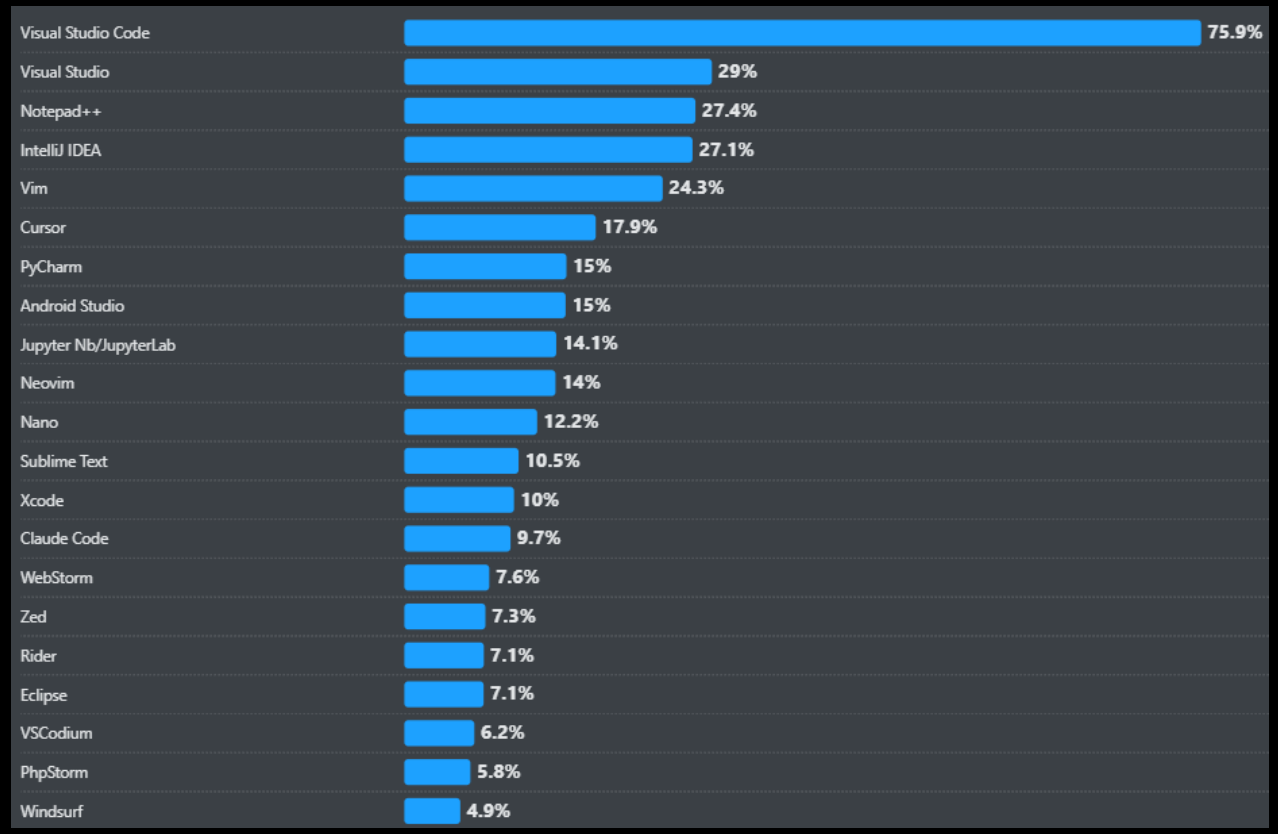
# Side by Side Comparison **Code RealTime** versus **Model RealTime**

The main reasons why you should consider migrating

# IDE Integration

- While Model RT is a plugin for the Eclipse IDE, Code RT is an extension for Visual Studio Code.
  - VS Code is currently the world's most popular IDE (~75% market share)
  - Eclipse is in decline (~7 % market share)
- Code RT also works with other popular IDEs that are VS Code forks (or support VS Code extensions)
  - Cursor
  - Windsurf
  - Eclipse Theia
- Code RT can also be run in a web browser
  - Part of DevOps Loop (from HCL or IBM)




survey.stackoverflow.co

# Language for RealTime Development

- Model RT is based on UML, with realtime specific extensions (UML-RT)
- Code RT uses a domain specific language called Art
  - Significantly smaller and easier to learn than UML
  - Same key concepts as before (capsules, ports, protocols etc)
  - Has no overlap with C++
  - Has both a textual and graphical syntax
  - Embeds C++ code snippets (declarations, expressions and statements)

```
capsule BaseC {
    [[rt::decl]]
    `
        int x = 0;
    `
    behavior port t : Timing;
    statemachine {
        state State1;
        choice c1;
        state X1;
        state X2 {
            entrypoint entryPoint_0;
            state X2_nested;
            else: entryPoint_0 -> X2_nested;
        };
        State1 -> c1 on t.timeout
        `
            Log::out << "Timeout occurred, x = " << x << std::endl;
        `;
```

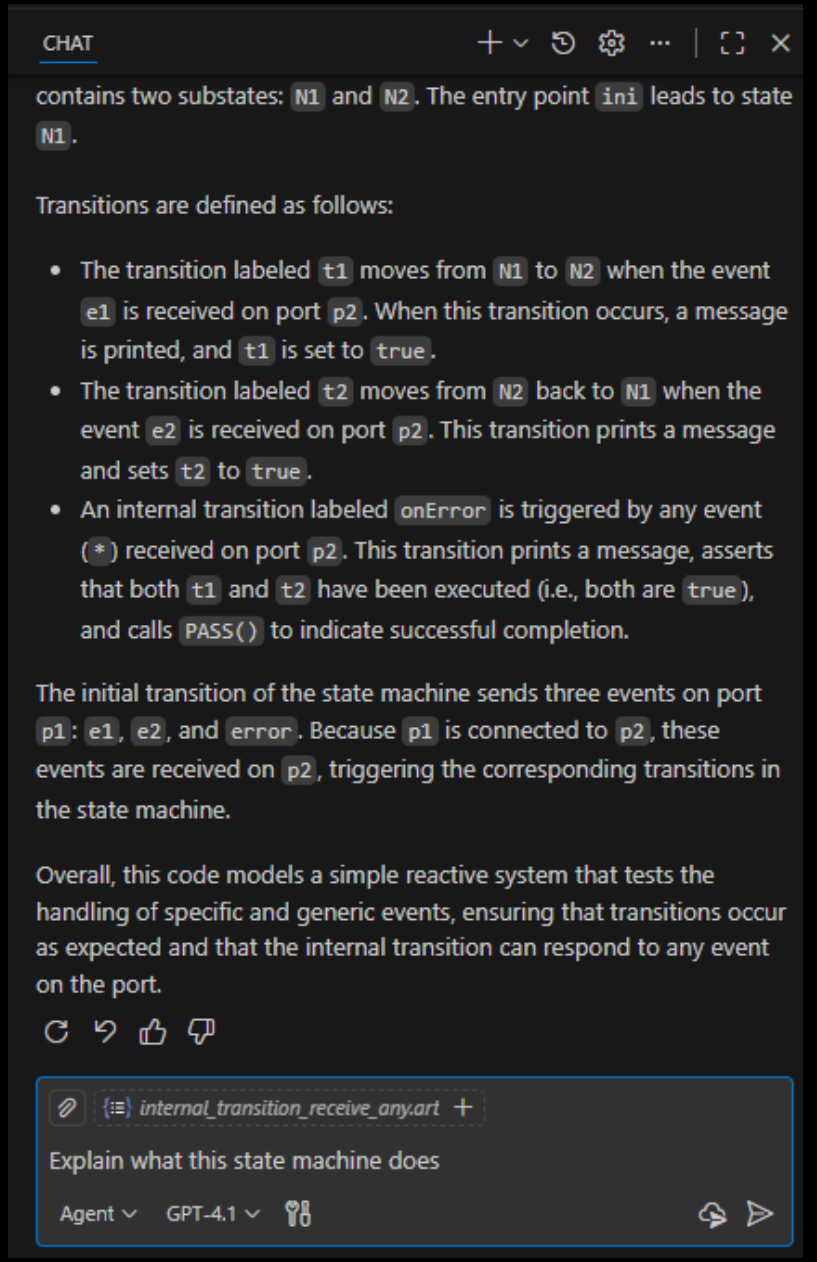
# File Format

- Model RT stores models as XML files, while Code RT directly uses the same Art files as you work with
  - Model RT binds references by unique IDs, while Code RT uses name binding.
    No risk of "model corruption" as can happen if editing Model RT XML files manually.
  - Editing of Art files in Code RT is often more efficient than corresponding model editing in Model RT*:
    - Cut/copy/paste
    - Textual search/replace
    - Multi-line editing
    - AI-assisted editing
- Code RT stores diagram information in JSON files
  - Clearly separated from the program logic in Art and C++ files. In ModelRT diagram information is embedded into the model XML files.

* On average 55% faster editing as measured in the BUMLE research project (ITEA)

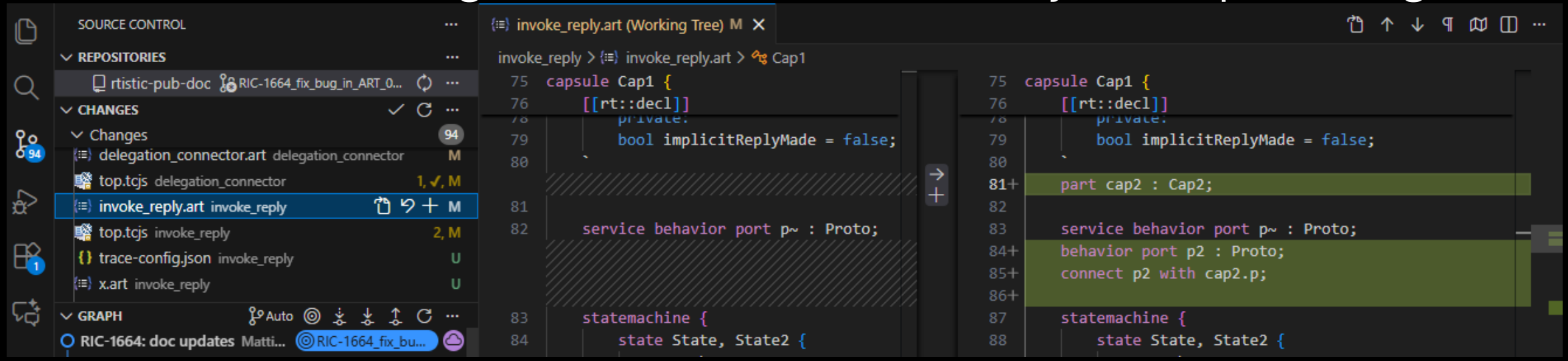
# AI

- In Model RT generative AI works only for C++, but not for UML models and diagrams, while LLMs used in Code RT are useful also for Art files
  - Also works well for C++ code embedded in Art files
- Visual Studio Code (and its forks) gradually evolve into becoming "agentic" IDEs
  - Similar features are lacking for the Eclipse IDE (they may come but with delay)



# Collaborative Development

- Model RT has a special implementation of Compare/Merge, while with Code RT any standard text-based diff and merge tool can be used
  - Same workflow for compare/merge of Art files as with any other source file
  - Model RT Compare/Merge can be complex and require lots of memory when models are big
- GitHub Pull Requests
  - Works well with Art files, but not so well with Model RT's XML files
- JSON files with diagram information are also easy to compare/merge

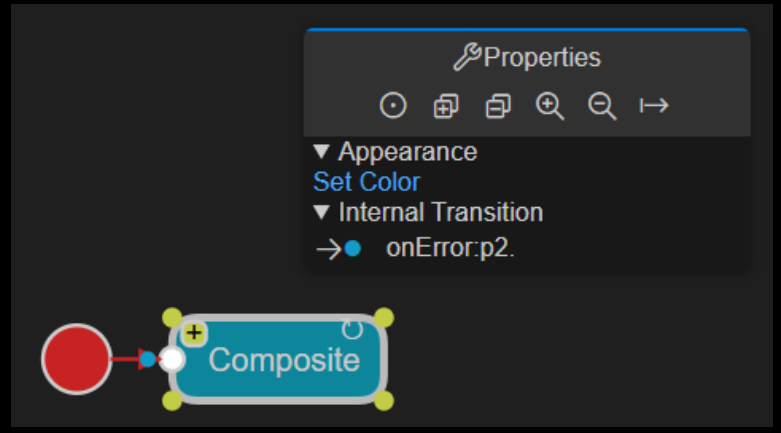


# C++ Support


- Model RT is based on Eclipse CDT while Code RT works with popular C++ language servers
  - Microsoft C/C++
  - Clangd
  - *Note that the C++ language server only impacts on C++ support in the IDE. Generated C++ code can be compiled with any C++ compiler.*
- Code RT allows mixed use of both C++ and Art files as sources
- Model RT has some limitations in the support of modern C++, but with Code RT any C++ can be used
- Code RT provides realtime specific C++ extensions
  - Type descriptor generation (e.g. `[[rt::auto_descriptor]]`)
  - Content assist for common C++ code snippets (e.g. setting a timer)

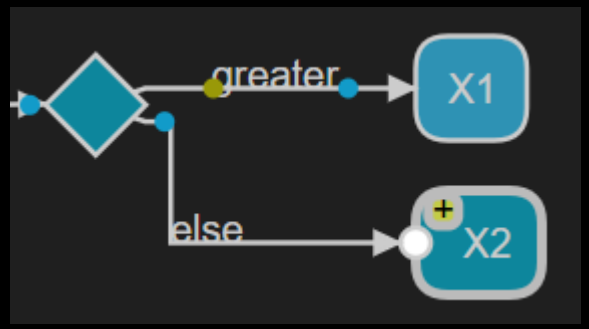
# Diagrams (1/3)

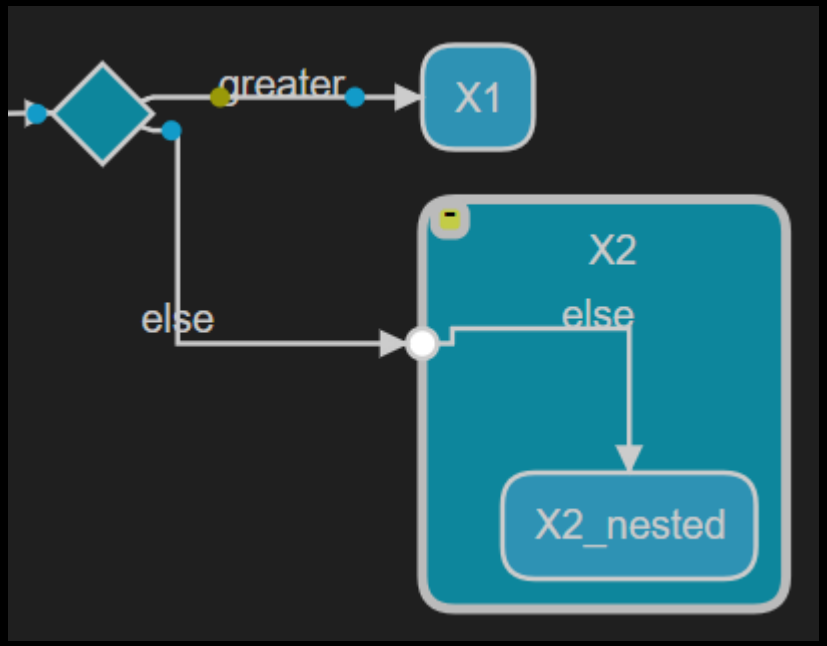
- Model RT requires the use of diagrams for editing models, while for Code RT it's optional to use diagrams
  - You can edit in Art files and use diagrams for graphical visualization
  - But it's also possible to edit in Code RT diagrams. Art files will then update automatically.
- Model RT uses a separate Properties view while Code RT provides a hideable Properties view directly in diagrams
  - Easier to filter what to show, and view or edit additional properties of selected elements
  - For example it shows internal transitions of a selected state



# Diagrams (2/3)

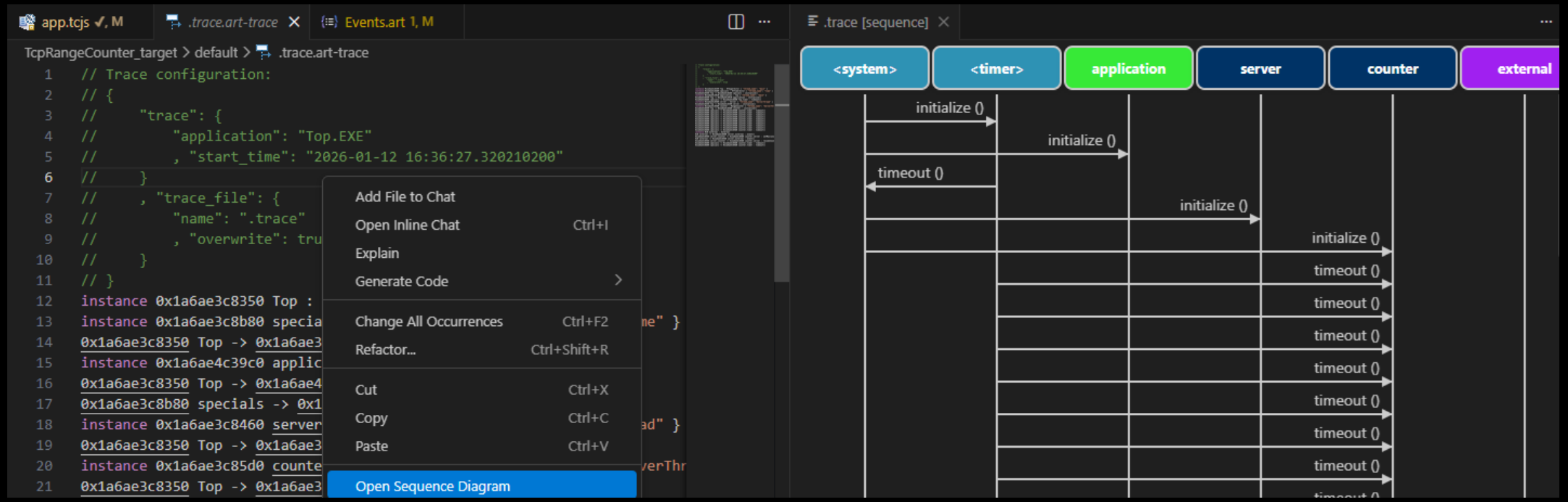
- Model RT requires nested state machines and composite structures to have separate diagrams, while Code RT supports inline expansion
  - This significantly reduces the number of open diagram editors
  - Easier to follow the execution flow through a state machine
  - Adjust the level of details shown in diagrams by expanding/collapsing symbols



# Diagrams (3/3)

- Code RT can visualize traces graphically in sequence diagrams
  - But also supports a textual notation for traces



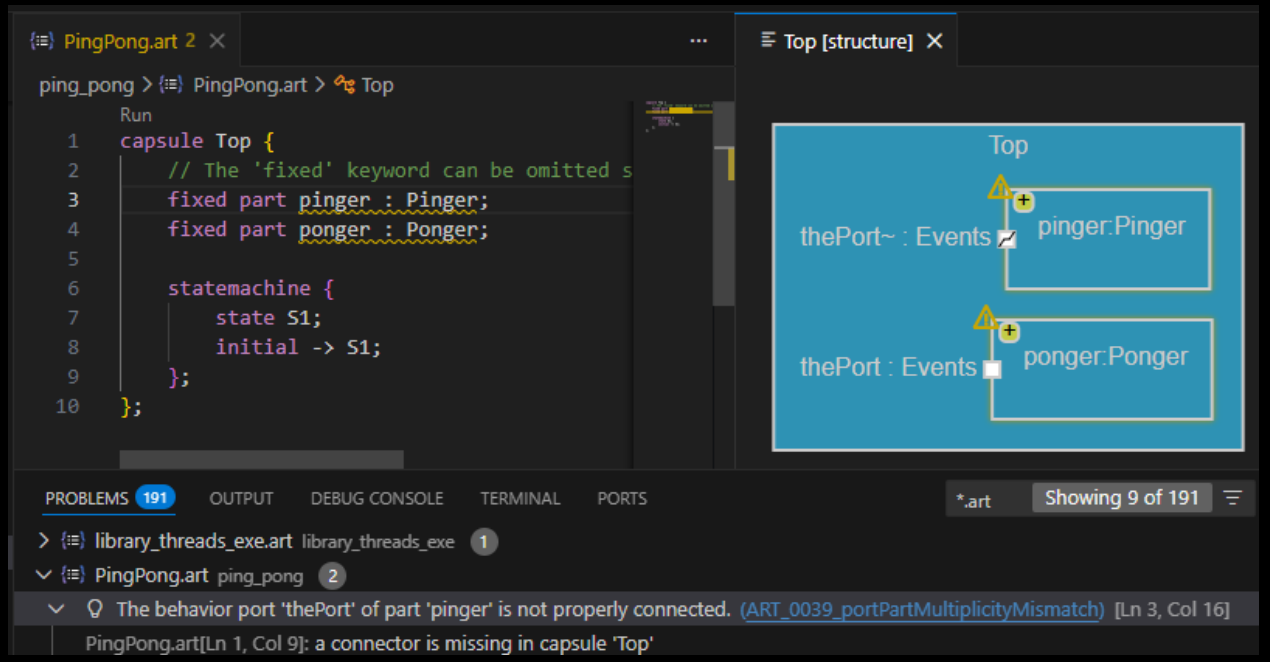
# Code Generation

- Code generation is manually triggered in Model RT, while in Code RT C++ code is automatically generated when an Art file is modified
  - Immediate understanding of how Art elements are translated to C++
  - Seamless navigation back and forth between Art and C++ files, with possibility to edit code snippets in either of them (i.e. "Code-to-Art Synch" is available)
- Compatible but optimized code
  - Code generated by Code RT is compatible with code generated by Model RT, and the same TargetRTS library is used
  - Some optimizations for reduced memory consumption and increased readability
- Command-line builds with the Art Compiler
  - Similar to Model RT's Model Compiler, but runs faster and uses less memory*

* Art Compiler has 35% of execution time and 73% of used memory compared with Model Compiler (measured for the same customer model)

# Validation

- Model RT only has a few basic validation rules for checking the correctness of a model
  - They have to be manually triggered
- Code RT provides ~70 validation rules to detect many types of problems
  - They run automatically as you type
- In Code RT problem severities are customizable, and validation rules can be turned off (globally or locally within an Art element)
  - Model RT only supports this for a small number of problem types
- Code RT provides Quick Fix actions for many problems
  - Automatically fix common problems



# Editions and Licenses

- Code RT comes in 3 editions:
  - Commercial Edition - HCL
  - Commercial Edition - IBM
  - Community Edition (free to use for non-commercial use)
- The Community Edition helps adoption in academia and non-profits
  - Open source documentation, test suites, samples at GitHub (with forum)
  - Only includes precompiled TargetRTS libraries (no source code)
- The Commercial Editions use the same license entitlement as Model RT
  - No extra cost to use Code RT for users of Model RT

# Migration of Models to Code RT

- An Art Exporter plugin can be installed on top of Model RT
  - Automates the translation of TCs and all their source model elements into Code RT Art, C++ and TC files
  - Diagram layout information is kept (but currently not all styling)
  - UML constructs beyond the scope of Art is translated directly to C++ files
  - *Note: Since Code RT has more validation rules, it may sometimes detect problems that went unnoticed in Model RT. Such problems can either be fixed in Model RT before migration, or suppressed in Code RT if not severe.*
- Models in RoseRT can also be migrated to Code RT, but first have to be migrated to Model RT
- Professional services, and the Code RT development team, are available to help with migrations
  - Both from Rose RT and Model RT

# Summary

- Code RT runs in modern, popular IDEs (VS Code and the likes)
- Textual Art language with graphical diagrams
- No restrictions on what C++ can be used
- Works great with generative AI and agentic IDE features
- Collaborative development works with standard tools
- Usability improvements in diagrams
- Same license entitlement as for Model RT - no additional cost
- Automated migration of Model RT models