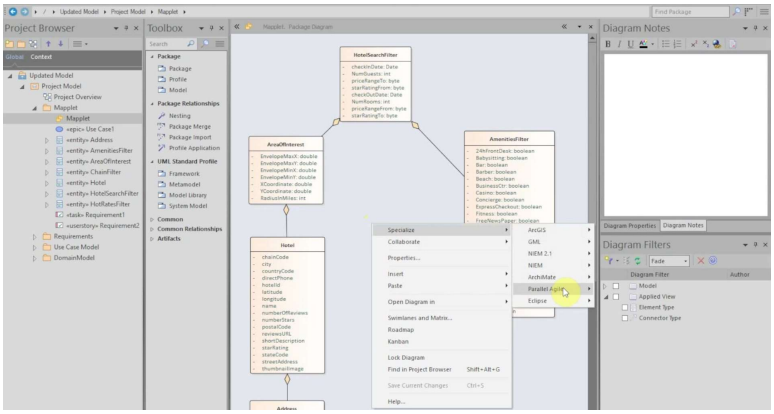
# Warp Speed to Project Completion

## *Introducing the Parallel Agile Add-in for Enterprise Architect*

**Doug Rosenberg**

Parallel Agile, Inc.
www.parallelagile.com
doug@parallelagile.com, training@parallelagile.com

This article will introduce you to both the Parallel Agile (PA) process and to the Parallel Agile Add-in for Enterprise Architect, which enables the PA process for Sparx customers.

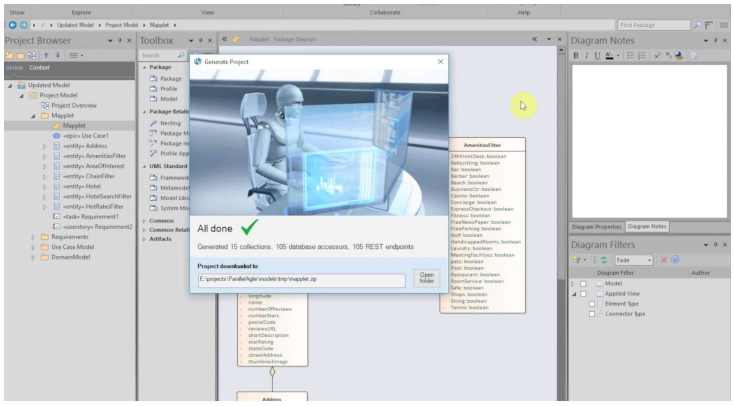



We'll discuss:

- What's Parallel Agile?
- Compressing schedules with parallel development
- Improving quality while compressing schedule
- Why did we build an Enterprise Architect Add-in?
- What's an Executable Architecture?
- What's a Parallel Agile CodeBot?
- Accessing and Setting up CodeBot
- Using CodeBot to create a domain-driven executable architecture
- Using our Cloud-Based Hosting Service to Test Your Generated API
- Use Case Complexity Analyzer
- Parallel Agile MDG Technology – supports Sprint Plans

The Parallel Agile Add-in generates database access code and REST APIs from domain models, and it works in conjunction with the ICONIX Agile DDT Add-in, which generates acceptance tests from requirements and use cases. Both add-ins are free.

Parallel Agile has its roots in ICONIX Process (which shouldn't be a surprise given that I'm the author) and rolls in some other time-tested ideas like Domain Driven Design and the Incremental Commitment Spiral Model (ICSM). ICONIX Process addresses analysis, design, and testing, but doesn't go deeply into implementation. By contrast, PA gets to code, and it gets to code very quickly, right at the inception of your project.



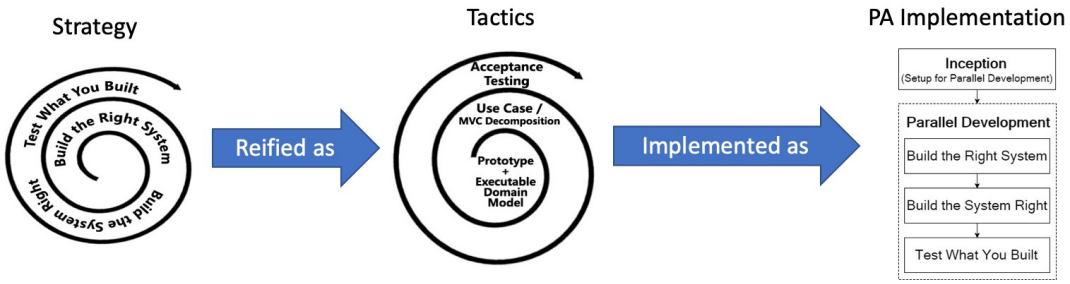

We've been working on developing, piloting, and refining the Parallel Agile process for the last four years, testing out various techniques for large-scale parallel software engineering with over 200 of Prof. Boehm's graduate students from the University of Southern California (USC) Center for Software and Systems Engineering (CSSE).

We're writing a book, ***Parallel Agile: Faster Delivery, Fewer Defects, Lower Cost***, that will be released in a few months. And, after four years of trying it successfully on multiple projects, we're excited to make it available to the Sparx Community. So here we go…

## What's Parallel Agile?

Parallel Agile is a development process focused on large scale parallel development, with the intent of simultaneously shortening schedules and improving quality. The saying goes *"Quality is King, but Schedule is God"*, and wouldn't it be nice to get higher quality and shorter schedules? Well, of course it would… but the question is how can we do that?

We've been having really good success by partitioning projects up into their use cases, and assigning a developer to each use case, then letting them work in parallel – and we've scaled this to a pretty decent number of developers. This sounds simpler than it is, because historically, integrating the work of large numbers of parallel developers has been problematic. We've gotten around this integration problem by producing something that we call an Executable Architecture by code-generating a NoSQL database and REST API from a domain model right at the inception of the project – and that's where the CodeBot comes in. We'll talk more about Executable Architectures and CodeBots shortly.



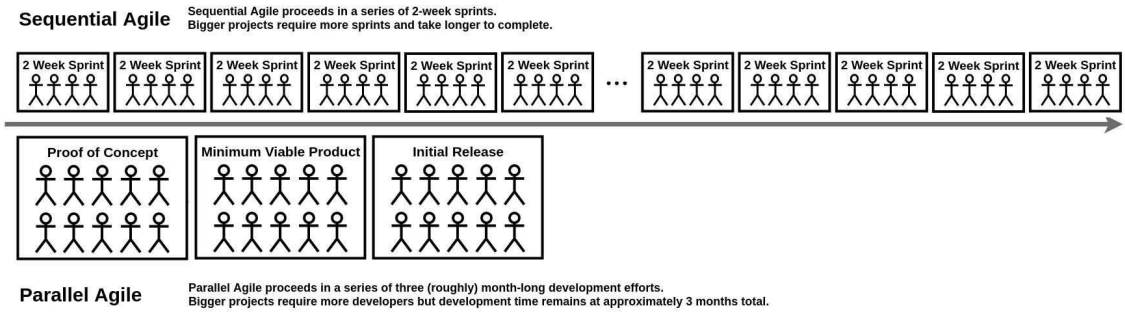

Parallel Agile adopts a 3-phase approach to development, starting with prototyping against a live database, then moving on to careful UML design and comprehensive acceptance testing. Since each developer is working on a small use case and all the developers are working in parallel, each phase can usually complete in around a month of calendar time.

## Compressing schedules with parallel development

Everybody would like to get their software projects done sooner. And there are many strategies that attempt to attack this problem from various angles. Parallel Agile attacks it from two directions:

- ***first, we use automation*** in the form of code generation and test case generation
- ***second, we leverage parallelism*** extensively – lots of developers working together can get things done in a hurry

Parallelism in general is a great way to get things done fast. High performance computing strategies typically leverage parallel processing to reduce compute time. Yet parallelism in software engineering is not typically employed to shorten schedules in a similar fashion. One reason for this is Brooks' Law, which states that it's impossible to accelerate a software project in proportion to the number of developers on the team.




You can think of Parallel Agile as an attempt to find a loophole in Brooks' Law. This pesky "law" is generally regarded as an absolute truth, which leads to agile development methods with small teams of developers proceeding incrementally in a never-ending series of short sprints, often leaving mountains of technical debt in their wake.

Brooks' Law stands in the way of large-scale parallel software engineering like Einstein's speed-of-light speed limit held back intergalactic travel until the invention of the warp drive, which (according to Wikipedia[1]) was invented by Zefram Cochrane in 2063. But Brooks' Law was postulated in the last millennium, before the advent of REST APIs, NoSQL databases, Domain Driven Design, Executable UML, and most importantly – CodeBots.

## Improving quality while compressing schedule

Getting to market faster (the big promise of agile methods) provides a lot less value to your business if the product you get to market with is under-specified, under-designed, and haphazardly acceptance tested. The final third of a Parallel Agile project is devoted to rigorous acceptance testing, focusing on *auto-generated test scripts* that exercise all of the corner cases, edge cases, and rainy-day scenarios – the real troublemakers on most software projects.

Comprehensive acceptance testing can take a lot of time, unless

- *each developer (working in parallel) is responsible for detailing out the sunny/rainy day behavior spec for their use case in the form of an EA "structured scenario"*
- *the structured scenarios are then used to automatically generate acceptance test scripts that can be run by an independent QA department*

These strategies are defined in the **Design Driven Testing** book that I wrote with Matt Stephens, and the automation exists within Enterprise Architect in the form of the free **ICONIX Agile DDT Add-in**. The add-in also generates a skeleton test case for each requirement and a few other things. Using these techniques within the PA process results in higher quality software being delivered faster than you might expect.

## Why did we build an Enterprise Architect Add-in?

I've been writing about, and teaching Enterprise Architect since EA version 3, a long long time ago. In my estimation EA has been (and remains) the best modeling tool on the market since I started using it. Over the years EA's industry-leading price/performance ratio has resulted in a market-leading position, so Sparx is the obvious choice. Beyond that, EA is the only modeling tool that offers automated support for Design Driven Testing, and DDT is a fundamental part of Parallel Agile (which makes the choice doubly obvious).

In addition, Sparx's aggressive support for agile concepts like backlog management and collaborative development works nicely with Parallel Agile's visual model sprint plans. Advanced capabilities like behavioral code generation from activity and state diagrams mesh well with Parallel Agile's "*generate code where you can*" strategy for shortening schedule and improving quality (generated code tends to not be very buggy).

---

[1] https://en.wikipedia.org/wiki/Zefram_Cochrane

So the bottom line is that we'd have been crazy not to develop an EA add-in. And now we've got one that we hope you'll like.

## What's an Executable Architecture?

Executable Architectures are a key enabler of massively parallel development and elastic staffing. Executable Architectures simplify collaboration among a large group of developers, because *everybody's code plugs into the architecture*.

If you follow the Parallel Agile process, you'll create a Domain Driven Architecture by turning your UML Domain Model into executable MongoDB database access code, and NodeJS code. You can do this without a CodeBot but it's far easier (and cheaper) with one. Manual coding of common database access functions is tedious, slow, error-prone, and expensive, so this saves you time on your project. It also enables large scale parallel development – **if your developers understand the domain model, they also understand the API, and a shared understanding of the API simplifies collaboration greatly**.

Domain Driven Design (DDD) is a popular design approach that has been proven successful over many years. A common, shared understanding of the problem domain is a critical success factor on most software projects. Without a shared understanding of the problem, solutions become chaotic. Domain Driven Architectures are the most resilient to change because the problem domain generally changes much more slowly than the software requirements. Executable Domain Driven Architectures let lots of developers work in parallel and integrate their work by calling the API for whatever domain objects they need.

Availability of REST APIs for common database access functions make it easy to develop for multiple client platforms, and combined with the flexible schema of NoSQL databases, make it possible to prototype extensively against a live database. Prototyping against a live database early in the project enables feedback-driven database design.

## What's a Parallel Agile CodeBot?

Parallel Agile's CodeBot is a cloud-based code generator that turns your UML diagrams into executable code. More specifically, it transforms a domain model (EA class diagram and package) into a set of NoSQL (MongoDB) collections, generates the basic set of database access functions (Create, Read, Update, Delete, and schema validation code), and wraps those accessors in a REST API (Node Express).

Generating an executable architecture from the domain model means that an API is created right at the very beginning of your project. This "instant API" turns out to be a key enabler of parallel development, starting with developers *prototyping against a live database* to discover requirements. Putting prototyping upfront and following it with rigorous use case design and

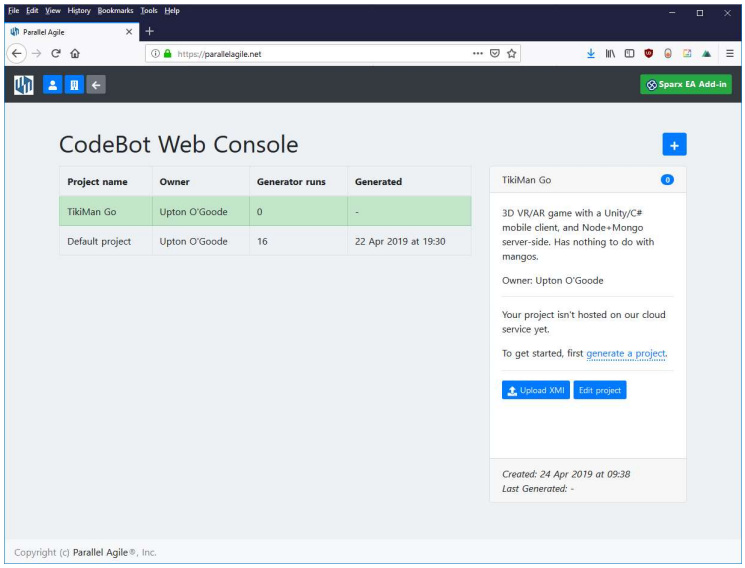
acceptance testing has a “space-warping” effect on your project schedule – your destination reaches you a lot faster.

You can see a video demo of the CodeBot in action at **http://tinyurl.com/CodeBot1**


## Accessing and Setting up CodeBot

To get started, register for a free 1-month account at https://parallelagile.net (in case you’re wondering, **parallelagile.com** is our main website, while **parallelagile.net** is the CodeBot web console where you can login while using CodeBot on your project, managing your free, secure cloud-hosted API etc).

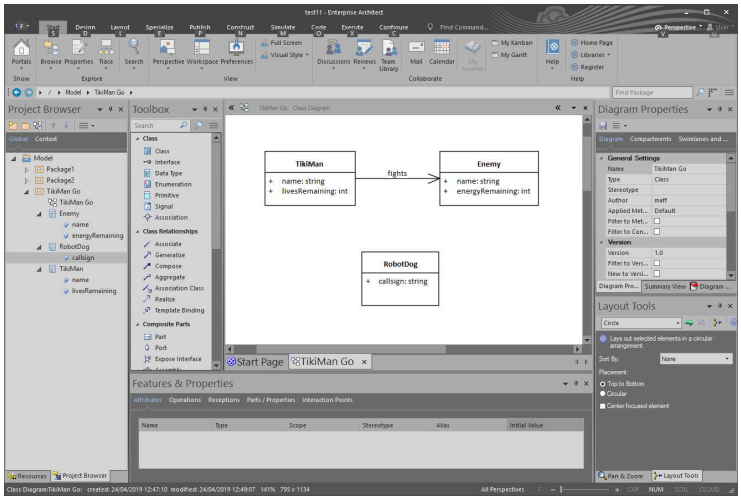
Once you’ve logged into the web console, download and install the add-in from the link at the top right of the page:



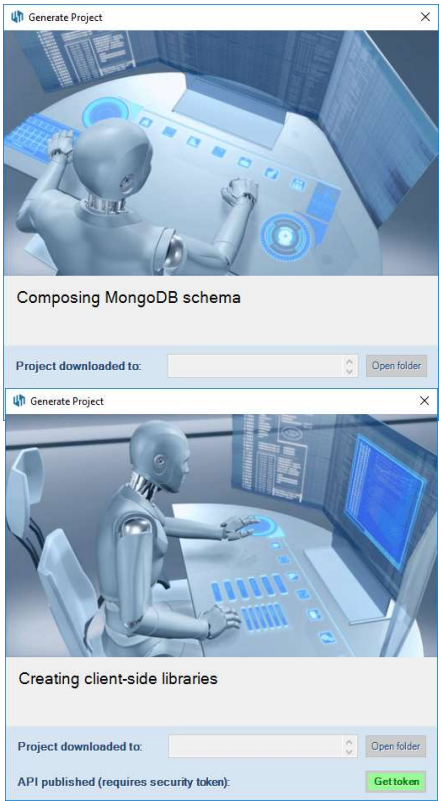
We start you off with a default project so that you can run CodeBot with a minimal amount of setting up; however you can also create additional projects with customized details if you need to. In the above screenshot, note that the API isn't yet hosted. This is of course because CodeBot hasn't been run yet via the add-in; so let's do that next...

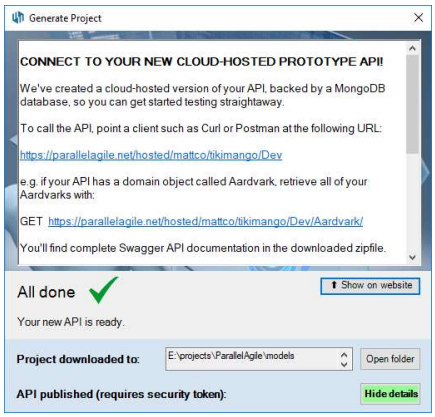
## Using CodeBot to generate a domain-driven executable architecture

To demonstrate, I've created a very simple example domain model with just 3 domain classes:




Right-click on your domain model package or in the diagram, and choose *Specialize… Parallel Agile… Generate Project*. The add-in will export your domain model and run it through our cloud-based CodeBot service:



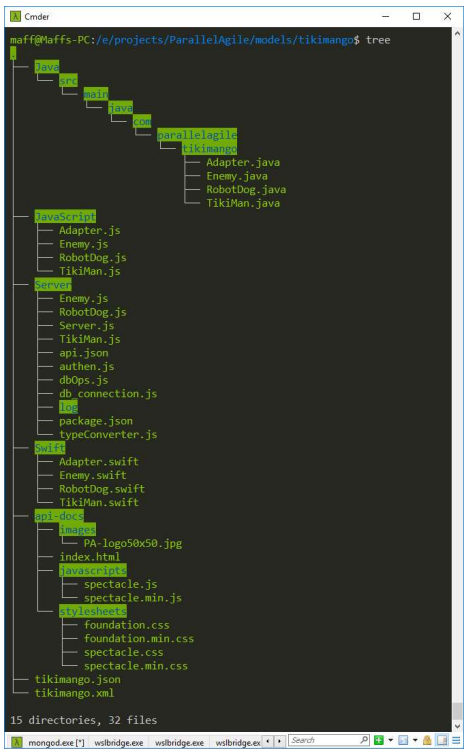

After a few seconds, you'll have downloaded a zipfile full of Node Express code including a MongoDB schema represented in the code, and Swagger API docs. CodeBot also creates client-side code in JavaScript and Swift, with a type-safe Java client following on soon. (Note that the Free License doesn't include the server-side Express code; however you can still use the new REST API using the generated client library, or Postman, Curl etc, via the hosted API).

The REST service that's generated can be deployed as a normal Express server application; however by integrating with the serverless-http library, it also works "out of the box" as a serverless NodeJS-based AWS Lambda function. We're also looking at adding more target platforms in the future, as well as supporting more databases and client-side languages.

If there's a particular language, platform or architecture that you'd like to see support for, please let us know!

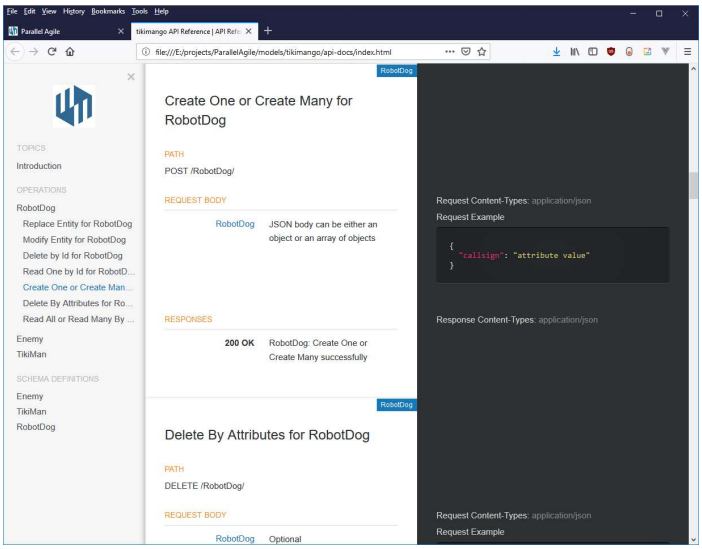
Once CodeBot has finished, you can see the connection details, security access token etc within the add-in. The same details are also available on the website, in Connection Details (which we'll get to in just a second).

Let's take a peek inside the generated zipfile (actually, better to unzip it so that the Swagger docs can be viewed correctly). You should see something like the following:



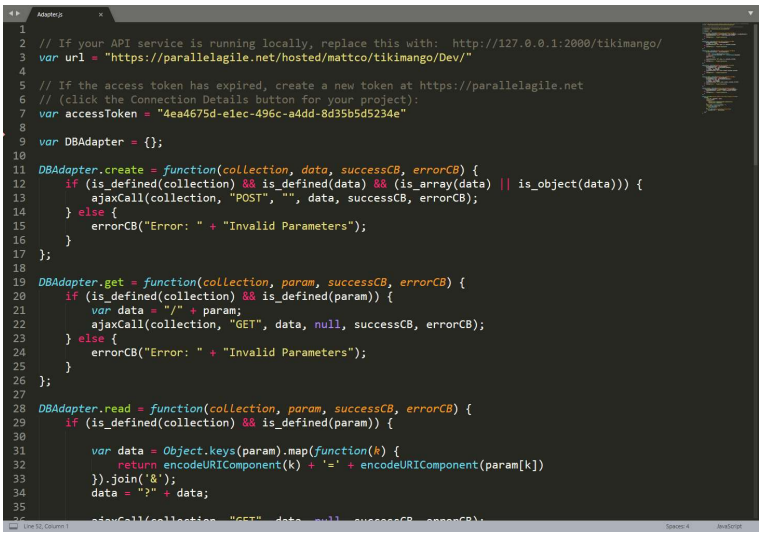

Please note you'll only see the Server folder if you have the Paid license. The Java folder currently contains a placeholder client for accessing the new REST API via Java; we're updating CodeBot to generate a proper, type-safe Java client. Watch this space!

Navigating into api-docs, you'll see Swagger documentation for the new API; this can be referred to when you're calling the cloud-hosted version (more about that in a moment):



Note the Request Example on the right – the example RobotDog class has just the one attribute, “callsign”; however it’s likely that your domain classes will contain lots of attributes; they’ll all be shown here as a JSON object that can be placed into a POST request body.

The JavaScript folder contains a REST client which is primed ready to call the hosted API, and has a 24-hour access token included at the top of the “Adapter.js” file:



```
// If your API service is running locally, replace this with:  http://127.0.0.1:2000/tikimango/
var url = "https://parallelagile.net/hosted/mattco/tikimango/Dev/"

// If the access token has expired, create a new token at https://parallelagile.net
// (click the Connection Details button for your project):
var accessToken = "4ea4675d-e1ec-496c-a4dd-8d35b5d5234e"

var DBAdapter = {};

DBAdapter.create = function(collection, data, successCB, errorCB) {
    if (is_defined(collection) && is_defined(data) && (is_array(data) || is_object(data))) {
        ajaxCall(collection, "POST", "", data, successCB, errorCB);
    } else {
        errorCB("Error: " + "Invalid Parameters");
    }
};

DBAdapter.get = function(collection, param, successCB, errorCB) {
    if (is_defined(collection) && is_defined(param)) {
        var data = "/" + param;
        ajaxCall(collection, "GET", data, null, successCB, errorCB);
    } else {
        errorCB("Error: " + "Invalid Parameters");
    }
};

DBAdapter.read = function(collection, param, successCB, errorCB) {
    if (is_defined(collection) && is_defined(param)) {

        var data = Object.keys(param).map(function(k) {
            return encodeURIComponent(k) + '=' + encodeURIComponent(param[k])
        }).join('&');
        data = "?" + data;
```

(You can always generate a new token via the CodeBot web console).

If you’re instead running your own instance of the generated Express service, simply change the URL in the above Adapter.js to your server IP address.

Of course, once you have all this code, it would be great if there was a way to simply test it all without having to do any further setting-up. Luckily, there is a way!

## Using our Cloud-Based Hosting Service to Test Your Generated API

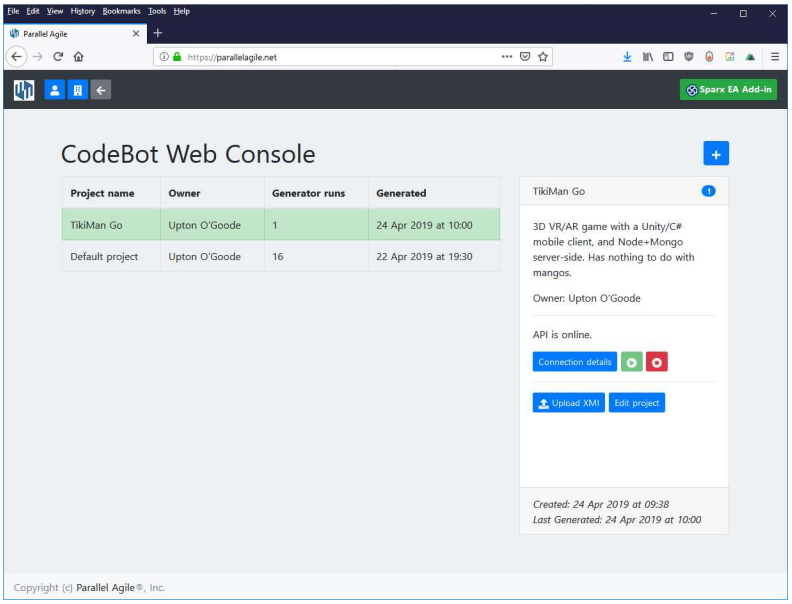
When you run CodeBot from the Enterprise Architect add-in, we automatically create and deploy a secure, cloud-hosted instance of your new REST API, complete with its own MongoDB database. The zipfile that you download is all set up to call this hosted API directly; so you’re all set to test your new executable domain model immediately, just from a few clicks within Enterprise Architect...

You should find that this process is pretty straightforward and self-explanatory; however just to illustrate how easy it is, in this section I’ll run through the process of calling the hosted API.

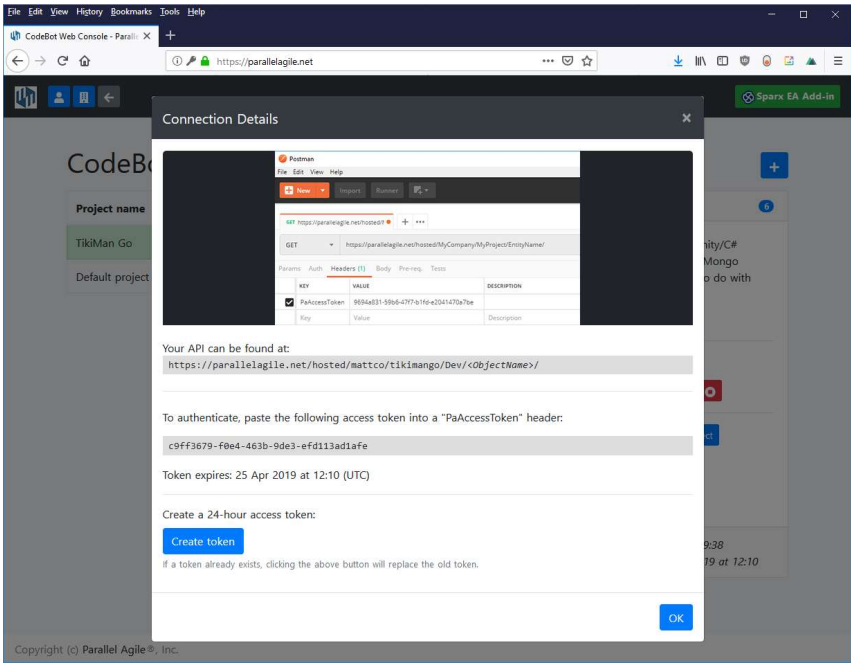
Using the hosted API is entirely optional, by the way. We would kind of expect you to use the hosted API for testing and prototyping, and then switch to a server or cluster within your own

network or hosted zone for UAT, production etc. That said, if you're interested in using our hosting service for a production system, please get in contact.

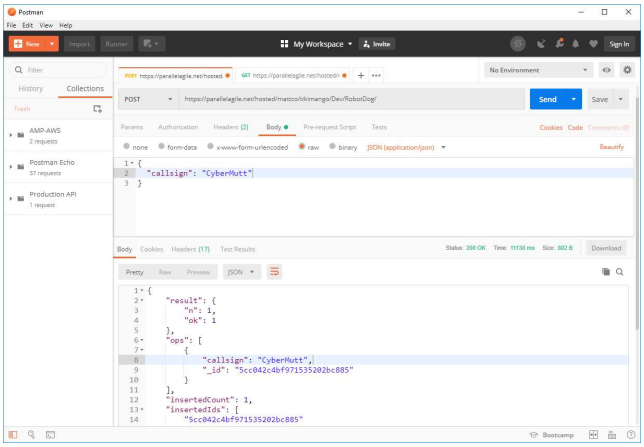
Time to return to the CodeBot web console... if you refresh the page, you'll see that there's now a hosted API available:



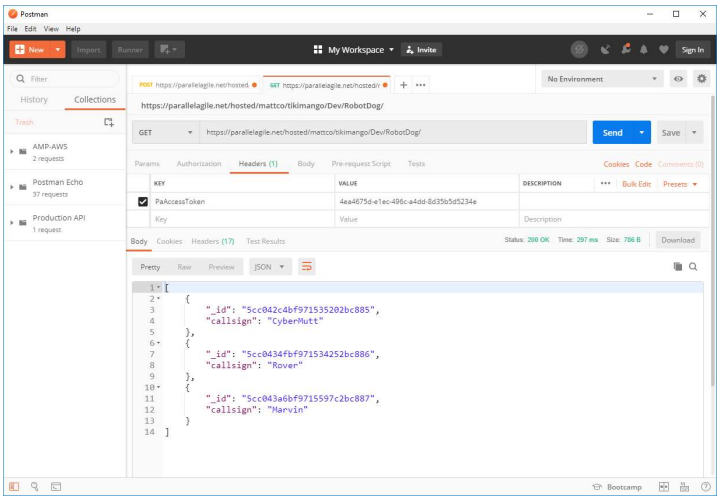

The hosted API can be switched on/off via the start and stop buttons. But for now let's try calling it. Have a look in Connection Details:



Following the instructions and example Postman screenshot, copy & paste the URL and PaAccessToken into Postman (or REST client of your choice). We'll send a few POSTs to create some test data, then retrieve all the rows via a GET:




The first request takes a good few seconds as the hosted API is running from a cold start; however, subsequent requests are faster. Here's the GET response:




There's plenty more to explore, and yet more features in active development, but hopefully that gives a good taste of how CodeBot can help to accelerate your agile project. In short, CodeBot creates a working, live, demonstrable REST API and NoSQL database directly from your domain model in Enterprise Architect. The idea is that this enables you to create a prototype very early in your project (on day one!), which can work wonders in terms of collaborative understanding, when iteratively improving the domain model through a feedback-driven approach, and while thinking about the design, architecture and UX.

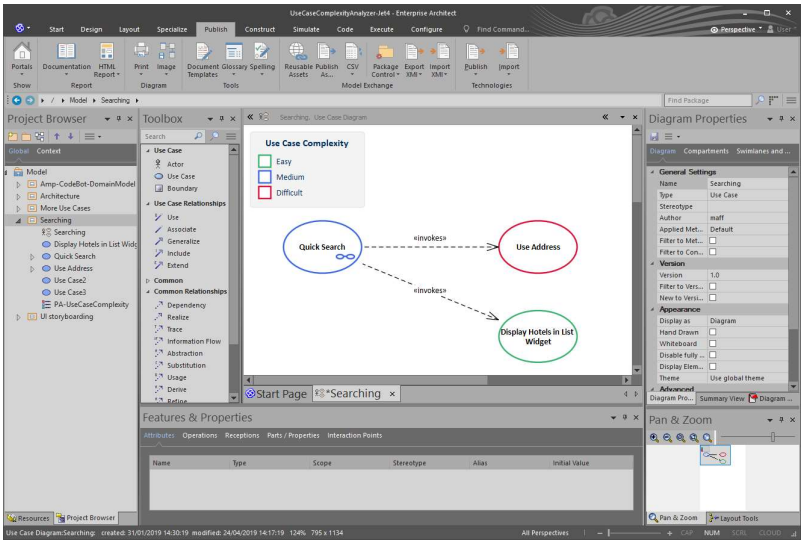
Mongo and Node are our initial targets for CodeBot, so if you're doing MEAN Stack development this functionality is immediately useful. We'll be adding additional target databases and APIs on an ongoing basis, so stay tuned to [parallelagile.com](parallelagile.com) for updates (or

better yet, sign up for our high signal/low noise mailing list at [parallelagile.com/registration.html](parallelagile.com/registration.html)).

I'll finish off by describing some additional features that are also available in our Enterprise Architect add-in.

## Use Case Complexity Analyzer

In addition to providing CodeBot access, we've built a few other useful things into the Parallel Agile Add-in. For starters, we've built a use case complexity analyzer that looks at any nested diagrams within the use cases (e.g. activity diagrams, sequence diagrams, robustness/MVC diagrams) and scores the complexity of your use cases as easy, medium, or difficult, then color codes the use cases on the diagram.



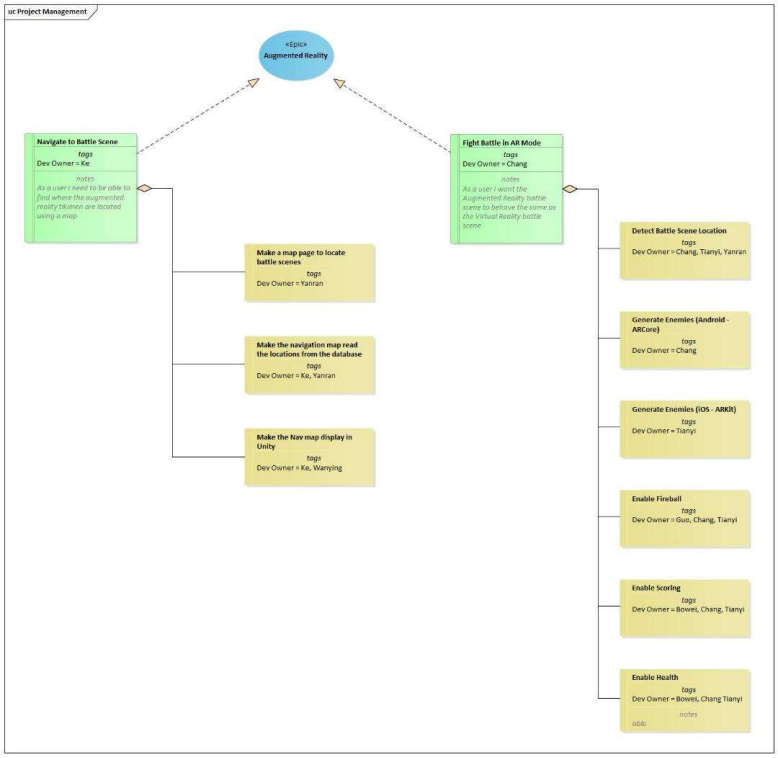

For those of you who thrive on details, here's how it works: The add-in puts the complexity value into the tagged values for each use case; it then creates a color legend based on the tagged values, and adds the legend to the diagram. If you navigate away from the diagram (say, to add some detail into one of the use case's sub-diagrams) and then return, the complexity values are automatically recalculated. So the colors aren't permanent – if you remove the legend from the diagram, the use cases will return to their normal formatting.

Rapid use case complexity analysis is a particularly valuable capability to have when you are partitioning a project for parallel development. The trick to keeping the three development phases at a month apiece is to make sure that no developer gets a task assigned that will take more than a month to complete. You can run the complexity analyzer at any time as your design evolves.

## Parallel Agile MDG Technology – supports Sprint Plans

Parallel Agile is agile (feedback-driven) on the project management side, but use case driven (plan-driven) on the technical side. These two worlds of feedback and planning intersect at the Sprint Plan.

When you install the Parallel Agile Add-In you'll automatically install the Parallel Agile MDG Technology. The MDG Technology includes a Sprint Plan diagram type that allows you to diagram out your User Stories, Tasks and Epics, and keep track of which developers are working on which tasks.




Over the upcoming months we'll be adding more functionality to automatically populate backlogs and Kanban boards from sprint plans.

The above screenshot is an example from one of our Parallel Agile test projects that I'm currently managing – a VR/AR mobile game project called TikiMan Go. We're currently in the middle of trying to accelerate the development of our Augmented Reality "battle scene" by adding more programmers. Despite Brooks Law claiming this is an impossibility, our experimental evidence is that it happens on a regular basis.

## Wrapping Up

This article has covered the main features of the Parallel Agile Add-in for Enterprise Architect, including the CodeBot, use case complexity analyzer, and Scrum profile. We've also given a quick introduction to how Parallel Agile allows product development to be accelerated by having a CodeBot-created Executable Architecture in place at project inception, greatly simplifying the task of integrating work done by many developers in parallel.

Please visit www.parallelagile.com for more details, and feel free to email me with questions.