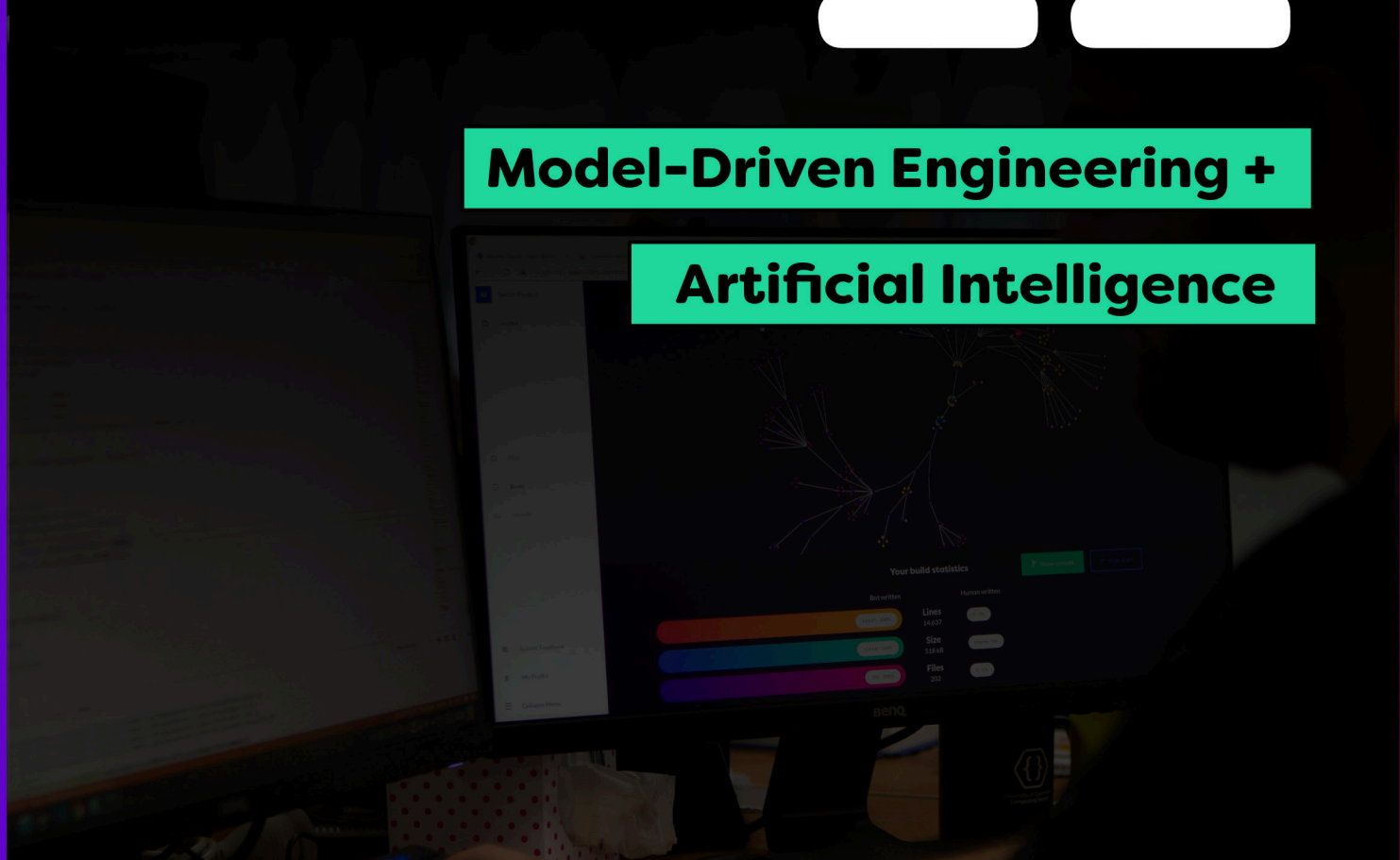
CODEBOTS

White Paper

'22

Model-Driven Engineering + Artificial Intelligence



"Access to developers is a bigger threat to success than access to capital."[1]

Message from the CEO

Decision makers are currently looking for innovative ways to amplify their people, processes, and tools amidst a world-wide shortage of software developers. This shortage poses a significant threat to organisational success. This white paper is an in-depth guide into amplifying business efficiencies during resource shortages. We will address many of your most important questions about Codebots and the role it plays in solving these problems to inform your operational and strategic decisions.

Codebots is on a mission to create bots that code like the world's best developers. Imagine if you could create a bot to code in any technology stack! The good news is we have discovered a world first approach on how you can achieve this today. It is now possible to use a bot to code the vast majority of a software application alongside your software developers. But this is the beginning of the story...

Our vision is to provide a platform that empowers cross-functional software development teams to build better software systems. This can be done by unleashing the potential of *models.* Once you accept a simple truth, that everything is a model, you can unlock the creativity of all your team.

The format of this white paper is to follow the five W's and H approach; who, when, what, where, why, and how. These will help you gather information about Codebots and start you on your journey of understanding. For example, *where* does Codebots fit in the market? This is the first question we address to help you orientate yourself and Codebots market position. We then dive deeper with other pertinent questions and finish the white-paper with some strategies and case studies.

Eban is the founder and CEO at Codebots. He received his doctorate from The University of Queensland (2013) in Model-Driven Engineering and his Masters from Queensland University of Technology (2004) in Artificial Intelligence. He is an advocate of using models as first class artefacts in software engineering and creating not just technologies, but methodologies that enhance the quality of life for software engineers. When he is not coding or running the company, you can find him mountain bike riding or finding his zen in martial arts.

Eban Escott
founder and CEO at Codebots

CODEBOTS

Table of Contents

Where does Codebots fit in the market?

Robotic Process Automation (RPA) is aimed at automating mundane routine tasks that a user must do on a legacy system. Some examples include data manipulation via a User Interface (UI), uploading/downloading files, and completing routine reports.

RPA has been gaining market attention as organisations are able to automate routine tasks and free up their people to do other, more meaningful work.

> "The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency."
>
> – *Bill Gates*

The limitation that the market has found with RPA is that it does not address the underlying problem of the legacy system that the tasks are being automated on. So, in effect, by using RPA they are kicking the can down the road. Unfortunately without longer term strategies, the problems associated with legacy systems are becoming worse.

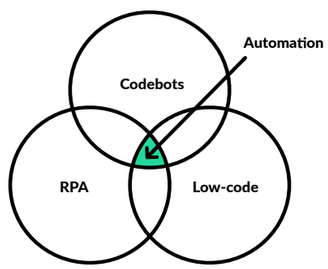
Low-code is an approach to building software that requires little to no programming experience. The low-code platform usually has a visual drag-n-drop UI whereby an application's UI, data, and business logic can be created. Low-code has been gaining market traction as it empowers citizen developers (non-developers) to make software applications.

The limitation that the market has found with low-code is that while it may empower citizen developers, this is only within the bounds of the low-code platforms pre-built components.

To go beyond these pre-built components requires a software developer however, it is extremely difficult for developers to customise. But this discovery can be too late for an organisation as they are vendor locked to the low-code platform and control has been lost.

Codebots intersects with RPA and low-code with a common goal of automation, but it differs in a number of ways:

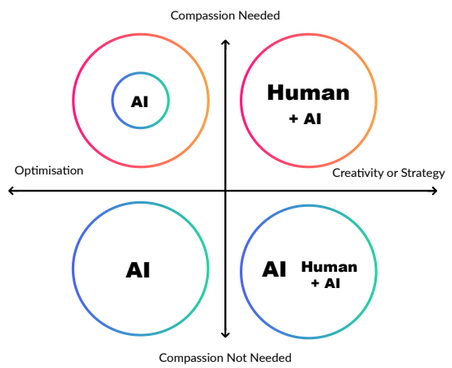
- RPA does not change/modernise the legacy system. Codebots can be used to modernise the legacy system by automating tasks that a software developer usually does (like writing code). Unlike RPA, Codebots addresses the underlying issue of the legacy system through a process of modernisation.
- Low-code runs counter to true innovation. It empowers citizen developers but this is at the expense of the rest of the team and the organisation's ability to innovate. Codebots brings balance to building software as it empowers the entire team.
- Both RPA and low-code platforms aim to vendor lock your organisation. Codebots avoids vendor lock by committing all the code, models, and templates into your source code repositories. Even though our platform is not open-source, the artefacts you need to continue using traditional development is under your control.



> We were invited to do the keynote talk at the low-code track at MODELS 2021, the premier annual conference on MDE. Controversially, we spoke about *Low-code: the good, the bad, and the ugly*. The video presentation is available on our website.[2]

It is generally accepted that computers and robotics are changing the landscape and future of work. This has been seen across many industries and in recent history, breakthroughs in artificial intelligence (AI) have brought this into focus. A big moment in history was in the late 90's when DeepBlue was able to beat the reigning world chess champion Garry Kasparov, after all the controversy has subsided, Kasparov now advocates for augmented teams where both computers and humans bring different strengths to the game.

In a TED talk by Kai Fu Lee[3], he takes the ideas of augmented intelligence further and looks at some different axis to help determine what types of jobs will end up with different uses for AI. As seen in the image below, the horizontal axis measures from **optimisation** to **creativity or strategy**. So routine to mundane tasks are found towards the left and creative thinking tasks are found to the right. Lee also introduces a vertical axis that measures the **compassion** needed for the job.



Source: Kai-Fu Lee: How AI can save our humanity | TED Talk

Jobs found in the bottom left quadrant are open to be replaced completely by AI. Other jobs will end up with a various mix of AI but a human will still be needed in some capacity. The question faced by decision makers is to what extent will it change their mix of people, process, and tools? People will need to be up-skilled, processes will need to evolve as the work has changed, and the best tools will be needed that leverage AI to augment teams to create success.

Where do software developers sit in the quadrant? And more widely, where do the various roles in a cross-functional software team sit?

There are many tasks that software developers do that fall under optimisation and no compassion needed. Our stats show that upwards of 80% of a software application can be written by a codebot. For some projects, it is as high as 99%. The reason for this is that many software applications use patterns and architecture that is about managing data. This is especially true in applications that are underpinned by a database. A lot of effort goes into CRUD[4] (create, read, update, and delete) operations, from database management all the way through to API's like REST and GraphQL. There are also cross cutting concerns like security that are consistently repetitive (good candidate for augmentation).

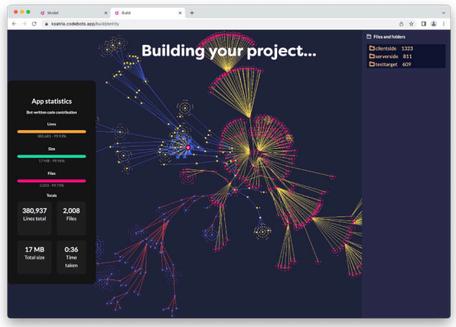
However, there are also many tasks that software development teams do that require creativity and compassion. The question then shifts to the best way to capture this and enable an AI to help. In the next section we discuss the role of *models* in software development and how they can be used to unleash the creative power of teams. Further to this, it is important to make sure that when a codebot writes code, it must look like a developer has written it. If a developer is able to understand the code written by a codebot, they can extend and customise it for any requirement. This is one of the areas where low-code platforms have failed.

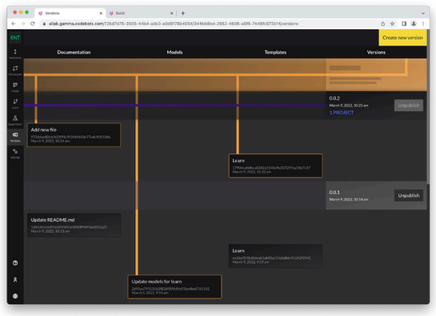
What is the Codebots platform?

The Codebots platform is made up of two products; AI Lab and App Studio.

App Studio

App Studio is used to build software applications. This is done by creating a new project and selecting a codebot like C#Bot or SpringBot. The team then updates the models for the application, hits the build button, watches the bot write code and commit it to a Git source code repository. The team can then pull the code from Git and can add their custom code alongside the bot written code to meet a requirement as necessary.



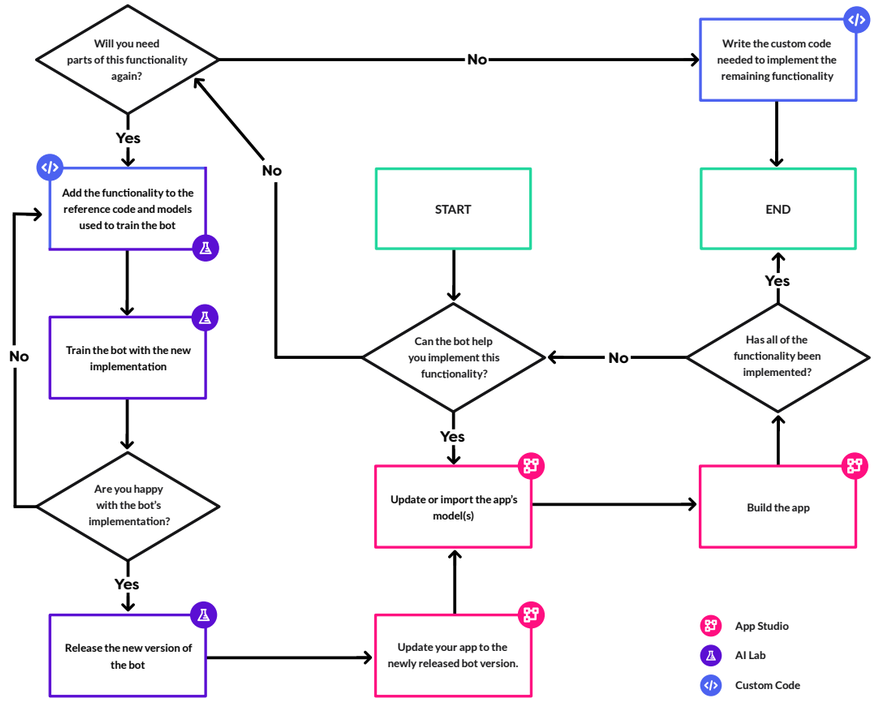


AI Lab

AI Lab is used to build your own codebot using the technology stack of your choice. When you use the default codebots from App Studio, like C#Bot or SpringBot, you can add custom code inside the protected regions but if you want to modify code outside these regions you are blocked. So, to take control of your codebot you will need to train it inside the AI Lab platform so you can update any of the code it writes. Alternatively, if you have a legacy system to use as a reference, this makes for a great starting point for your modernisation strategy.

The following flowchart shows the process of how a developer teaches a bot to code in any technology stack. At a high-level (see the legend), App Studio is where developers create many applications and use the codebots to help implement some functionality. AI Lab is where developers create and teach a bot. Once they are happy with the bot, they release the bot for use inside the App Studio. Lastly, developers can write custom code to either show the bot a good example, or, to implement extra functionality.

One of the most important aspects of this flowchart is that a developer writes the reference code and models used to train a bot. Like teaching a child, it is best if you first show the child a good example so they can replicate it. Similarly, a developer writes the reference to show the bot what it is to learn. In this way, this flowchart can be followed many times and a bot can continue to evolve and learn new things. Or, if the bot does what you need, you can reuse it on many projects inside App Studio!



Who is Codebots for?

Codebots is for cross-functional software teams. Generally, software teams are made up of a variety of people with different skill sets. Some people are able to bridge a number of different roles, though it is accepted that a diverse team, will result in better solutions.

- A *product owner* is the visionary and drives the direction of the project. They ensure standards are met by working with the team throughout the way of working and ultimately approving checkpoints along the journey.
- A *squad lead* is the organiser of the project and the go to that manages the expectations of all stakeholders. Their goal is to bring out the best in people and foster a spirit of pride and comradery. They work closely with all team members but are the main conduit between the product owner and the squad.
- An *account manager* is the sherpa of the project by guiding and advising the team on the path to success. They are the educators of product owners to begin expectation management early and continuously nurture the account by building trust.
- A *designer* makes the team look good! They do this by applying design principles to not just the visual aspects and user experience of an application, but present thinking outside of the box and ways to bring out the creative side of all team members.
- A *software developer* is more than just a coder, they are problem solvers that seek the best solutions through use of architecture and technologies. They thrive in the creative process and ultimately bring the solution to life through their code. It is an art and science.
- A *DevOps developer* breaks down the barriers between development and operations. They seek the truth by enabling metrics, insight, and gates through the use of pipelines at any stage of the way of working. They have skills in both coding and platforms for DevOps practices. They pride themselves at the ease of releasing and insights into the full lifecycle.
- A *citizen developer* is a domain expert and technically capable of understanding and solving business problems. Even though citizen developers are not proficient at coding, they are able to use and connect different technologies and tooling together for creative solutions.
- A *solution architect* is a wise developer that has seen many projects from start to end. They inspire the squad to master the quality of work on projects and ensure that high risk tasks are addressed early through tech spikes. They consider both the functional and non-functional requirements of a software application.

Other common roles found in software teams include business analysts, quality assurance, and others. The important point is that software teams are made up of lots of different roles and everyone needs to be empowered to do their job. Low-code tools do not live up to this.

The Codebots mantra is to empower the entire software team including the developers. This is done by making sure the source code is high quality, easy to understand, and can be extended the way developers are use to. It is also achieved by using models to create a shared understanding of the application. Read more about this on the next page.

Who benefits from using models?

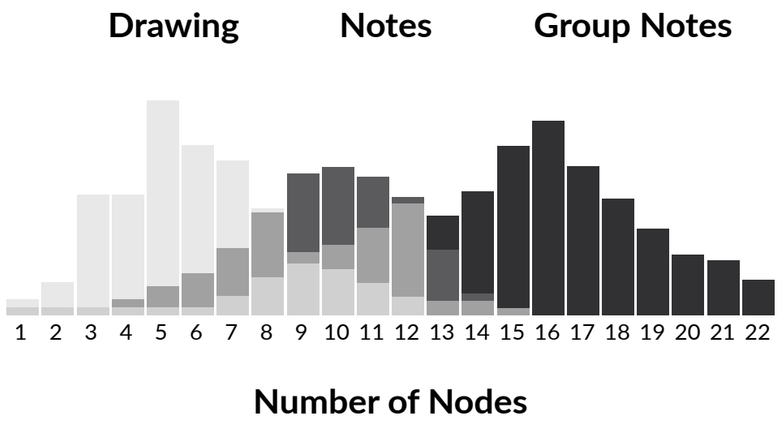
Using models is one of the keys to unlocking the creative power of your team to solve complex problems. A really good way to understand the depth of this statement is to solve a wicked problem: tell me how you make toast[5]. In this TED talk, Tom Wujec uses a simple process of how we make toast as a way to highlight how collaboration and shared creativity leads to better solutions.

In the first experiment, participants are asked to draw the process of how they make toast. In the TED talk, Tom calls this a systems design, but what people are making is a model. Most people draw between 5 and 13 nodes (steps) in their process of making toast.

In a second experiment, participants are given some sticky notes and asked to do the same thing. Interestingly, the number of nodes in the process goes up significantly compared to simple drawing. A contributing reason for this is that the sticky notes make it far easier for the participant to update their process by adding in new sticky notes or moving around the sticky in the process for a better solution. The ability to easily manipulate the model, in this case through sticky notes, allows people to better explore solutions.

In a third experiment, participants are also given sticky notes but now work in groups. The sophistication of the solution goes up again, as seen in the graph below that compares the three experiments. People working together on a problem leads to robust discussions and creates a forum for ideas to be heard and included, merged, or influence the outcome in different ways. This shows insight into another one of the keys that using models unlocks, collaboration!

How does this relate to Codebots? For us, everything is a model. So, using models will allow for better solutions. Like using sticky notes, a model allows people to explore solutions without getting locked in. But on a deeper level, what you need is for your team to be able to create a model that fits their specific domain. And to achieve this, you must have control of your metamodel. What is a metamodel you ask? The metamodel describes what can be found in a model. So once you have control of the metamodel, you can model anything. That is powerful stuff.



Source: TED Talk Tom Wujec - Got a wicked problem? First, tell me how you make toast

How does Codebots work?

Codebots is a mix of Model-Driven Engineering (MDE) and Artificial Intelligence (AI). There is a blog article written by Dimitris Kolovos, Professor of Software Engineering from the University of York, called What is Model-Driven Engineering[6] that has an explanation for beginners to this area. We use AI to learn the templates from the reference and model to create a target. This is best visualised in the graphic below.

At the top of the graphic is the classic input-process-output flow that all computers must conform to (put here to guide your thinking in this way).

The middle flow is representative of what happens in the AI Lab. The reference and the model are used as input for the bot learning. The bot makes templates and generates a target. Lastly, the bot compares the target to the reference to see how well it learnt and it may update the reference to be consistent with the templates (this helps future learning).

The bottom flow in the graphic is representative of what happens in the App Studio. A developer models the application and the templates are used to generate a target application.

The templates are what was previously learnt in AI Lab.

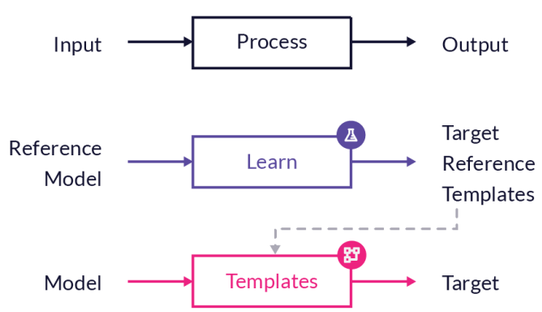
To make the above input-process-output flows work, we use the RMTT (Reference-Model-Template-Target) pattern and each of these four are stored in seperate Git repositories for easier reference. An explanation of each is found below:

Reference: The example application that the bot learns from. A developer writes this application in their preferred language, framework, and architectural style. This lets the developer control what the bot learns from.

Model: The model that is used to represent the application. A developer can create new types of models or reuse industry standard models like an entity diagram, OpenAPI, RAML, etc. The developer is able to do this because they have control over the meta-model.

Template: The template is a standard code generator that developers are use to. There are variables with selection and repetition statements like most programming languages.

Target: The target application is what is generated from the templates.



How do you learn about Codebots?

You can sign-up and start using the platform today at Codebots.com. There are three ways for you to get started on Codebots:

App Studio: Use an existing codebot and get familiar with the development process. A good option for a gentle introduction.

AI Lab: Get stuck in and create a new codebot for yourself. Not for the faint hearted!

BotCamp: A guided learning process lead by the world's best model-driven engineers. Awesome for onboarding your team.

BotCamp

BotCamp is the continuous education of people, processes, and tools to enable organisations to build better software. BotCamp achieves this by covering three streams of education:

- Codebots. CB = MDE + AI
- Way of Working
- Projects and other technologies

Course format:

- Short: One week full-time with a deep dive into Codebots to cover both theory and practice using AI Lab and App Studio in a CI/CD pipeline.
- Long: Starts out the same as the short course. Then the second and third week are an immersion into an Agile way of working while building out a capstone project using Codebots and your choice of project.

Delivery format:

- Majority of lectures are pre-recorded and the students are sent them ahead of time.
- Some lectures are live. This includes the welcome lectures at the beginning of a course, and the summary/showcase lectures at the end of the course.
- Each day there are two time slots put aside for students to join via a video link for a Q&A to allow for the best global timezone coverage. This time will vary from course to course. Students attend one session per day.
- The time slots for Q&A will have assigned topics to ensure the correct experts are available.
- The examples spoken about in the lectures have code repositories setup so the students can clone them and submit their own work via branches and merge requests.
- There is a messaging channel setup for the course where students are free to interact and ask each other questions. There will also be a long running BotCamp zoom meeting whereby the students can jump on anytime with other students or request an instructor to jump in.

> Interested to learn more about BotCamp? Information about the course formats is readily available on our website.

When is Codebots used with Agile?

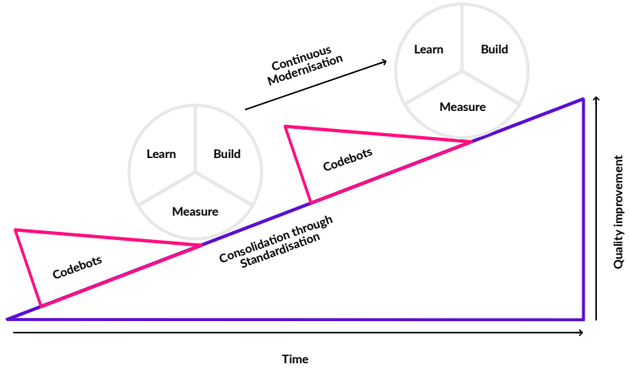
Every organisation will have their own way of working. A common thread that most organisations have accepted is some inspiration from the Agile Manifesto's[7] values and principles. This guiding light helps shape the organisations way of working and many of the modern methodologies that can be found.

Codebots does not replace your way of working and is complimentary to agile processes. This can be demonstrated by addressing what is common to all methodologies. The first and most obvious is the code. All software projects have source code that is being developed and committed to a repository. Not having control of your source code is a fatal mistake and this is where low-code, no-code, and RPA tools hamstring their customers. Codebots however approaches this differently, all source code written by a bot is committed to a source code repository that you control.

Collaboration between people is another common thread across all software development methodologies. In the Agile Manifesto, one of the four values is about collaboration. Unlocking the creativity of people through a shared understanding is considered to be one of the human centred tasks that an AI will not be able to achieve (well not anytime soon).

In your way of working, you will most likely have specific meetings, ceremonies, or stages in the development process where problems are solved through collaboration. To take this concept to the next level is to have a codebot subsequently use those models to write the vast majority of the code for the project (icing on the cake).

Another common thread across all software development methodologies is that the process is about increasing the quality of the outcome while decreasing the risks of it going off the rails. Setting the governance and standards of a project is extremely important. This can be around the processes used to develop the software, or it can be about the actual implementation in the source code. For example, as you build a software application through a series of iterations, you can use AI Lab to train a codebot on something like an API protocol. Once the codebot has been taught this, any future requirements will also use the same architecture and implementation. This is an amazing way that you can control the standards and governance across your source code. This is best visualised in the following graphic. As time progresses forward, the quality of the project increases as the codebot is used to set the standards.



When is Codebots used with DevOps?

Some organisations have started on their DevOps journey. If you haven't started yet, now is the time. DevOps is about breaking down the barriers between development and operations. Traditionally, these departments have operated independently and this has lead to all sorts of friction in the software development lifecycle. DevOps can heal some of these wounds and opens up many opportunities for other improvements as well.

As seen in the following graphic, the stages of DevOps are represented using an infinity symbol. As wise developers know, software projects are never complete, so the infinity symbol is a great visualisation as the project flows through the various stages. There is no mandate to what the stages of the process are, organisations are free to specify their own.

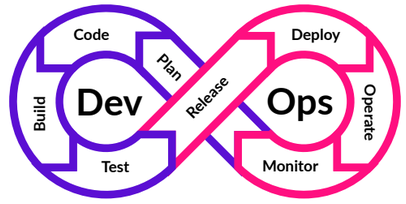
As teams become more mature using DevOps they move up from continuous integration, to continuous delivery and finally to continuous deployment. There is an emphasis on the various tools, or toolchains, that can be used to help with a teams maturity. The CI/CD toolchain (a.k.a. pipeline) is a key concept in DevOps. Using the pipelines various tools can be used to ensure the quality of the software is high; like testing, security, code quality, and performance, just to name a few.

However, to get the most out of your pipelines there is something that you must have control of - your source code. This is another failing of low-code as access to your source code is usually limited and difficult to control. At Codebots, we commit all source code into git repositories that you control.

This has many profound knock-on effects. One of the best is that it enables better DevOps as you have control of the source code in your own git repositories.

The source code that we commit is not just the application code, it also includes the model and template code. No low-code platform gives you that. For software teams to be empowered they also must have control of the models and templates because it leads to the next exciting knock-on effect; model-driven DevOps! We are taking on the grand challenges of MDE[8] and you can read more about this in our journal publication Using DevOps Toolchains in Agile Model-Driven Engineering[9].

DevOps will heal the world.



Modernisation strategy

Across the globe today, legacy systems are one of the biggest problems faced by the software industry. The problem cannot be overstated, and as we travel further into the future, legacy systems could arguably become the most common problem across all business sectors and industries.

Industry leaders are looking at different modernisation strategies. For example, Gartner recommends seven options for modernising[10]: encapsulate, rehost, replatform, refactor, rearchitect, rebuild, or replace. Or, Cognizant recommends a smaller set of five options: total transformation, gradual replacement, duct tape approach, improve existing, or no system change. But there is something about these options that doesn't sit right, every time we modernise a legacy system aren't we just creating the same problems all over again for tomorrow?

On the next page, there is a description of some technical design patterns for the modernisation of legacy systems. To date, many organisations have used a combination of the anti-corruption layer pattern[11] and the strangler fig pattern. During the initial phases of a modernisation, anti-corruption layer is used to encapsulate the legacy system so that it is isolated. From there, the strangler fig pattern is used to divide-and-conquer the legacy system into a modern system. But again, the new systems become the legacy of tomorrow and history repeats.

The best chance you have to minimise the impact of this occurring is to use the exoskeleton pattern. Before undergoing the divide-and-conquer approach of the strangler fig, an exoskeleton is built around the legacy system that captures the requirements of the legacy system at a high-level of abstraction than the code. The exoskeleton is built using model-driven engineering, which includes models, templates, and transformations that all become as important as the source code itself. From this position, modernising tomorrow's legacy system is far smoother as it's possible to use the models, templates etc. that were not previously available.

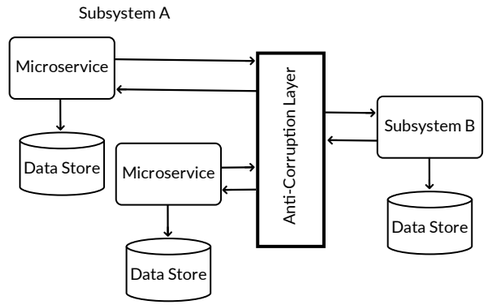
> "The entire history of software engineering is that of the rise in levels of abstraction."
>
> – *Grady Booch*

So, why hasn't everybody been doing this? The truth is that traditionally it takes a huge amount of time to build all of the models, templates, and transformations needed for a legacy system. And when people are faced with this amount of effort and what is required understanding the legacy system, it usually gets thrown in the too hard basket and people revert to what they know. In other words, there is a high costs of entry for the perceived return on investment. But the good news is that we have discovered a world first approach using AI that saves huge amounts of time on a project making this approach a viable economic choice for people today.

As explained in the section about how does Codebots work, from the RMTT (reference, model, templates, and target) pattern we can see that the reference and model and inputs into the AI learning. So, the AI can learn the legacy system as the reference and infer a model from something like a database schema or API. The output from the AI learning is a set of templates whereby the target matches the reference. Importantly, the templates traditionally would take developers a significant amount of time, but now we have an AI that can do this for us. This bring the cost of entry down and makes it possible for you to enjoy the well-known benefits of model-driven engineering.

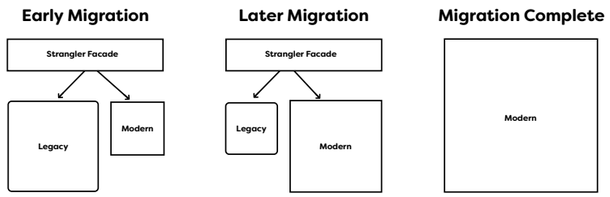
Anti-corruption Layer Pattern

Implement a façade or adapter layer between different subsystems that don't share the same semantics. This layer translates requests that one subsystem makes to the other subsystem. Use this pattern to ensure that an application's design is not limited by dependencies on outside subsystems.



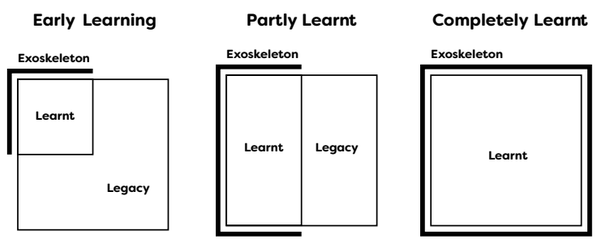
Strangler Fig Pattern

Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services. As features from the legacy system are replaced, the new system eventually replaces all of the old system's features, strangling the old system and allowing you to decommission it.



Exoskeleton Pattern

Build an exoskeleton around the legacy system to automate the migration process. The external skeleton is built using a model-driven engineering approach so that the requirements of the legacy system are captured at a higher-level of abstraction. The models can subsequently be used to refactor the legacy system or forward engineer into new applications and services.



Case studies and testimonials

workingmouse

WorkingMouse is a global leader in developing and designing software applications through innovative people, processes, and tools. WorkingMouse takes on both new greenfield projects and brownfield legacy migration projects, but for this case study the focus will be on one of the bots they built to strategically be used on greenfield projects, C#Bot.

WorkingMouse develops many new projects each year. The customers range from government, enterprises, through to associations and startups. It was identified that a lot of these projects (though in different domains) required many similar technological requirements. So, instead of starting each new project from scratch, C#Bot was introduced as a common foundation for all projects.

The technology set that C#Bot uses is impressive and it writes full-stack applications. Broadly speaking, the server-side uses C#, Entity Framework Core, PostgreSQL, and the client-side uses React. The API between the server-side and client-side uses REST with emphasis on the OpenAPI and Swagger definitions. It also uses GraphQL and a scattering of other cool technologies.

> "Build a codebot like C#Bot and use it on many projects!"

Some of the benefits gained are:

- development efficiency as the bot writes the majority of code on projects
- increased quality as there are less manual tasks for developers as bots write tests too
- consistent architecture that facilitates better knowledge sharing between projects and teams
- reduced technical debt as refactoring at scale is better enabled
- better collaboration as the team uses models to reason about requirements.

Clients

BlueScope, CS energy, Master Electricians Australia, UniLodge, New South Wales Police Force, Australian Government Department of Defence, alex bank, Moreton Island Adventures, Escape the Fake

Australian Government Department of Defence

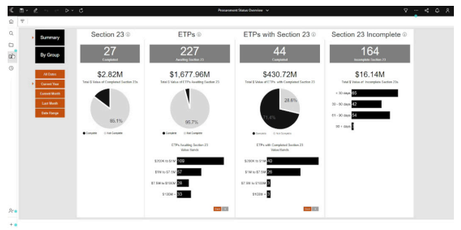
For the past five years, Codebots has assisted Defence with legacy modernisation and application development. Codebots is the technology that enabled the modernisation of Non Material Procurement digital processes. Prior to Codebots assistance, Defence had over 13 disparate systems (Excel, Access, Outlook etc.) to monitor and administer large scale procurement process. The Commercial Tracker (CT) now consolidates all this information and workflow into a single application. The comprehensive reporting that is delivered with the CT provides detailed insight into:

- what is being requested
- the volume of work and dollar value of what is being requested
- customer feedback
- who in NMP is handling each request
- Endorsement to Proceed data
- major contract data and supplier report cards.

The business intelligence delivered by Codebots provides management insight not previously available. Management have access to relevant data to inform operational and strategic decisions. The success of the Commercial Tracker expanded the requirement to include a central, user driven procurement capability called My Procurement System (MPS).

Available to every user on the Defence Protected Environment (DPE), MPS leads the user through a workflow leveraging the appropriate Defence and/or Federal Government procurement policy to drive their own procurement. The user completes the process through to Endorsement to Proceed, the required approvals and ultimately the signed contract. In addition to the automation and reporting benefits, the system will free up the NMP specialists to focus on more value-added work. The system is integrated with Defence ERP and will provide data directly to the core financial system to avoid duplication of effort.

Legacy migration is one of the major issues CIO's face, this issue is amplified in large enterprise organisations. Whether this is rouge databases and applications that have proliferated through the organisation or inflexible behemoths that control core systems, Codebots has demonstrated through its five years of assisting Defence, that we can help alleviate the risk of modernising any similar systems in large government enterprise environments.



Endnotes

1 https://stripe.com/files/reports/the-developer-coefficient.pdf

2 https://codebots.com/low-code/low-code-the-good-the-bad-and-the-ugly

3 https://www.ted.com/talks/kai_fu_lee_how_ai_can_save_our_humanity

4 https://codebots.com/crud/what-are-crud-operations

5 https://www.ted.com/talks/tom_wujec_got_a_wicked_problem_first_tell_me_how_you_make_toast

6 https://codebots.com/app-development/what-is-model-driven-engineering

7 https://agilemanifesto.org

8 https://link.springer.com/article/10.1007/s10270-019-00773-6

9 https://www.researchgate.net/publication/356491511_Using_DevOps_Toolchains_in_Agile_Model-Driven_Engineering

10 https://www.gartner.com/smarterwithgartner/7-options-to-modernize-legacy-systems/

11 https://docs.microsoft.com/en-us/azure/architecture/patterns/anti-corruption-layer