DESIGNING A CONSOLE FOR AGENT-BASED MODELS

INTRO

Simudyne is a tool for building agent-based models (ABMs). My role was to design an interface for our users to code and test models. I was responsible for user research and prototyping through to high-fidelity mockups and design specifications.

THE USER

Our users worked in banks and research institutions, building predictive models using spreadsheets and scripting languages like Python.

To build a picture of our average user and their typical user journey, I shadowed several users through the process of building a model and held user interviews to discover their pains and their goals.

Sam Brown
Risk Manager

Sam is a 37 year old risk analyst at the Royal Bank of Scotland. He has worked there for 12 years, producing predictive models and reports to quantify risk and help his bank comply with regulations.

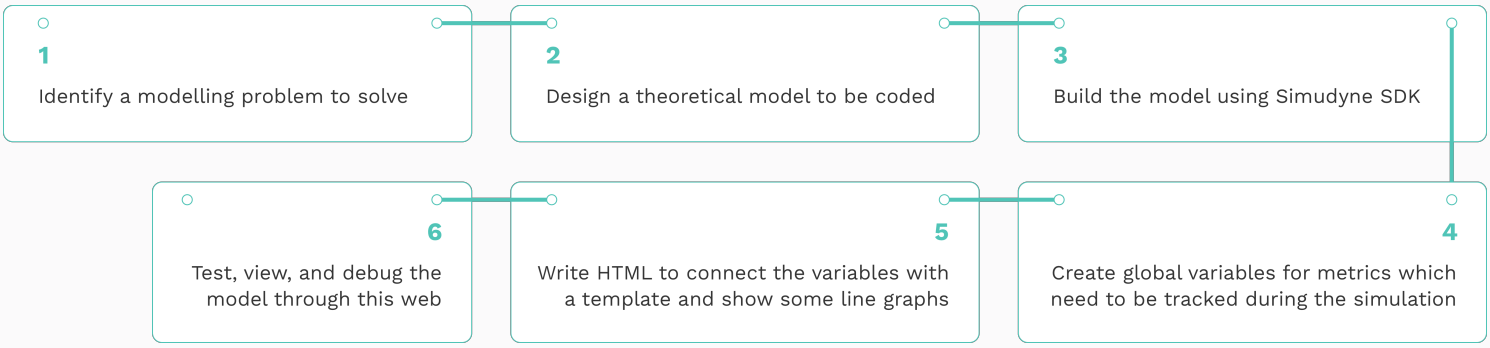
Motivations

Sam values the reputation he has built based on the reliability of his risk models and ability to present them in key reports with strict deadlines. However, as a fairly senior risk manager, he is also looking to champion new technologies and be a pioneer in the field.

Frustrations

Whilst Sam has a fairly large budget available to him, his time is limited. Building new and complex models without the experience as a computer scientist would require spending a lot of time training a new team. He wants tools that save him time and effort, to let him build the models he has in his head.

User persona of a typical user from a bank

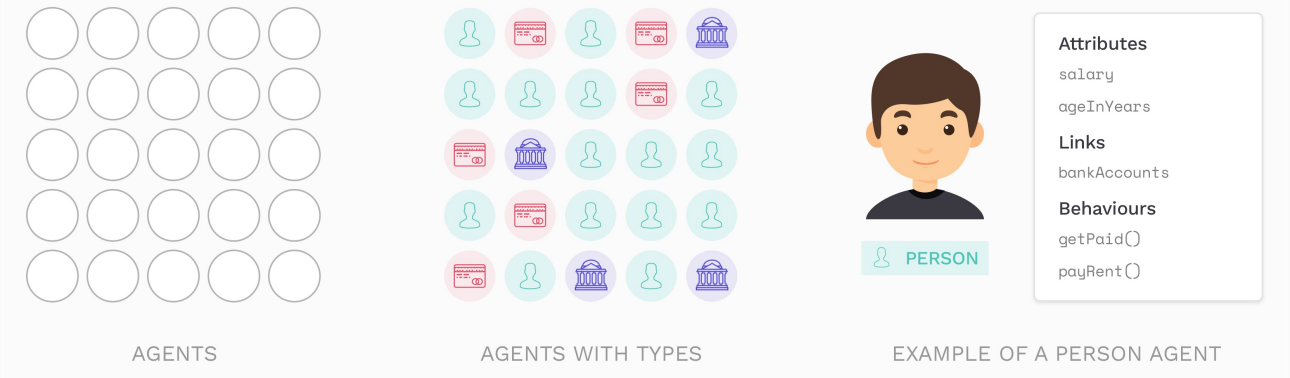


Old user flow to build models

From the initial research, it seemed that the most obvious action to take would be to cut out steps 4 and 5, by building a console that would be able to inspect and run their models without the need to do any extra coding. Furthermore, as we found that our users were particularly risk averse and time constrained, the only way to provide true value would be to focus on the novel agent-based modelling paradigm, and build the platform around this innovative technology.

AGENT BASED MODELS

As part of my research, I built several agent-based models entirely from scratch and made sure to deeply understand the modelling paradigm. In simple terms, agent-based models are representations of real-world systems, made up of independent parts which interact with each other through a set of pre-defined behaviours, links, and attributes.



Agent-based models

The fundamental principles of ABMs are quite intuitive, as the paradigm largely mimics the way the real world is laid out. The

difficulties for our users were in building and debugging computational models within an agent-based framework, using tools that are not optimised for this way of thinking, such as spreadsheets which only allow one way links, and Python which would involve a large amount of coding.

Building ABM into the core of the design was key to creating a product for our users that would distinguish us from our competitors and greatly improve the modelling experience. One example of a feature that was designed with this in mind was the way that variables were analysed from the console.

The old way: users would create global variables in their code and then attach them to a chart within a javascript file in their web template.

The new way: users would simply fire up their console to show their model and from there be able to access each attribute of each agent type.

DIMENSIONALITY

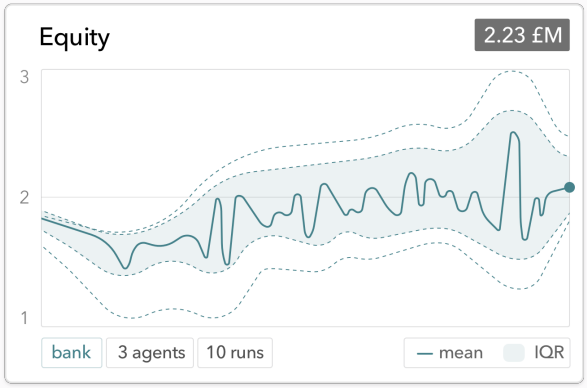
One of the key challenges that arose in designing an ABM console was the number of dimensions of data we had to deal with. ABMs are not deterministic, but probabilistic. This means they should be simulated many times to produce a distribution of results. So every time a model is simulated, data is generated for each agent attribute, and at every time point, creating three dimensions in the data.

We wanted to avoid 3D graphs at all costs because rendering many 3D graphs on the same page would be a UX nightmare. Instead we grouped together the agents and the runs of the simulation, so for example ten runs of ten agents would generate 100 points on the same graph. We would then allow the user to choose which attributes they wanted to display on the UI. We decided to test this feature by creating a clickable prototype on Sketch and giving people a set of instructions to see how well they could follow them.

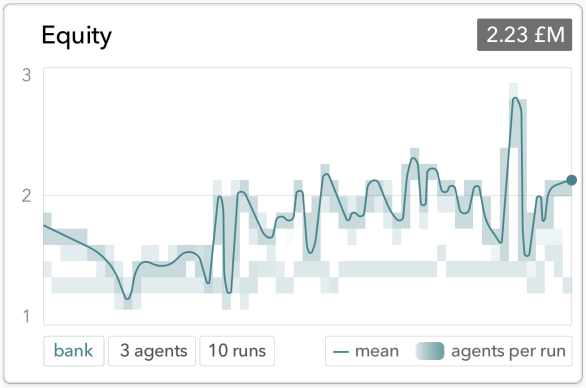
Sketch Prototype

Through the process of continual idea generation and prototyping we designed a novel way of dealing with the dimensionality challenges in a way that avoided cluttering the interface with vast swathes of meaningless numbers, but which allowed the modellers to easily view high-level variables whilst being able to drill into lower-level information when necessary.

In order to show the dimension of agent population and simulation runs, we give the user the choice of three types of chart: average, density and simply showing all of the data.



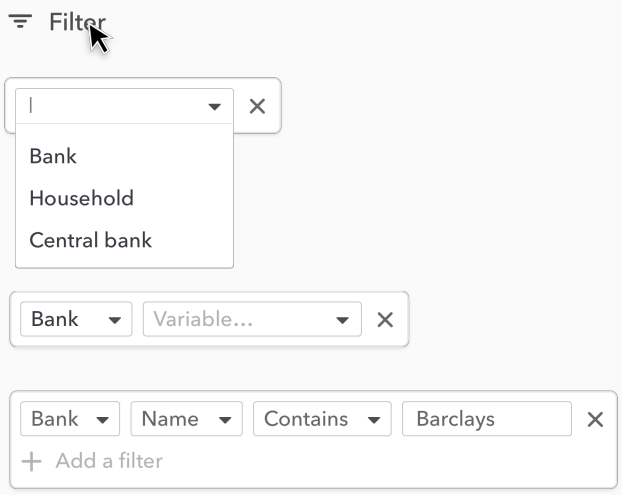
SHOWING MEAN AND INTERQUARTILE RANGE



SHOWING A DENSITY PLOT

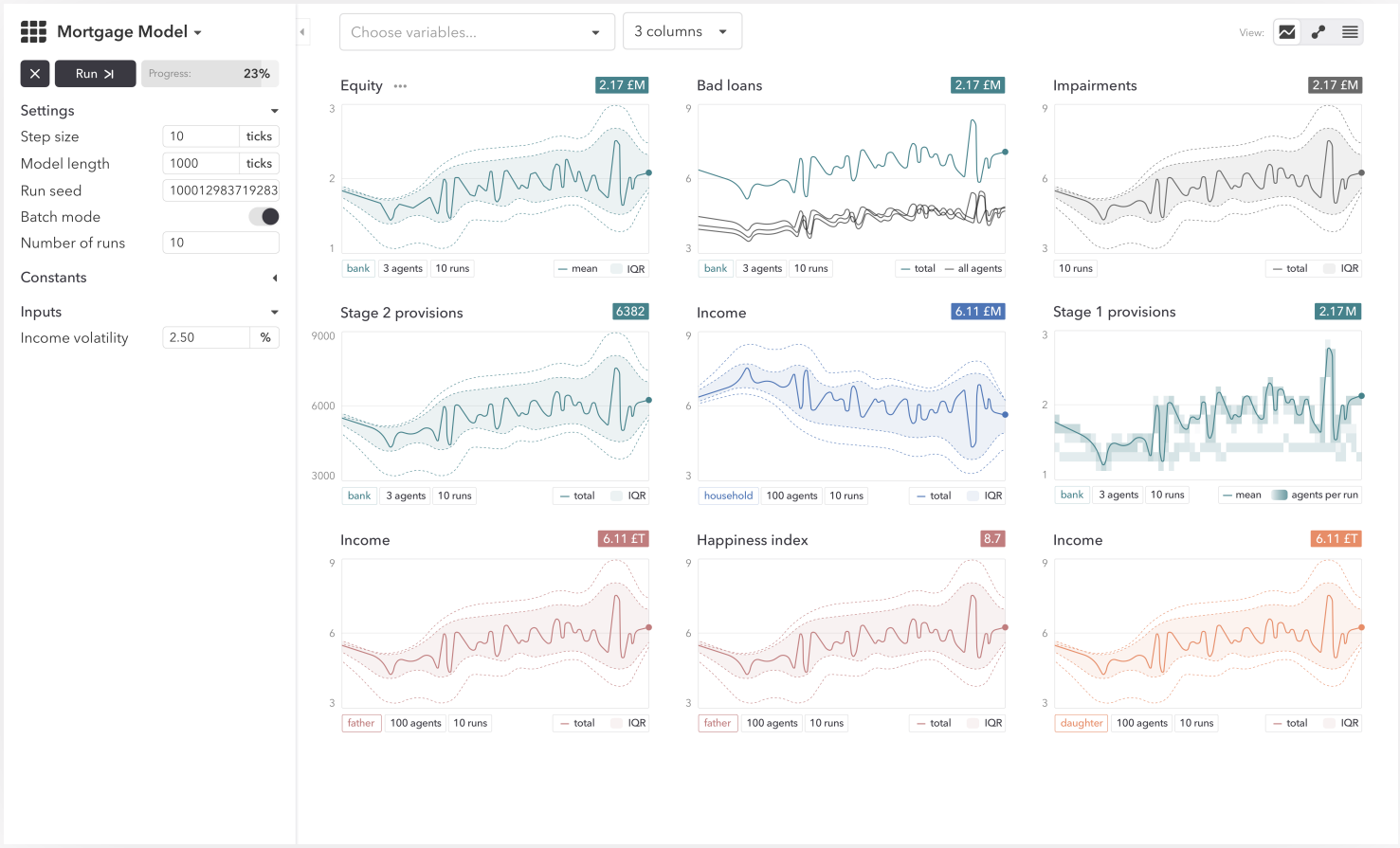
Two types of chart the user could choose from

The user is also able to show and hide tiles to choose the relevant information, and then to filter by results, attributes or agents.



Filtering charts

The final design included many features which were held back for future releases in order to be able to release something for users earlier. A process was designed to filter and prioritise future features as documented here.



Final design