The Computational Twin Engine:

POWERING INTELLIGENT DECISION MAKING

FACULTY

01. Introduction

The transition into the era of Artificial Intelligence (AI) will create huge opportunities to change the way organisations make decisions.

As more organisations make this transition, the deployment of AI to automate and enrich decision making will become increasingly necessary to remain competitive. But for organisations and society at large to actually realise the benefits of AI, maintaining human oversight, governance, and compliance, and ensuring decision making remains safe, connected and intrinsically human-centric will be essential.

However, today's approach to data and AI-led decision making has not provided what is needed to make the necessary shift to increasingly intelligent, connected, and governable decision making.

To address these challenges a new discipline known as Decision Intelligence (DI) has emerged that provides decision makers with the technology needed for making informed decisions. It allows them to understand:

- **What is happening?:** up to date information detailing the current situation, with the ability to dynamically cut the data into the appropriate context to understand what is driving the situation
- **What will happen?:** predictive technology to forecast likely outcomes, risks, or bottlenecks
- **What should I do about it?:** scenario analysis of potential actions, so that the quickest and most effective measures can be implemented to achieve organisational objectives, enabling decision makers to respond proactively to optimise their key business metrics

The Computational Twin Engine (CTE) is a general purpose technology for connecting AI models and implementing DI at scale. The CTE abstracts away significant complexity by combining cutting edge advancements in AI, simulation, optimisation, and causal inference, all framed around a decision-oriented method, into a unified package.

It provides the following four foundational technical capabilities:

1. **Simulation:** a powerful discrete-event simulator that connects AI models and provides outputs that are coherent, probabilistic, causally accurate, and explainable.
2. **Decision optimisation:** compare decisions in terms of downstream business KPIs, and a Bayesian optimisation module that allows end-users to find optimal scenarios, as well as allowing configurers of the CTE to shape reward functions for decisions in terms of downstream KPIs.
3. **Decision governance:** a governance framework allowing end-users to set bounds within which they expect the system to operate. The CTE can monitor violations of these bounds, influence decisions to try and remain within them, or block decisions that would cross them whilst escalating the decision to the end-user.
4. **Decision deployment:** integrate optimised decisions directly into the operational decision-making systems.

These capabilities allow for realistic interactive models of an organisation and for representing decision making in software. The CTE makes it possible for decisions to be coherently and globally optimised against meaningful downstream KPIs.

The architecture that powers the CTE is both modular and extensible, thereby providing the flexibility to serve a wide range of decisions without compromising on performance, scalability, or fault tolerance. There are three primary architectural components: an Entity Component System which provides an scalable pattern for managing simulation state, numerous CT Modules that handle system behaviour, and a Message Bus to transfer information during a simulation.

The CTE is a major advancement in state-of-the-art Decision Intelligence. In this document, we introduce concepts surrounding the CTE and detail its high-level capabilities, structure, and architecture.

02. Definition and concepts

02.1. Computational Twin Object Model

The Computational Twin (CT) Object Model is a comprehensive set of objects that provide the core building blocks for representing decisions in software. The objects come in two types: structural and simulation objects.

02.1.1. Structural objects

Structural objects make it possible to specify the full connected structure of a system:

Elements - units that flow around the system. Their flow through the system will be the key determinant of the organisations or system's performance. Elements can have nested data attached to them in the form of attributes made up of one or more properties.

Stages - collections of elements that share a certain state. Membership of a stage is defined in terms of a finite collection of boolean selectors on the attributes of elements. These selectors are combined conjunctionally with AND statements.

Resources - objects that are used by Elements while they are in certain stages: i.e whilst an element is in the associated stage the resource will be depleted by a certain amount.

Metrics - a quantitative description of an aspect of the system's state: for example the number of elements in a certain stage. Metrics should contain all of the business/system KPIs you care about when making decisions.

02.1.2. Simulation objects

Simulation objects govern the generation of events when a CT is run by the CTE. Events mutate the state of the CT by either creating/destroying elements, or adding/removing/amending attributes. During a simulation, synthetic events are generated and are then used to change the CT's state. This allows the simulation of possible futures subject to certain input parameters. Historical data can also be used to create events: running these in order through the CT can recreate what the system looked like at a particular point in the past.

The following objects generate events:

Processes - emulate a real-world part of the system which cannot necessarily be directly controlled. They are used to generate events for things we are not looking to optimise, either because we don't have control over them or we are just not interested in doing so.

Decisions - used to model something the organisation has control over and is looking to optimise. They are prescriptive deterministic models that generate events within the CT. Unlike processes, Decisions are architected to facilitate their deployment in the real world.

LLMs - integrates LLMs into the CT to automate routine business processes and generative tasks within the context of a wider decision optimisation. This leads to optimised decisions while ensuring adherence to governance and compliance policies.

Processes, Decisions, and LLMs can be arbitrarily complex, from simple rules to predictive probabilistic models, and can be coded from scratch, or simply be calling an external API.

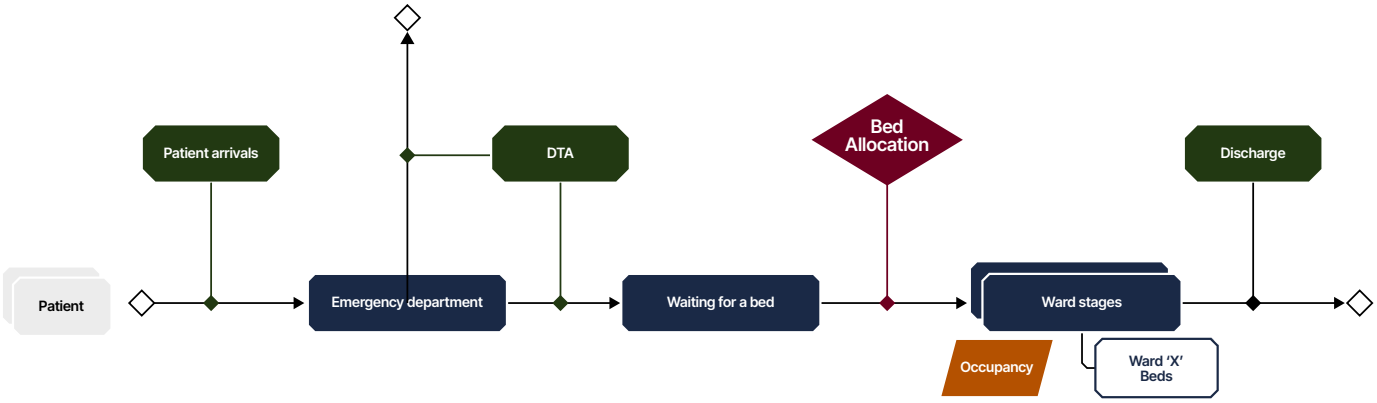
How they work is governed by Policies and Levers:

Levers - parameters which can be passed to any of the above three objects and used to modify their behaviour. These are often defined in the language of the decision, so an external user will understand what they do.

Policies - monitor the value of metrics by checking whether they are defined within boundaries. In historical mode these function like boolean metrics, but in simulation mode they can be used to regulate the outputs of decisions and LLMs, preventing events they generate from causing a policy violation.

02.2. An example: patient flow in a hospital

In this section we give an example of a patient's flow through a hospital. The CT diagram is below:



The objects in this example are defined as follows:

- **Elements:** patients with attributes, including their sex and their medical issue (a categorical variable)
- **Stages:** emergency department (where patients go on arrival), Waiting for a bed (patients who have been admitted and are waiting for a bed), Ward Stages (a collection of stages, each one representing a ward)
- **Resources:** ward "X" beds - the beds associated with ward "X"
- **Metrics:** occupancy (the number of people across all of the ward stages)
- **Processes:** patient arrivals (generates patients in line with historic admissions), Decision to admit (decides which patients to admit), Discharge (discharges patients according to attributes, and ward assignment)
- **Decisions:** bed allocation that decides which patients to allocate to a ward

03. Core capabilities

The CTE has four foundational technical capabilities, **(1)** Simulation, providing coherent, probabilistic, causally accurate, explainable outputs, **(2)** Decision Optimisation, **(3)** Decision Governance, and **(4)** Decision Deployment.

03.1. Simulation

The key benefit of the CTE is its ability to help users answer the following questions:

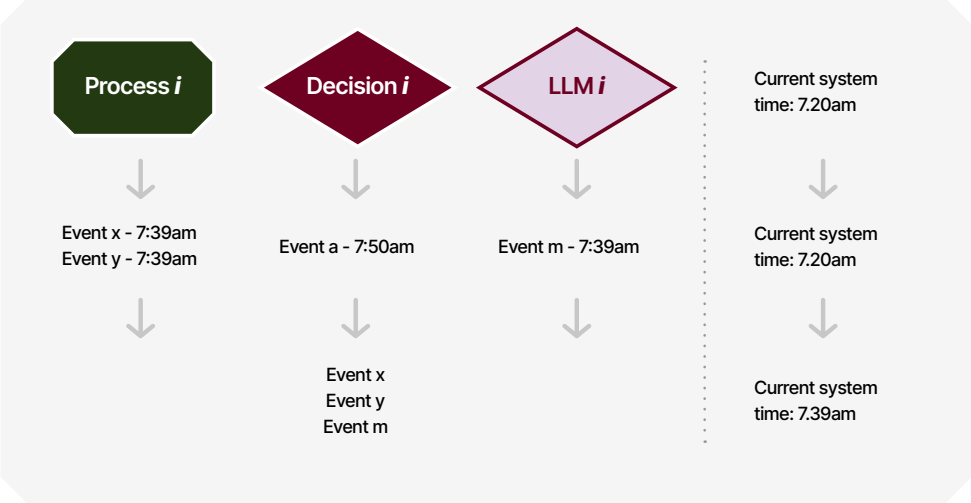
- What will likely happen in the future?
- Why did, or will something happen?
- What can I do about it?

These benefits are provided by the simulation mode of the CTE, which is a discrete-event simulator.
The way simulation mode runs is described by the diagram below.

Step 1: simulation objects are asked which events they want to generate next

Step 2: simulation objects output an ordered list of events they'd like to generate

Step 3: the earliest events are picked and then executed, with the CTE's internal clock advancing to this time



Note that simulation objects decide when and what events to generate based on their observations - these can be any part of the CT's state and are essentially analogous to features in machine learning.

The above diagram describes how the simulation logic in the CTE is able to evolve the CTE's state. One of these runs is called a trajectory, and it describes one possible future of the system being modelled. When the CTE is run in simulation mode multiple trajectories are in fact created, thus producing many possible futures for the system. How these are used is described in more detail in section 3.1.2.

What makes the CTE so valuable as a simulation tool for making business decisions is that its outputs are coherent, probabilistic, causally accurate, and explainable.

03.1.1. Coherent outputs

All metrics derived from CTE simulations are coherent with each other: i.e. any structural relationships between metrics are preserved. This is because all metrics are by construction functions of the same underlying data.The prediction of any metric at any time in the horizon is calculated from the data which makes up the CT state at that time, and possibly data from previous states.

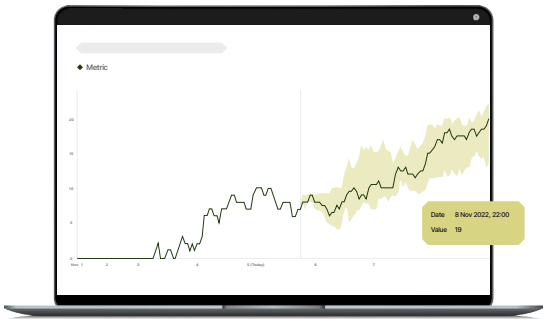
This data can be divided up into multiple levels of aggregation, following a tree-like structure. In the hospital example, the top level may refer to all patients, the subsequent level to patients in each ward, and the subsequent level to patients in each age group in each ward. Of course, data does not need to disaggregate in a unique hierarchical manner.

Since each metric is associated with a specific subset of the common upstream data, relationships between different metrics within the hierarchy are automatically preserved. In the above example, the prediction for the total number of patients is by construction the sum of the predictions for number of patients in the individual wards, since both refer to the same patient population.

03.1.2. Probabilistic outputs

A key advantage provided by CTE simulations is that it provides probabilistic predictions with expressed uncertainty. This is achieved by any process that has inherent uncertainty being modelled using probabilistic methods, often probabilistic ML models.

In each simulation the CTE propagates individual realisations of predictions from probabilistic processes through the system, thereby generating a single sample of a CT trajectory representing one possible future. Generating multiple such CT trajectories allows the user to access the distribution of any metric at a given point over the possible futures. Specifically, it allows us to calculate statistics over metrics, e.g. histograms, credibility bands, error bands, or point estimates of most likely outcomes.



The picture above shows a probabilistic forecast from the CTE. The space to the left of the vertical dotted line represents the past, whereas the space to the right represents the future. The shaded confidence interval shows a range of possible outcomes.

There are two benefits of the CTE's probabilistic forecasts to a business user:

- The CTE doesn't show just the most likely outcome, it shows a range of possible outcomes. This means that users can produce plans which take into account the uncertainty within the system, and have the ability to compute the risk of something unwanted happening.
- Even if a user is only ultimately interested in an expected outcome, having access to probabilistic models for the underlying processes is crucial for calculating this accurately. Non-probabilistic ML algorithms are often optimised using maximum likelihood estimation, but if a chain of these models simply passed its estimate downstream, its composite generally would not be the maximum likelihood estimate of the whole chain.

03.1.3. Causally accurate outputs

The CTE allows the user to specify causal representations of the underlying system being modelled. Any simulation outputs are faithful to the specified cause-and-effect relationships.

When run in simulation mode, the CTE is fundamentally a data generating process, modelled according to the substructure of the business process in question. As such, any data generated by the CTE – whether a simulation of the future, a playback of the past, or a hypothetical alternative history – has an inherent causal structure that is implicitly contained within the CTE program that generated it.

Rather than having to learn the system's causal structure from data, and being left at the mercy of any spurious correlations that may lie within it, it derives much of its causal structure from its definition.

This gives the CT two big advantages compared to isolated point ML models:

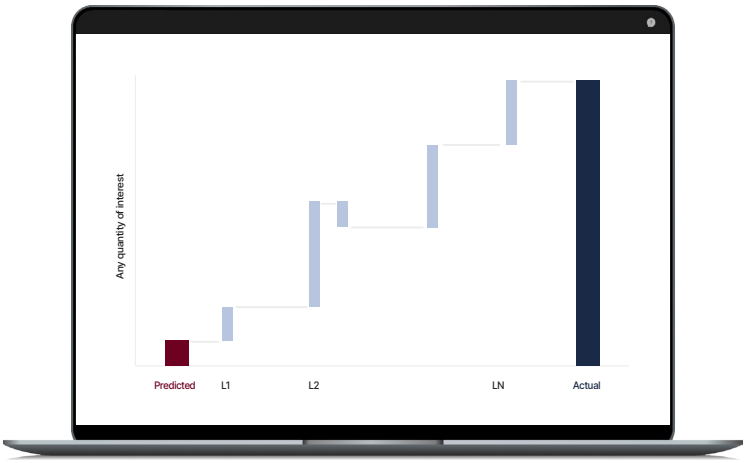
- **Generalisability:** robust inference is possible even for unseen scenarios since their downstream effects can be derived by accessing the cause-and-effect relations encoded in the CTE. In the hospital example, the CTE would be able to simulate the effect of changing the number of beds available in the hospital, or the effect of a different bed allocation strategy. An attempt to enable the former via an ML approach would be to expose available beds as model features, though this approach would be vulnerable to only being able to produce reliable outputs for bed configurations that had been observed previously. In the latter case of changing bed allocation strategy, traditional ML cannot cope with this at all as the ML can only (at best) learn the previously used bed allocation strategy from data.
- **Data efficiency:** having access to causal relationships from its definition allows for better predictions at the same volume of data by adding the right causal knowledge (i.e. inductive bias). In the hospital example, imagine our goal is to predict how many patients have to be deferred to other hospitals, like in an emergency situation due to full occupancy. If this either has not or has rarely happened before, an isolated ML model trained on historical data is unlikely to predict deferred patients accurately; the CT, however, has the right causal knowledge, knowing that any patient arriving during full occupancy is a deferred patient.

03.1.4. Explainable outputs

Outputs of the CT are explainable. The CT offers two methods which enable explanations of outputs: the first one is intrinsic to metrics of the CT, while the other is a dedicated root cause analysis technology.

The first is **explainability through metrics queries**. The CT enables filtering metrics on various properties such as metric type, tags, time, or sub-populations, which provides insight on what makes up unexpected changes in KPIs. Since each filter with respect to a specific attribute is associated with a unique subset of the common upstream data which makes up a CT trajectory, it is possible to understand which subset is associated with an unusual behaviour.

The second is **explainability through Shapley values**. The CT provides technology that can explain discrepancies between CT simulations and reference outcomes, e.g. from a real-world actual past or other simulations. The technology is enabled by Asymmetric Shapley values (ASVs), an explainability framework developed by Faculty which incorporates causality into explanations of machine learning models [Frye et al., 2020].



The Shapley value technology in the CTE can explain which time intervals or processes are responsible for discrepancies between CT simulations and reference outcomes.

The CTE can utilise Shapley value-based explainability techniques for two different use cases. The first is Time-ASVs. These allow a user to understand when discrepancy in the KPI of interest between simulation and reference is generated, with a temporal resolution set by the user. This is achieved by dividing the simulation time window up into time steps (specified by the user) and running simulations from each time step, starting from the corresponding reference state, to the end of the simulation horizon, and deriving the KPI for each resulting trajectory. The difference between KPIs associated to two subsequent time steps states how much discrepancy in simulation vs. reference is due to events happening between these two time periods.

Process-SVs provide insight into which processes generated the discrepancy in the KPI of interest between simulation and reference. This technology is enabled by considering the processes as players in the classic Shapley value paradigm, that is, the contribution of each process is calculated as if in a cooperative game by considering the average marginal contributions across all possible process permutations.

03.2. Decision optimisation

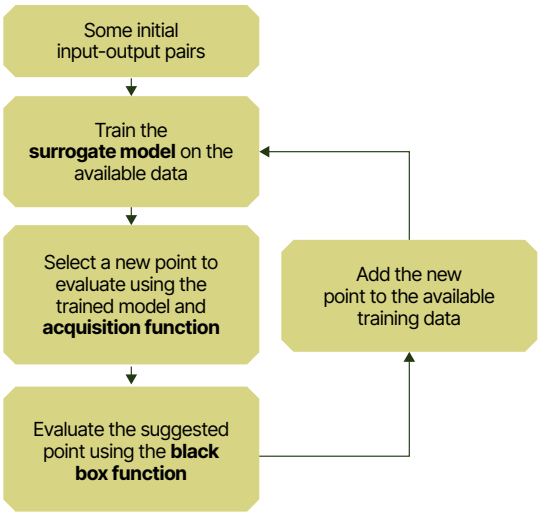
The CTE is designed to help find more optimal decisions. Optimisation techniques are used to determine the best decisions to be performed within the system, given the data available at decision time and the projected downstream KPIs that decisions should be tuned to optimise.

03.2.1. Quantifying the downstream effect of decisions

Part of the CTE's benefit lies in its ability to understand the impact of (locally optimal) decisions on the wider system in terms of downstream KPIs. For example, a particular decision in the CTE might simply be a greedy optimisation algorithm which, each time it is called, makes a decision that is optimal at the moment in time based on the available input parameters and the specified cost function. This would still be possible without the CTE, but by plugging this into the CTE a user can understand what effect this will have on the wider system rather than just in terms of the locally-defined cost function. Additionally if this decision has configurable input parameters (e.g. levers, resources) that are controllable, then a decision can be made on which values to pick based on the effect on the downstream business KPIs. This means the CTE is able to make decisions more globally optimal, by allowing an end-user to customise them with respect to the KPIs that are important at a system level.

03.2.2. Simulation-based optimisation with the CTE

The CTE has the capability to do simulation-based optimisation, specifically Bayesian Optimisation. Bayesian Optimisation is a sequential model-based optimisation technique that leverages Bayesian statistics to efficiently search for the optimal solution within a given parameter space. It builds a probabilistic surrogate model of the objective function, iteratively suggesting new points for evaluation based on the model's predictions, ultimately converging to the best possible outcome.



The CTE uses this for:

- **Reward-shaping for greedy-optimal decision making:** one can define a greedy reward function to be a linear combination of metrics that can each be computed locally in terms of the observation and candidate action. Each of these metrics is multiplied by a coefficient that can be considered a hyperparameter of the optimisation algorithm. Bayesian optimisation is then used to tune these hyperparameters under CTE simulations to determine which values of the hyperparameters will lead to the most preferable downstream KPIs. This offers something in between basic greedy optimisation and full longer-term (i.e. downstream outcome) optimisation. In particular, it offers the safety, transparency, and efficiency of greedy optimisation along with the performance upside of longer term optimisation.
- **Optimal scenario discovery:** the Bayesian optimisation module can find the optimal input parameters for a CT with respect to some downstream KPI, for example finding the optimal lever values or capacity of resources. When these input parameters are something that the organisation controls, this can allow decision managers to find which scenarios are (more) globally optimal for their organisation.

03.3. Decision governance

Another core capability of the CTE is its ability to govern decisions. This capability is critical to meet the needs of decision intelligence at scale, and is the core function of the policy object.

In the CTE, the Policy object is designed to observe if the values of metrics are <, ≤, >, ≥ a certain value. They have two modes of operating:

- Policies have the ability to **monitor** metrics; essentially in this mode policies act as a metric, where the metric in question is a time series of binary variables. Any breaches of policies can then be flagged to end users.
- Policies are able to **regulate** the output of decisions and LLMs, i.e. to potentially alter the events they generate if these events will cause a policy to be breached. In practice, the CTE supports this through two types of policy regulation:
  - **Influence:** here, policies acting in regulation mode are passed to the decision. It is then up to the engineer defining the CT to decide how the policy regulation should work by including it in the decision function logic.
  - **Enforce:** in this mode, events emitted by a decision are checked by a policy which is acting in regulation mode. If the events cause the policy to be violated then the events are not triggered, and in the case of a deployed decision it is then escalated to an end-user.

One common way to implement the influence type of policy regulation is by passing the policy as a constraint to the optimisation algorithm in the decision function. Doing this ensures that the decision function will only produce events that don't violate the policy. Allowing flexibility in terms of how policies regulate decisions ensures that this regulation is sensitive to the specific problem at hand.

03.4. Decision deployment

The CTE provides powerful technology to integrate predictive models into a powerful causal simulation framework that helps organisations get a deep understanding of what's happened up to now, why, what's likely to happen, and what is best to do to achieve their objectives. However, improving decisions requires that any optimised decisions are integrated directly into the operational decision making systems. The CTE provides this capability.

A Decision in the CT Definition is a deployable object that informs an action in the system being considered. In general, a Decision can contain any kind of algorithm, from ML and optimisation algorithms, to simple rules. Every Decision developed using the CT Object Model is deployable independently of the CTE and can be integrated into an external system as a Decision Point. Deployed decisions can be implemented through either:

- **Decision Automation:** enacting the action suggested automatically, or
- **Decision Augmentation:** suggesting action(s) to be taken that are executed by a human operator.

The CTE exposes a function that allows for a Decision Object to be exported as a Decision Point. To ensure the same decision logic is used during simulation, decision optimisation, and when deployed, the decision is defined in the CT Definition in two independent code locations:

- **Decision function:** the logic and model that encodes the decision algorithm itself. It has arbitrary inputs and outputs, and is used both in simulation and in production.
- **Decision wrapper:** an adaptor around the Decision Function that allows it to operate within the context of a CTE simulation. It translates the CT state into the language of the decision function.

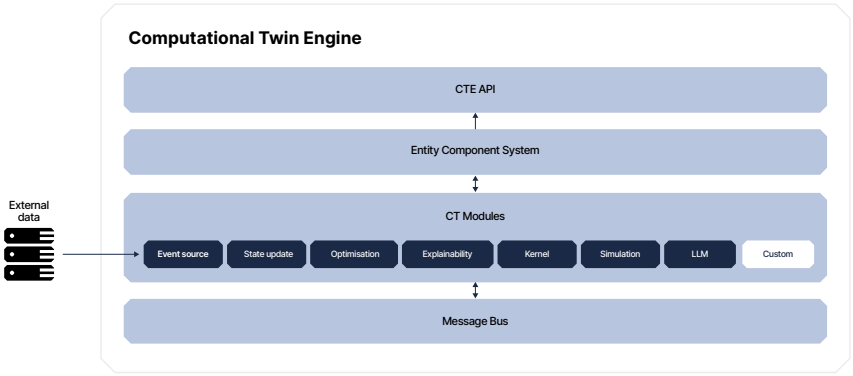
The CTE Decision Point deployment system is designed with the flexibility required to be able to integrate with any operational system. In addition to the Decision Function, a Decision Point includes:

- The supporting **Lever and Policy values** that make up the optimised Decision
- A **Monitoring Wrapper** that wraps the Decision Function, sending each decision to a Monitoring system
- **Adaptors** for operational systems and Decision Functions. These are custom code made up of three parts:
  - A data ingestion function that reads the current state of the system from operational databases or from an HTTP request, and transforms it into a format acceptable to the Decision Function
  - A result serving function that takes the output of the Decision Function and transforms it to something that makes sense for the operational system it is deployed into
  - Custom infrastructure where the deployed Decision Point runs

Whilst the specifics of decision deployment are custom to the operational system, the CTE allows for the logic and models that underpin the decision-making process to be shared between production systems and the CTE simulation, ensuring there is coherence between decision planning and decision execution.

04. CTE architecture

The architecture of the Computational Twin Engine is designed to be both modular and **extensible,** thereby providing the **flexibility** to serve a wide range of decisions without compromising on **performance, scalability, or fault tolerance.**



There are **three** primary architectural components:

01.

Entity Component System (ECS): serves as the foundational piece that governs the representation of the entire system. ECS is a software architectural pattern predominantly used in complex system simulations and real-time computer games.

An ECS architecture consists of three primary elements:

- **Entities**: a thin container for objects that populate your system. Every top level object in a CT definition are represented by an entity in the state.
- **Components:** Collections of data and logic attached to an entity, designed to process, store and update information as the system evolves.
- **Systems:** higher level functionality that instantiates and orchestrates runs of "trajectories" of the overall system, for example evolving the state of a given CT in historical mode over some time window.

02.

CT Modules, handling different aspects of system functionality and CT behaviour. Some of these manage core functionalities like data ingestion, state management, and orchestration of CTs, with others focussing on the specific behaviours, simulations, and decision-making capabilities associated with individual CTs. Important examples of modules include the kernel, which serves as the central management unit of the CTE, and the event source, which is responsible for ingesting data into the CTE. Custom modules can be developed independently and plugged into the CTE.

03.

Message Bus: the mechanism through which information is passed between parts of the CTE, and between modules. This serves as the communication channel within the CTE, allowing different modules to exchange information and events without being directly connected to each other. It acts as a centralised broker that routes messages between the senders and the receivers.

To interface with the CTE there are two main APIs, python and HTTP. The python API exposes functions that can be called from other python code, and the HTTP API provides RESTful endpoints for the deployment, management, and monitoring of Computational Twins.

By adopting this architectural pattern, the CTE is able to tackle complex simulations and optimisations while maintaining high performance, flexibility, and extensibility, making it a groundbreaking tool in the AI and Decision Intelligence space.