Representing Lifecycles of Entities using States + SSI

Neil Smyth, Cherrytwist, v0.1

The goal of Cherrytwist platform is to facilitate the collaboration between multiple stakeholders as they interact within the context of complex goals. The top level context for interactions are Challenges, which can be viewed as a goal / target that is being aimed for even if there is uncertainty in the exact path to get there, as well as who could contribute. At the lowest level there are Projects, which are formal agreements between parties as to work to be done to make progress. Challenges + Projects are examples of Entities.

State based representations are widely used, with Finite State Machines (FSMs) being a formalism whereby a system can be in exactly one from a defined set of states, with clearly defined criteria / rules for when a transition can take place. Examples include Kanban boards, RFCs as well as more formally business process modeling approaches.

A state based representation is a natural match for collaboration on complex goals – allowing clarity for the multiple parties interacting as it is always clear where the Entity is (exactly one state) as well as the allowed transitions with guards that determine how an Entity moves between the potential States.

Use Case

Each Entity on the platform, such as a Challenge or a Project, is assigned a digital identity. It in effect becomes its own agent, with the controller being (for now) the platform. It potentially later has resources associated with it.

Each User interacting with the platform also has a digital identity. Note that a user does not need to be a single individual but could for example represent an organization. Logically it is a single actor.

Each Entity has a Lifecycle associated with it, which is a set of states together with the transitions between those states i.e. it is a FSM representation. The Lifecycle of an entity is defined at the moment it is created (for now assume immutable but migrations can be explored). The Lifecycle Manager is a separate component that manages Lifecycles.

In a decentralized platform, each Entity needs to be able to independently determine which actions are allowed. Centralized authorization is simply not feasible. The obvious example is Authorization, but this can be made more generic.

Taking the Authorisation use case and the usage of Verified Credentials for this:

- The Platform issues a VC to a User (Holder) for a particular Entity (Issuer) stating that the User is allowed to make certain State changes.
- The Lifecycle Manager (verifier) receives the request to transition between states.
- The User is challenged to provide the appropriate VC, presents the relevant VC, which is verified by the Lifecycle Manager and if it is valid then executes the action.