Why generalized request?
Over the years, there are other kinds of requests that are currently handled through e-mails and human interactions. It is highly desirable to incorporate them into PhEDEx. In current implementation, to serve a new kind of request means significant development effort. Generalized Request project is set to provide a framework so that future request types could be easily implemented, including the complex approval process, in which, multiple approval parties are involved in certain logical relationship.

Generalization is abstraction and realization
The concept of generalization is to abstract current processes to their conceptual level which does not depend on their essences. That is, a "generic request" that is not associated with any current "types", and, as a set of decisions that deal with it. This generic request is general enough to accommodate current and future requests. Then, attach type specific information to individualize it. In the end, to create service to a new type of request is to "define" it to the system. It will be more an incremental configuration change rather than a code update.

States rather than steps
For a generalized request, we have to think not from the procedures that need to be done for a particular request but from the stages of its general life cycle and the actions that move it from one stage to another. We define the states of a general request and it is realized by a deterministic finite state automaton, DFA. The next state is a function of current state and action taking place. Therefore, the control flow, the business logic, is implied by this DFA. Not all requests go through this DFA the same way, but the mechanism is the same.

Roles are the players
A "role" is an entity that acts upon the requests. In general, it is, but not necessarily, a person with certain privilege, such as site data manager, group manager, global admin and so on. In some cases, it could be a process, acting like a person, that executes predefined actions based on predefined conditions.

Complex approval process
An approval process could be as simple as a single decision by a role or as complex as involving many decisions from many roles, some of whom are required to make decisions and some of whom only need to do if none of their peers made one. For each role, the decision could be in one of three values: approved, denied, or undecided. If we relax the dependency among the decisions, all approvals can be represented by an approval plan, consisting of logical expressions. The current status of an approval can be determined by evaluating its corresponding logical expression using current values of its elements.

- A and B – need both roles A and B to approve
- A or B – either A or B can approve
- A or (B and C) – If A approves, it is approved. If A denies, it would take both B and C to approve it.
- A and (B or C) and (D or E)

Recommendations
- Represented by as to do it if none of their peers made one. A request has own life cycle, including creation, approval, notification, and book keeping and the details depend on its type. Currently, only two kinds of requests, transfer and deletion, are fully integrated in PhEDEx. They are tailored specifically to the operations workflows. To be able to serve a new type of request it generally means a fair amount of development work.

After several years of operation, we have gathered enough experience to rethink the request handling in PhEDEx. Generalized Request Project is set to abstract such experience and come up with a request system which is not tied into current workflwo yet it is general enough to accommodate current and future requests. The challenges are dealing with different stages in a request’s life cycle, complexity of approval process and complexity of the ability and authority associated with each role in the context of the request.

Why generalized request?
We start with a high level abstraction driven by a deterministic finite automata, followed by a formal description and handling of approval process, followed by a set of tools that make such system friendly to the users. As long as we have a formal way to describe the life of a request and a mechanism to systematically handle it, to server a new kind of request is merely a configuration issue, adding the description of the new request type, rather than development effort.

References