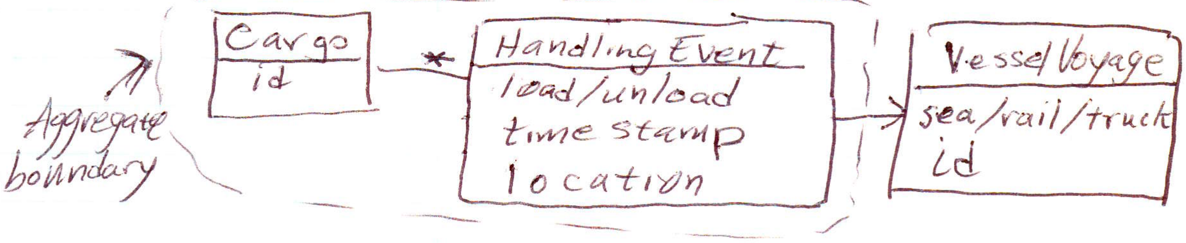
NOTES ON DDD SAMPLE APP — ERIC EVANS, 2007-12-06

STAGE 1: CARGO TRACKING

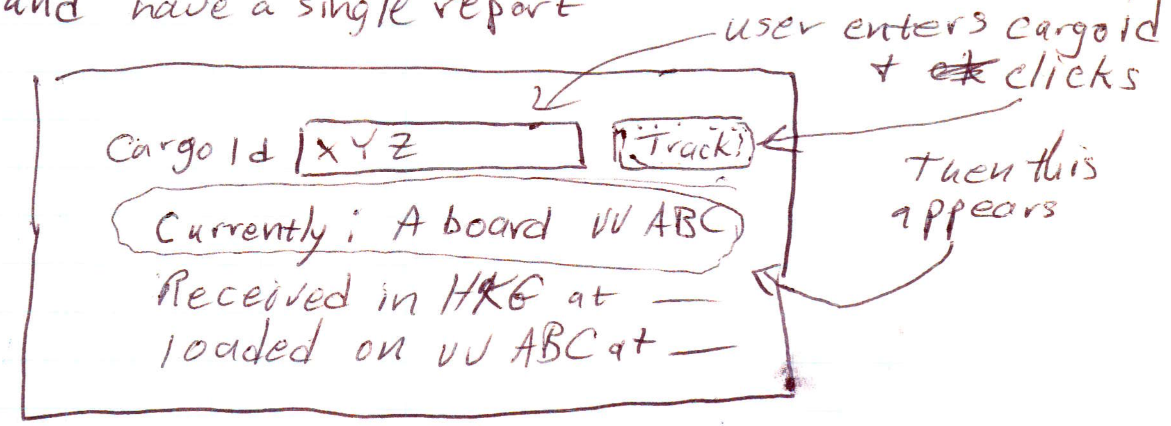
Feed in Domain Events, such as

CargoXYZ received in HKG at 18:15 2007-10-5
CargoXYZ loaded on vessel voyage ABC in HKG at timestamp
" unloaded from " " in LGB at timestamp
" loaded on vessel voyage DEF in LGB " "
~~"rail~~ ~~type "rail"~~
" unloaded from " in ~~DAL~~ ~~LGB~~ DAL " "
claimed in DAL at timestamp

Except there would be at least 2 cargos whose events are being fed to the system, and time-stamps can be slightly out of order, (because of ~~late~~ simulated latency or operational delays in entering data.) Some steps might be missing.
structure would be something like this:



So, the first version of the app could receive such events and have a single report

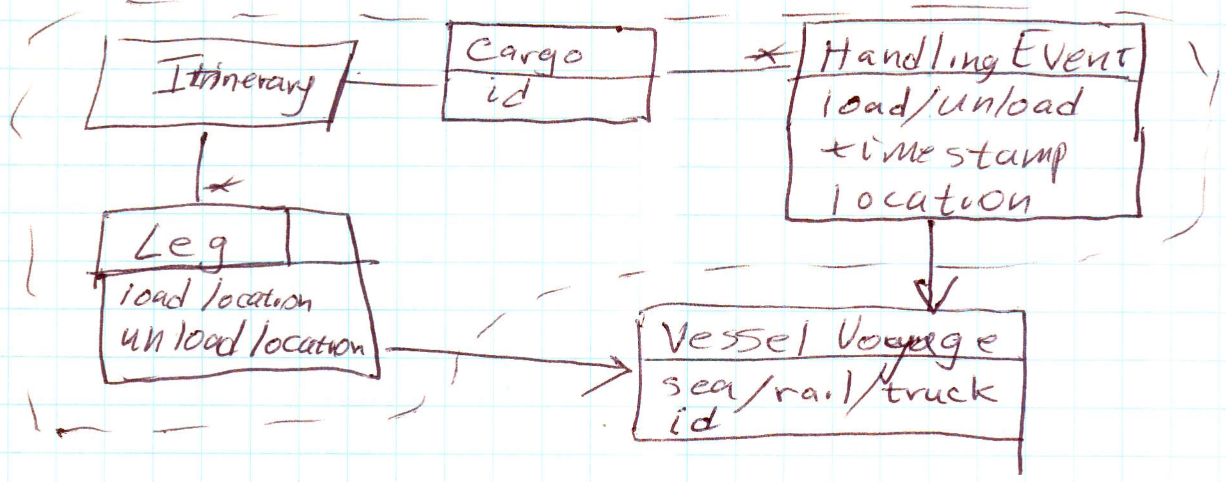


This ~~part~~ Stage would demonstrate:

- Domain Events
- Asynch external data feed (with tranlation from other model)
- UI reflecting model in non-trivial way
- O-R mapping of entity + domain events
- Architectural layers (UI, App, domain) + Technology stack.
- Aggregate (simple)

STAGE 2: Tracking against itinerary

Structure:

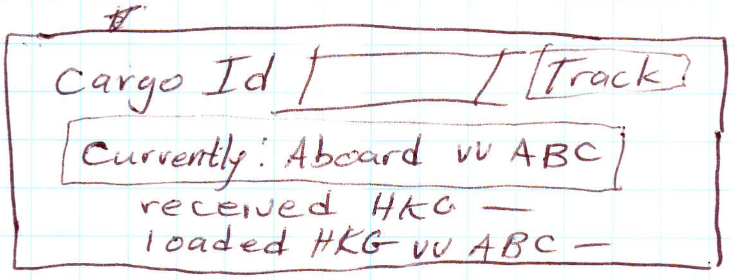


Attach an Itinerary like this

Leg1: VesselVoyage ABC, load HKG, unload LGB
Leg2: Vessel Voyage DEF, load LGB, unload DAL

Feed Handling Events as before:

Case 1: received HKG —
loaded HKG onto VV ABC

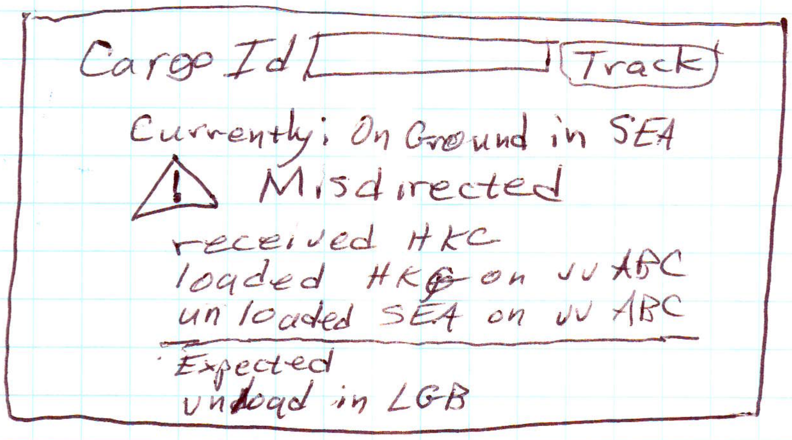


Cargo Id [ ] [Track]

Currently: Aboard VV ABC

received HKG —
loaded HKG VV ABC —

Case 2: received HKG
loaded HKG on VV ABC
unloaded SEA from VV ABC



Cargo Id [ ] [Track]

Currently: On Ground in SEA

⚠ Misdirected

received HKG
loaded HKG on VV ABC
unloaded SEA on VV ABC

Expected
unload in LGB

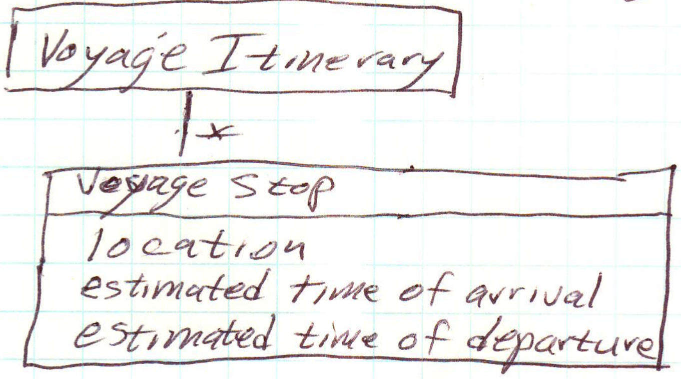
STAGE 2b: Notification

Trigger a notification when:
- Cargo is unloaded at destination
- Cargo is misdirected (when misdirection is detected)

STAGE 2c: Advance Notification

Trigger a notification:
- 24 hours before cargo arrives at destination.

For this, we need a vessel voyage itinerary.



This stage will show:

- Complex Value Objects (eg Itinerary)
- Aggregate states derived from multiple objects ~~(eg misdirec~~ (eg "misdirected" + "next expected")
- ~~Stage I [illegible]~~
- System/App actions in response to domain state (STAGE 2B)
- Tricky system/domain state combinations (STAGE 2C)

Oh! And an important part of this:

- O-R mapping of complex value objects.

~~STAGE 3 could involve~~

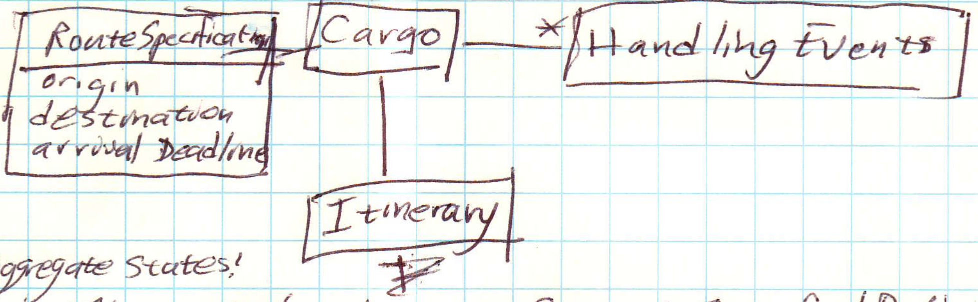
NOTES ON DDD SAMPLE APP — Eric Evans, 2007-12-06

STAGE 3: CARGO EDITOR + REROUTING

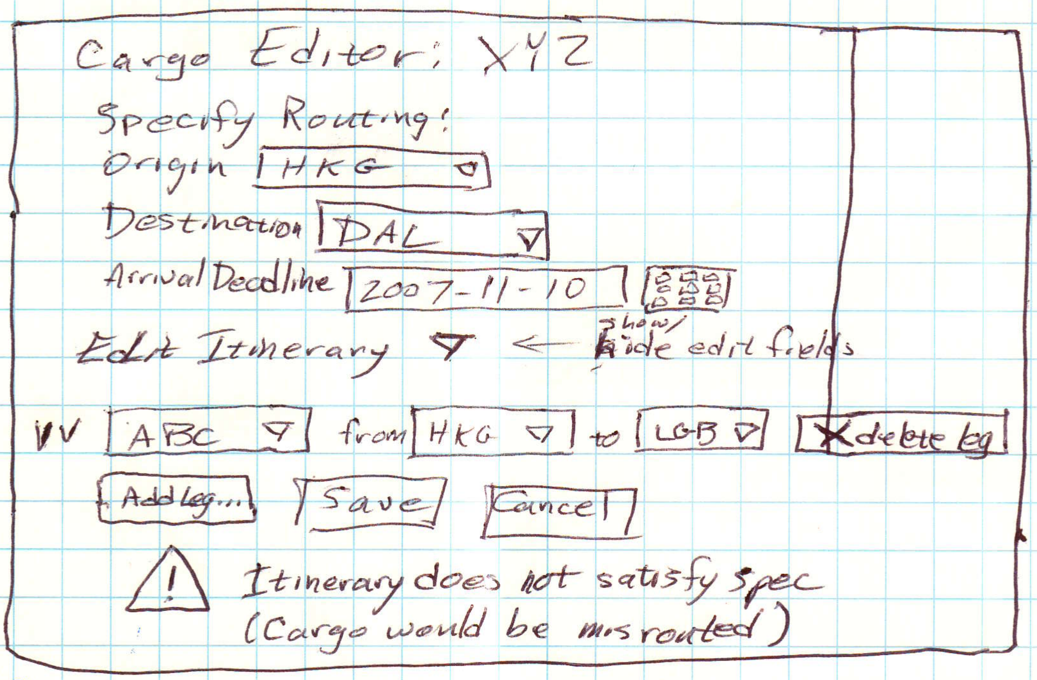
We can fill-in the details on this when/if we get this far.

Provide a database of vessel voyages (just additional instances ~~of~~ of what was done in 2c)

Structure of Cargo Aggregate



Aggregate States:
Misrouted = ! routeSpec.isSatisfiedBy(itinerary)
Misdirected = ! handlingEvents conform to Itinerary



Cargo Editor: XYZ

Specify Routing:
Origin [HKG ▽]
Destination [DAL ▽]
Arrival Deadline [2007-11-10] [calendar]

Edit Itinerary ▽ ← show/hide edit fields

VV [ABC ▽] from [HKG ▽] to [LGB ▽] [X delete leg]

[Add leg...] [Save] [Cancel]

⚠ Itinerary does not satisfy spec (Cargo would be misrouted)

...You get the idea.

This stage demonstrates.
- Connecting an editor to a complex object model with feed back from model.
- Saving updated state: Entity, Value Objects + respecting Aggregate boundaries