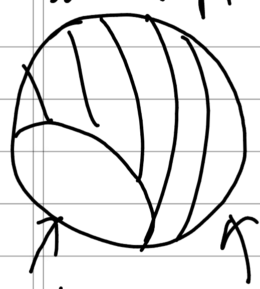
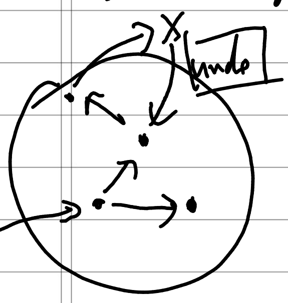
2. State graph obscurity

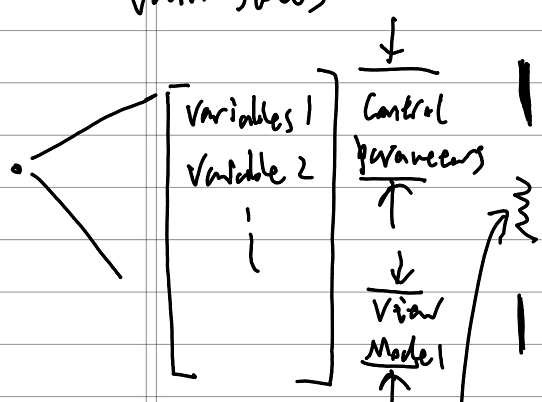
1. State Graph Presentation

State Graph (static)





Valid states



State vector (all state variables)

3. State Refinement & Invalid state handling

(dynamic) User has no prior knowledge about the validity of states before encounterment.

program | Human

working memory

o means partial info about the system

1. State graph presentation ⟷ | short-term memory / working

① visibility of system status

⑥ recognition rather than recall (not too less)

⑧ aesthetic and minimalist design (not too much)

'X' | state | name / validity
ST

indexing: classification

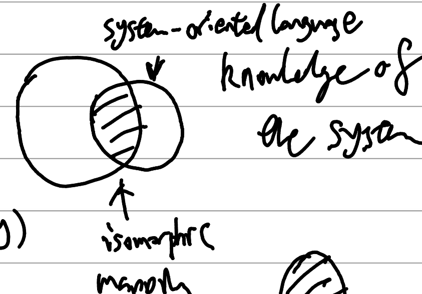
2. (State graph indexing) ④ ⟷ ② Long term memory

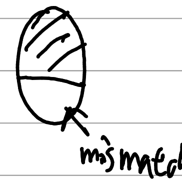
② match between system and the real world

⑦ → flexibility and efficiency of use (customization at runtime)

| encoding: knowledge of real world (different domains have different languages)
| convention (systems with years' history)
i.e. user base been trained

* mapping between language spaces

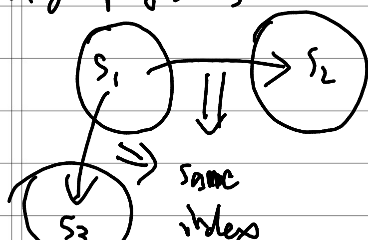




④ consistency and standards.

e.g. namespace collisions (most of time, not a problem)

i) a group of states



ii) similar state vector presentation & similar values in these state variables present.

3. State refinement, Invalid state handling at runtime

③ User control and freedom
* control

⑤ error prevention (static state graph)
prevent user from getting into invalid states
* control

Concurrent system

① help users recognize, diagnose, and recover from errors
* error message / state indicator
Design choice: (static state graph)

more: | toward expected valid states
e.g. action suggestions
back to previous valid states in history
e.g. redo/undo, state history

missing: (dynamic state graph)
- (checking of state validity at runtime
- more

4. documentation

(b) documentation