6894 · workshop in software design

team A · october 7, 1998 · reverse engineering

john chapin, jay corbett, daniel jackson

daniel jackson
mit lab for computer science

summary

what we investigated

- assignment of aircraft to RAs
- how aircraft enter and leave the CM

what we produced

- informal descriptions
  - narrative, filtered call graphs
- an object model and an ELH model
- abstraction function: how code maps to models
- some anomalies and lots of questions

how we did it

- used Lackwit to find sites that access relevant fields of aircraft tree record
  - ac->active, ac->landed, ac->ra_index
- constructed call graphs using
  - team B's tools, for local details
  - Lackwit, for showing all funs that pass ac structure around
- … read code

anomalies

- states of RA and CM not apparently always in sync
  - messages sent but aircraft tree record not updated
    - new_category_set_ra_owner sends SET_RA_OWNERSHIP
    - update_active_ac_specific data sends AIRCRAFT_INACTIVE
    - check_entry_to_landed_zone sends AIRCRAFT_INACTIVE
  - no explicit initializations of fields
    - ac->active, ac->landed
- incorrect documentation
    - send_category_amendment called by ism_handle_fp_amd
    - sets ra->index to 0
- functionality not accounted for
  - handoff_aircraft_to_ra seems to assume that RA and CM not in sync
    - broadcasts UNSET_RA_OWNERSHIP when ac->ra_index is NOT_SET

ELH

some sample roles

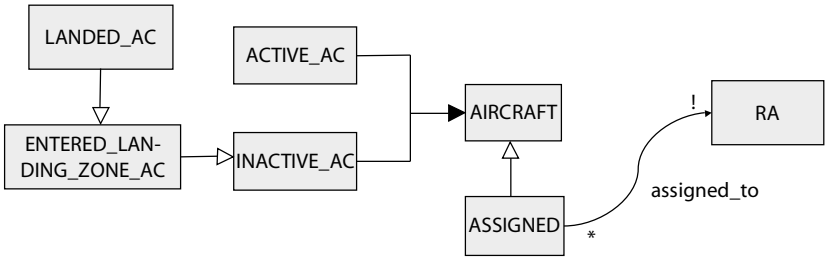
- see report for others
- very unsure about these!

- // cannot become inactive unless previously active
- (become_active become_inactive)* [become_active]

- // cannot assign until active
- [become_active assign (unassign assign)* [unassign]]

- //no significant events while unassigned
- ((unassign assign) | fp_amend | become_inactive)*

object model



questions

- what does active mean?
- freeze events?
- ignoring aircraft outside center?
- fp amendment within landing zone?
- category changes

what we learnt (1)

where tools helped

- Lackwit
  - + finding accesses to ac
  - + small call graph (90 funs, 10 shaded funs, 5 globals)
  - - missed accesses thru typecasts (eg, load_buffer)
  - - bugs, esp. static funs
- Team B tools
  - + answering queries about callers/ees of funs
  - + file dependences
  - - big fun call graphs
- grep, etc
  - + finding message sends (kind depends on use of constant)
  - - tons of chaff

what we learnt (2)

good models are hard to extract

- ELHs
  - order is highly data-dependent; need global analysis
  - identifying events is sometimes impossible (eg, freeze)?
- OMs
  - basic structure easy: from aircraft tree
  - abstract invariants hard, because abstraction function not given
    - eg, are landed aircraft inactive?
- would save a lot of work if could ignore the code
  - construct models by interviewing designers

construction of models raises hardest questions

- what exactly is active?
- category amendments?
- filtering of fps
  - ignore add if out of range but then consider later?