Black & White, Never Grey:

On Interfaces, Synchronization, Pragmatics, and Responsibilities

Franz Puntigam

Vienna University of Technology
Vienna, Austria

franz@complang.tuwien.ac.at

Black & White versus Grey

Grey: Dependence on unstable parts $\Rightarrow$ not substitutable

Black & White: Well-specified stable parts (interfaces) and hidden unstable implementation details $\Rightarrow$ substitutable

Practical problems:

- Which parts are stable in which aspects?
- How to specify stable parts?
- How to ensure conformance?

Interface Specification

Pragmatic spec. = informal description of relevant aspects

- simple, cheap, and powerful solution for usual cases
- ambiguous and incomprehensible in unusual cases

Formal specification with tool support for specific aspects

- use of tools is expensive and requires expert knowledge
- complete specifications turn black into white

Dividing / moving responsibilities from component to user

- avoid unnecessary exposion of implementation details
- for many aspects not explored sufficiently

Specifying Synchronization in Interfaces

- Users need information to avoid synchronization conflicts.
- Complete synchronization info turns black into white.
  Partial synchronization info insufficient to avoid conflicts.
  (Only simple mutual exclusion cannot cause conflicts.)
- Usual synchronization patterns are well-known (buffer . . .).
  Documentation mentions exceptional cases (overflow . . .).
- Unusual cases: Give complete synchronization information or describe all possible conflicts (huge number).

Required Synchronization

- Required sync = sync a component requires from a user
  Example: "init" must be invoked before any other routine
- Required sync is specified only in the interface.
- Users obey restrictions on invocation sequences.
- Multiple users must be coordinated to ensure linearity:
  - only a single user $\Rightarrow$ simple, no tool support needed
  - splitting/combining linear threads $\Rightarrow$ requires tools
- Invocation restrictions are white, details remain black.

Required Synchronization: Example

```
interface WindowIcon {
   // abstract states: icon, noIcon
   void iconify(int);   // when noIcon; next state: icon
   void uniconify(int); // when icon; next state: noIcon
}
...
   WindowIcon w = ...;  // w not shared; state: noIcon
   while (true) {
      for (i=0; pressedButton() == false; i++) wait(...);
      w.iconify(i);
      for (i=0; pressedButton() == false; i++) wait(...);
      w.uniconify(i);
   }
```

How to Avoid Grey

Usual cases: Use spec based on common understanding.

Describe all deviations from usual solution(s).

Unusual cases: Use a technique to divide responsibilities and move them (partially) to clients.

Some techniques are very specific and not widely used.

Turning black to white can be an intermediate step when developing appropriate tools and techniques.

Conclusions

- Pragmatic interface specifications are of great importance.
- We need techniques to divide and move responsibilities (more than those to specify aspects in interfaces).
- Desire of white and grey boxes = sign of bad factorization.