



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Sviluppo software in gruppi di lavoro complessi¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Anno accademico 2024/25, II semestre

¹ © 2025 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>

1



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Lezione XVII: *Design by Contract* con Eiffel

131



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Contratti ed ereditarietà

Il *principio di sostituzione di Liskov* stabilisce che, perché un oggetto di una classe derivata soddisfi la relazione *is-a*, ogni suo metodo:

- deve essere accessibile a pre-condizioni uguali o più deboli del metodo della superclasse;
- deve garantire post-condizioni uguali o più forti del metodo della superclasse;

Altrimenti il “figlio” non può essere sostituito al “padre” senza alterare il sistema.

132



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Principio di sostituibilità

Le due condizioni sono quindi:

$$\begin{aligned} PRE_{parent} &\implies PRE_{derived} & (1) \\ POST_{derived} &\implies POST_{parent} & (2) \end{aligned}$$

- (1) in un programma corretto non può succedere che PRE_{parent} valga e $PRE_{derived}$ no; l'oggetto *evoluto* deve funzionare in ogni stato in cui funzionava l'originale: non può avere **obbligazioni** più stringenti, semmai più lasche.
- (2) in un programma corretto non può succedere che valga $POST_{derived}$ ma non $POST_{parent}$; un stato corretto dell'oggetto *evoluto* deve essere corretto anche quando ci si attende i **benefici** dell'originale.

133

Principio di sostituibilità (cont.)



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Un modo per garantire che le condizioni (1) e (2) siano automaticamente vere consiste nell'assumere implicitamente che, se la classe evoluta specifica esplicitamente una precondizione P e una postcondizione Q , le reali pre- e post-condizioni siano:

$$PRE_{derived} = PRE_{parent} \vee P \quad (3) \quad PRE_{parent} \implies PRE_{derived}$$

$$POST_{derived} = POST_{parent} \wedge Q \quad (4) \quad POST_{derived} \implies POST_{parent}$$

In Eiffel: `require` `else` `e` `ensure` `then`

134

Contratti "astratti"



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

```
extend (x: G)
-- Add 'x' at end of list.
```

```
require
  space_available: not full
deferred
ensure
  one_more:
    count = old count + 1
end
```

```
full: BOOLEAN
-- Is representation full?
-- (Default: no)
do
  Result := False
end
```

Stronger precondition... ma weaker (uguali in realtà) in astratto

```
full: BOOLEAN
-- Is representation full?
-- (Answer: if and only if
-- number of items is equal
-- to capacity)
do
  Result := (count = capacity)
end
```

135

Problema: i parametri...



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

- Animale mangia Cibo (is_a Cosa)
- Mucca (is_a Animale) mangia Erba (is_a Cibo)

Ma questa **covarianza** è contraria al principio di Liskov perché restringe le precondizioni. La controvarianza (Mucca mangia Cosa, Sather) e l'invarianza (Mucca mangia Cibo, Java) vanno bene.

Eiffel invece è covariante... (il che, impedendo un controllo di conformità statico, introduce parecchie complicazioni $\rightsquigarrow$ CATcall, run time type identification...).

136

Trattamento delle situazioni anomale



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Nel modello di Eiffel hanno un ruolo importante le eccezioni, che vengono trattate in un modo differente da quello dei più diffusi linguaggi di programmazione (Ada-like).

Exception

An exception is a run-time event that may cause a routine call to fail (**contract violation**). A failure of a routine causes an exception in its caller.

137



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Sicuramente c'è un **difetto**.

- Se il contratto è corretto:
 - violazione delle precondizioni: responsabile è il *client*
 - violazione delle postcondizioni o invarianti: responsabile è l'implementatore

138



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

Due modalità:

- ① **Failure** (organized panic): clean up the environment, terminate the call and report failure to the caller.
- ② **Retry**: attempt to change the conditions that led to the exception and to execute the routine again from the start.

Per trattare il secondo caso, Eiffel introduce il costrutto *rescue/retry*. Se il corpo del 'rescue' non fa 'retry', si ha un failure.

139

Esempio



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

```
make_with_current_time
local
f: RAW_FILE
t: INTEGER
do
  create f.make_open_temporary
  t := f.date.integer_remainder (24*60*60)
  set_seconds (t.integer_remainder (60))
  set_minutes (t.integer_remainder (60*60).integer_quotient (60))
  t := t - t.integer_remainder (60*60)
  set_hours(t.integer_quotient (60*60))
  f.delete
end
```

140

Esempio



Svigruppo

Monga

Contratti ed ereditarietà

Eccezioni

```
make_with_current_time
local
f: RAW_FILE
t: INTEGER
do
  create f.make_open_temporary
  t := f.date.integer_remainder (24*60*60)
  set_seconds (t.integer_remainder (60))
  set_minutes (t.integer_remainder (60*60).integer_quotient (60))
  t := t - t.integer_remainder (60*60)
  set_hours(t.integer_quotient (60*60))
  f.delete
rescue
  make (0, 0)
end
```

141

Esempio



```
div (num: REAL, denom: REAL): REAL
  require
    denom /= 0
  deferred

quasi_inverse (x: REAL): REAL
  -- div(1, x) if possible, otherwise 0
  local
    division_tried: BOOLEAN
  do
    if not division_tried then
      Result := div (1, x)
    else
      Result := 0
    end
  rescue
    division_tried := True
    retry
  end
```

142

Svigruppo

Monga

Contratti ed
ereditarietà

Eccezioni

Correttezza



Per ogni feature (pubblica) f :

- $\{PRE_f \wedge INV\} body_f \{POST_f \wedge INV\}$
- $\{True\} rescue_f \{INV\}$
- $\{True\} retry_f \{INV \wedge PRE_f\}$

143

Svigruppo

Monga

Contratti ed
ereditarietà

Eccezioni