



Svigruppo

Monga

Eiffel

What & How

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 2022/23, II semestre

¹ © 2023 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

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Lezione XV: *Design by Contract* con Eiffel

129



Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Organizzazione delle asserzioni

- pre/post-condizioni sulle feature (`require`, `ensure`)
- Invarianti di classe (`invariant`)
- asserzioni (`check`)
- loop invariant
(`from .. invariant .. until .. variant .. loop .. end`)

130



Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Il supporto del linguaggio

- **invarianti di classe** sono condizioni che devono essere vere in ogni momento “critico”, ossia osservabile dall'esterno. In pratica e come se facessero parte di ogni pre- e post-condizione.
- è possibile avere un supporto run-time alle **violazioni**: se una condizione non vale viene sollevata un'eccezione
- L'eccezione porta il sistema nel precedente stato stabile ed è possibile
 - terminare con un fallimento
 - riprovare

131



```

class
  GCD

feature

gcd (x, y: NATURAL): NATURAL
  require
    x /= 0 and y /= 0
  local
    t: NATURAL
  do
    from
      Result := x;
      t := y
    invariant
      Result > 0
      t > 0
      mathgcd (Result, t) = mathgcd (x, y)
    until
      Result = t
    loop
      if Result > t then
        Result := Result - t
      else
        t := t - Result
      end
    variant
      t.max (Result).as_integer_64
    end
  ensure
    Result > 0 and Result <= x * y
    dividex: x.integer_remainder (Result) = 0
    dividey: y.integer_remainder (Result) = 0

```

132

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni



```

end

feature {NONE}

mathgcd (x, y: NATURAL): NATURAL
  do
    from
      Result := x.min (y)
    until
      y.integer_remainder (Result) = 0 and then x.integer_remainder (Result) = 0
    loop
      Result := Result - 1
    end
  end
end

```

133

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Procedurale vs. Dichiarativo



Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Spesso si scrivono le “stesse” cose due volte:

```

do
  balance := balance - x
ensure
  balance = old balance - x

```

- Implementazione e specifica

- How & What

Il client è responsabile delle precondizioni, il fornitore di postcondizioni e invarianti.

134

Contratti ed ereditarietà



Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

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.

135

Principio di sostituibilità



Le due condizioni sono quindi:

$$PRE_{parent} \implies PRE_{derived} \quad (1)$$

$$POST_{derived} \implies POST_{parent} \quad (2)$$

- (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.

136

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Principio di sostituibilità (cont.)



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 preconditione 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`

137

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Contratti "astratti"



```
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
```

138

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Problema: i parametri...



- 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 preconditioni. 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...).

139

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni



Svigruppo

Monga

Eiffel

What & How

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.

140



Svigruppo

Monga

Eiffel

What & How

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

141



Svigruppo

Monga

Eiffel

What & How

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.

142



Svigruppo

Monga

Eiffel

What & How

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
```

143

Esempio



```
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
```

144

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

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
```

145

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni

Correttezza



Per ogni feature (pubblica) f :

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

146

Svigruppo

Monga

Eiffel

What & How

Contratti ed ereditarietà

Eccezioni