



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Sviluppo software in gruppi di lavoro complessi¹

Mattia Monga

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

Anno accademico 2018/19, I semestre

¹ © 2018 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Lezione VII: Documentazione dei componenti



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

La suddivisione del lavoro sw

Come suddividere il lavoro, senza la continua necessità di coordinazione?

Perché un sottogruppo di lavoro possa procedere in "isolamento" dovrebbe conoscere i componenti sviluppati da altri (o che altri svilupperanno). Cioè il loro comportamento

- in situazioni *fisiologiche* (correttezza)
- in situazioni *patologiche* (robustezza)

A questo scopo è quindi necessario specificare il funzionamento del sistema



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Correctness & robustness

IEEE *Software and Systems Engineering Vocabulary* (http://pascal.computer.org/sev_display/index.action):

Correctness

The degree to which a system or component is free from faults in its specification, design, and implementation.

Robustness

The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.



Svigruppo

Monga

Divisione del lavoro

Asserzioni
Contratti

Una specifica è una descrizione delle proprietà del marchingegno/componente utilizzato per risolvere un problema (a sua volta definito dai requisiti di progetto).

Le specifiche, perciò, sono una *descrizione* delle parti che compongono la soluzione: le modalità computazionali però sono lasciate impredicate.

What vs. How

90



Svigruppo

Monga

Divisione del lavoro

Asserzioni
Contratti

Le specifiche costituiscono naturalmente l'interfaccia fra gruppi che si suddividono l'implementazione di un sistema complesso.

- Il coordinamento **rimane necessario** a livello di specifica: ma accordarsi su **cosa** sembra più facile che sul **come**;
- I sottogruppi avranno la responsabilità di **aderire alle specifiche** nelle loro implementazioni.

91



Svigruppo

Monga

Divisione del lavoro

Asserzioni
Contratti

Perry & Evangelist (nel 1985) identificano una serie di "Interface Fault" che rimangono sostanzialmente comuni anche nei sistemi complessi di oggi.

- *Construction (mismatch interface/implementation).*
- *Inadequate functionality.*
- *Disagreements on functionality.*
- *Misuse of interface.*
- *Data structure alteration.*
- *Violation of data constraints.*
- *Initialization/value errors.*
- *Inadequate error processing.*
- *Inadequate postprocessing (resource deallocation).*
- *Inadequate interface support.*
- *Changes/Added functionality.*
- *Coordination of changes.*
- *Timing/performance problems.*

92



Svigruppo

Monga

Divisione del lavoro

Asserzioni
Contratti

Il meccanismo base per monitorare/verificare l'aderenza di una implementazione alle specifiche (e ridurre gli *interface fault*):

Assertion

(1) a **logical** expression specifying a program state that must exist or a set of conditions that program variables must satisfy at a particular point during program execution. (2) a function or macro that complains loudly if a design assumption on which the code is based is not true.

93

assert (3)



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

```
NAME
assert - abort the program if assertion is false

SYNOPSIS
#include <assert.h>

void assert(scalar expression);

DESCRIPTION
If the macro NDEBUG was defined at the moment <assert.h>
was last included, the macro assert() generates no code, and
hence does nothing at all. Otherwise, the macro assert() prints
an error message to standard error and terminates the program
by calling abort(3) if expression is false (i.e., compares equal to zero).

CONFORMING TO
POSIX.1-2001, C89, C99. In C89, expression is required to be of type int.

BUGS
assert() is implemented as a macro; if the expression tested
has side-effects, program behavior will be different depending
on whether NDEBUG is defined. This may create Heisenbugs
which go away when debugging is turned on.
```

94

Ubiquo



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Ormai presente in quasi tutti i linguaggi nativo o nelle librerie standard:

Java assert

Python assert

PHP assert

Javascript console.assert (non in Explorer...)

...

95

Usi delle asserzioni



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

È utile ragionare su “pattern” di asserzioni, spesso codificati in assertion languages/libraries.

D. S. Rosenblum, “Towards a Method of Programming with Assertions”, ICSE 1992 (Most influential paper award ICSE 2002).

Descrive un preprocessore (APP) per produrre asserzioni: il preprocessore lavora su speciali “commenti” `/*@ @*/`:

- assume
- promise
- return
- assert

96

Esempi



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

```
int square_root(int x);
/*@
  assume x >= 0;
  return y where y >= 0;
  return y where y*y <= x
  && x < (y+1)*(y+1);
*/

void swap(int* x, int* y);
/*@
  assume x && y && x != y;
  promise *x == in *y;
  promise *y == in *x;
*/
void swap(int* x, int* y) {
  *x = *x + *y;
  *y = *x - *y;
  /*@ assert *y == in *x; @*/
  *x = *x - *y;
}
```

97



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

- Consistency between arguments
- Dependency of return value on arguments
- Effect on global state/Frame specifications
- The context in which a function is called
- Subrange membership of data/Enumeration membership of data
- Non-null pointers
- Condition of the else part of complex if (and switch)
- Consistency between related data
- Intermediate summary of processing



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Se usate estensivamente, le asserzioni possono costituire una vera e propria **specifica** delle componenti del sistema. L'idea della **progettazione per contratto** (B. Meyer, 1986) è che il linguaggio per descrivere specifiche e implementazioni è lo stesso: la specifica è parte integrante del codice del sistema. La specifica è parte del "contratto" secondo cui ciascun componente fornisce i propri servizi al resto del sistema.



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

Che tipo di specifiche si usano?

$$\{P\}S\{Q\}$$

Ogni esecuzione di S che parta da uno stato che soddisfa la condizione P (pre-condizione) **termina** in uno stato che soddisfa la condizione Q (post-condizione).
Ogni programma che termina è corretto se e solo se vale la proprietà precedente.



Svigruppo

Monga

Divisione del lavoro
Asserzioni
Contratti

La tripla di Hoare $\{P\}S\{Q\}$ può diventare un **contratto** fra chi *implementa* (fornitore) S e chi *usa* (cliente) S

- L'implementatore di S si impegna a garantire Q in tutti gli stati che soddisfano P
- L'utilizzatore di S si impegna a chiedere il servizio in un stato che soddisfa P ed è certo che se S termina, si giungerà in uno stato in Q vale

Il lavoro dell'implementatore è particolarmente facile quando: Q è True (vera per ogni risultato!) o quando P è False (l'utilizzatore non riuscirà mai a portare il sistema in uno stato in cui tocchi fare qualcosa!). Weakest precondition (data Q) o strongest postcondition (data P) **determinano** il ruolo di una feature.