PSP_{DC}: An Adaptation of the PSP to Incorporate Verified Design by Contract

TSP Symposium
September 2012

Silvana Moreno, Universidad de la República
Álvaro Tasistro, Universidad ORT del Uruguay
Diego Vallespir, Universidad de la República

Motivation

- To investigate whether PSP improves quality of software when incorporating formal methods, specifically verified design by contract.

- This implies:
  - Adapting PSP incorporating verified design by contract.
  - Quantitatively evaluating the application of the new process.

PSP_{DC} (Design by Contract)

- PSP_{DC} : adaptation of the PSP that incorporates Verified Design by Contract.
- Adding up advantages of the PSP and of formal methods.

Verified Design by Contract

- Design by Contract (DbC) is a design technique that uses formal notation for specifying classes (pre- and post-conditions of methods and class invariants.)
- Verified DbC adds the requirement of proving that implementations satisfy the specifications.

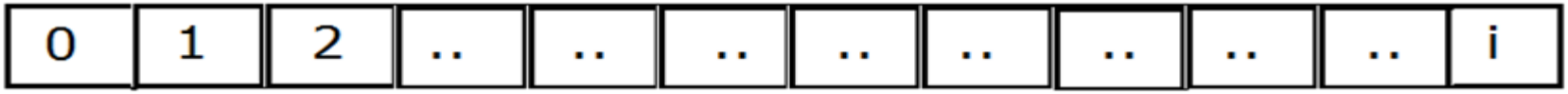
Verified Design by Contract

- Types of statements used in Design by Contract
  - Pre-conditions of methods
  - Pos-conditions of methods
  - Invariants of classes

Verified Design by Contract

- Pre-conditions must be satisfied each time the method is invoked.
- Post-conditions must be satisfied at exit of each method.
- Class invariants must be satisfied at every visible state of each instance of the class.

Example:
Calculate the mean of the elements of an array using PSP_{DC}



Formal Specification

```
/*@
  @ requires list.length != 0;
  @ ensures \result == (\sum int I; 0 <= I && I
  < list.length; list[I])/list.length;
  @ ensures (\forall int I; 0 <= I && I <
  list.length;
  @          list[I] == \old(list[I]));
  @*/
public static double Media(double[] list) {}
```

Formal Specification Review

- Sample error: index the array from the position 1, instead from the position 0.

Formal Specification Compile

- Sample error: missing “;”

Code

```
/*@
  @ requires list.length != 0;
  @ ensures \result == (\sum int I; 0 <= I && I <
  list.length; list[I])/list.length;
  @ ensures (\forall int I; 0 <= I && I < list.length;
  @          list[I] == \old(list[I]));
  @*/
public static double Media(double[] list) {
  double media = 0;
  for (int i = 0; i < list.length; i++){
      media = media + list[i];
  }
  return media/list.length;
}
```

Proof

The postcondition is that the variable **media** contains the sum of the array

The loop invariant is that **media** contains the sum of the array to the **i - 1** position.

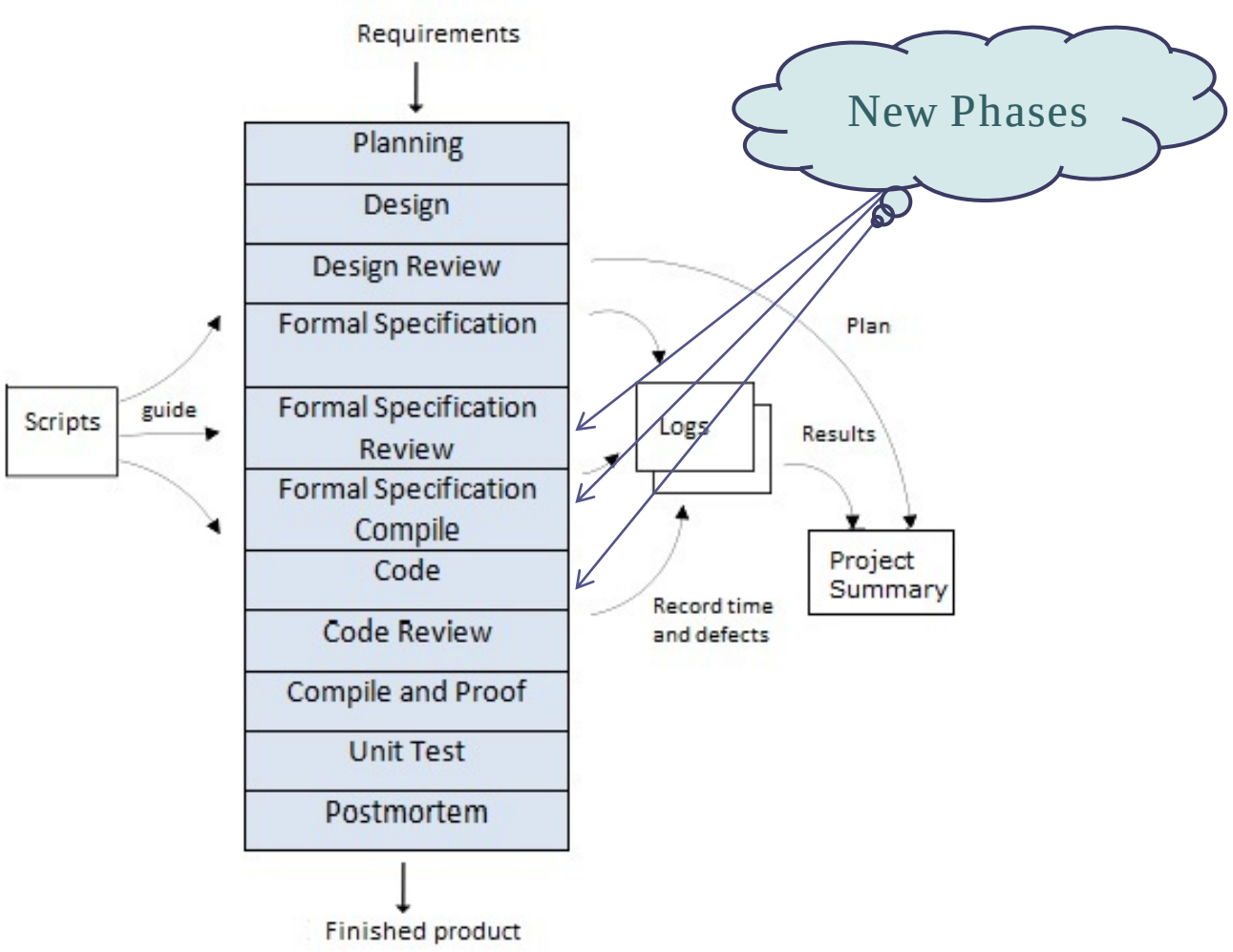
The end of the loop will be when **i == list.length** (end of the array).

PSP_{DC}

- PSP_{DC} incorporates new phases:
  - Formal Specification
  - Formal Specification Review
  - Formal Specification Compile

- Modifies existing phases
- Includes scripts and check lists.
- Includes a classification of defects injected into the formal specification.

PSP_{DC}


Requirements
New Phases
Planning
Design
Design Review
Formal Specification
Formal Specification Review
Formal Specification Compile
Code
Code Review
Compile and Proof
Unit Test
Postmortem
Finished product
Scripts
guide
Logs
Plan
Results
Project Summary
Record time and defects

PSP_{DC}: Formal Specification

- Specify pre- and post-conditions of methods and invariant of each class, using formal notation.
- This constitutes a new phase—we want to measure the time spent in formal specification.
- It follows Design Review—reviews are effective for detecting injected defects.

PSP_{DC}: Formal Specification

Script

I. Construction:
   - defining each class with its method headers (in the computarized environment)

II. Specification:
   - write down in the carrier language the pre- and post-conditions of each method as well as the class invariant

PSP_{DC}: Formal Specification Review

- Formal Specification Review has the objective of detecting as many defects as possible that has been injected during Formal Specification.
- To avoid them propagating into later phases.

PSP_{DC}: Formal Specification Review

Script

I. Review: inspecting the sentences of the specification using a check list

II. Correction: all defects detected during Review are removed

III. Checking: looking over the corrections in order to verify their adequacy

PSP_{DC}: Formal Specification Compile

- Any computerized tool supporting Verified Design by Contract will be able to compile the formal specification
- It is worthwhile to detect all possible errors in the formal specifications before any coding is carried out

PSP_{DC}: Compile and Proof

- The phase Code Compile is modified in order to provide, besides the compiled code, evidence of its correctness with respect to the formal specification (formal proof).
- The formal proof gives evidence of the correctness of the code with respect to the formal specification.
- Tools can be used to help derive the proof.

Review Process

Planning
Design
Design Review
Formal Specification
Formal Specification Review
Formal Specification Compile
Code
Code Review
Compile and Proof
Unit Test
Postmortem

Planning
Design
Design Review
Test Case Construct
Formal Specification
Formal Specification Review
Formal Specification Compile
Pseudo code
Pseudo code Review
Code
Code Review
Compile and Proof
Unit Test
Postmortem

Conclusions

- We construct a process to adapt PSP to incorporated Verified Design by Contract.

- Added:
  - New phases
  - New script
  - Forms
  - Checklists
  - New classification for the defects of the formal specification.

Future Works

- Specify and make changes to the support tool.

- Evaluate PSP_{DC} by performing:
  - Case Studies
  - Controlled Experiments

Thanks!