

MICHIGAN STATE  
UNIVERSITY

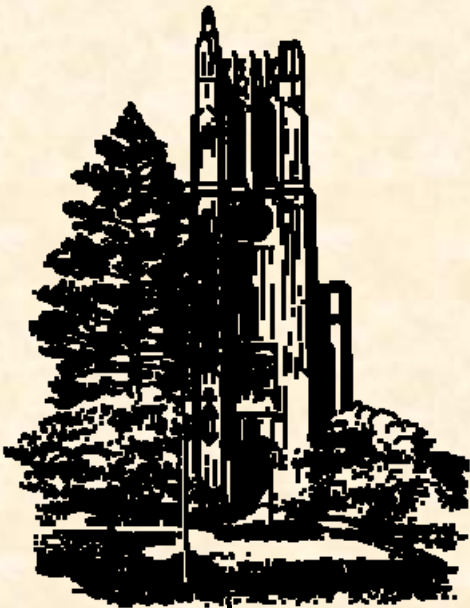
# From Z to Ada: A Case Study

CSE 814  
Formal Methods in Software Development

Troy Brown

Department of Computer Science and Engineering  
Michigan State University

Fall 2009





# Introduction

- Learn an Abstract Specification Language
- Reason Formally about an Abstract Specification
- Refine an Abstract Specification to a Concrete Specification



# Method

## Case Study:

- Informal Written Specification
- Z Specification (Abstract)
  - Initial Specification
  - Type Checking
  - Proof
  - Error Handling
- Intuitive Refinement to SPARK



# Informal Written Specification

## Capabilities:

- The Job Scheduler shall provide the capability to:
  - Be initialized to an empty scheduler.
  - Add a job to the scheduler.
  - Remove a job from the scheduler.
  - Select the highest priority job from the scheduler.
    - Note: (0=highest priority, 1 = next highest priority, etc).
- The Job Scheduler shall remove the job from the scheduler after it has been successfully selected.

## Constraints:

- The Job Scheduler shall prevent a job from being added to the scheduler if:
  - It already exists in the scheduler.
  - Its associated priority already exists in the scheduler.



# What is in a Z Specification?

- State Based Model
- Discrete Mathematics
- Paragraphs
  - Types
  - Schemas
  - etc.

# S

## Initial Z Specification



*[JOB]*

*PRIORITY == N*

*Scheduler*

*schedule : JOB ↔ PRIORITY*

*pending : P JOB*

*pending = dom schedule*

*InitScheduler*

*Scheduler'*

*schedule' = ∅*

*pending' = ∅*

# S

## Initial Z Specification

[*JOB*]

*PRIORITY* ==  $\mathbb{N}$

Basic Types

*Scheduler*

*schedule* : *JOB*  $\leftrightarrow$  *PRIORITY*

*pending* :  $\mathbb{P}$  *JOB*

*pending* = dom *schedule*

*InitScheduler*

*Scheduler'*

*schedule'* =  $\emptyset$

*pending'* =  $\emptyset$

# S

## Initial Z Specification

[*JOB*]

*PRIORITY* ==  $\mathbb{N}$

State Schema

*Scheduler*

*schedule* : *JOB*  $\leftrightarrow$  *PRIORITY*

*pending* :  $\mathbb{P}$  *JOB*

*pending* = dom *schedule*

*InitScheduler*

*Scheduler'*

*schedule'* =  $\emptyset$

*pending'* =  $\emptyset$

# S

## Initial Z Specification



[*JOB*]

*PRIORITY* ==  $\mathbb{N}$

*Scheduler*

*schedule* : *JOB*  $\leftrightarrow$  *PRIORITY*

*pending* :  $\mathbb{P}$  *JOB*

*pending* = dom *schedule*

Initialization Schema



*InitScheduler*

*Scheduler'*

*schedule'* =  $\emptyset$

*pending'* =  $\emptyset$

# S

## Initial Z Specification (Cont'd)

### AddJob

$\Delta$ Scheduler

*job?* : JOB

*priority?* : PRIORITY

*job?*  $\notin$  pending

*priority?*  $\notin$  ran schedule

$schedule' = schedule \cup \{(job? \mapsto priority?)\}$

### RemoveJob

$\Delta$ Scheduler

*job?* : JOB

*job?*  $\in$  pending

$schedule' = \{job?\} \triangleleft schedule$

### ScheduleJob

$\Delta$ Scheduler

RemoveJob[*job!*/*job?*]

*job!* : JOB

$schedule \neq \emptyset$

$job! = schedule \sim (\min(\text{ran } schedule))$

Operation

Schema

# S

## Initial Z Specification (Cont'd)

### AddJob

$\Delta$ Scheduler

*job?* : JOB

*priority?* : PRIORITY

*job?*  $\notin$  pending

*priority?*  $\notin$  ran schedule

$schedule' = schedule \cup \{(job? \mapsto priority?)\}$

$\Delta$ Scheduler

Scheduler

Scheduler'

### RemoveJob

$\Delta$ Scheduler

*job?* : JOB

*job?*  $\in$  pending

$schedule' = \{job?\} \triangleleft schedule$

### ScheduleJob

$\Delta$ Scheduler

RemoveJob[*job!*/*job?*]

*job!* : JOB

$schedule \neq \emptyset$

$job! = schedule \sim (\min(\text{ran } schedule))$

# S

## Initial Z Specification (Cont'd)

### AddJob

$\Delta$ Scheduler

$job? : JOB$

$priority? : PRIORITY$

$job? \notin pending$

$priority? \notin ran\ schedule$

$schedule' = schedule \cup \{(job? \mapsto priority?)\}$

### RemoveJob

$\Delta$ Scheduler

$job? : JOB$

$job? \in pending$

$schedule' = \{job?\} \triangleleft schedule$

### ScheduleJob

$\Delta$ Scheduler

$RemoveJob[job!/job?]$

$job! : JOB$

$schedule \neq \emptyset$

$job! = schedule \sim (\min(ran\ schedule))$

### $\Delta$ Scheduler

Scheduler

Scheduler'

### $\Delta$ Scheduler

$schedule : JOB \mapsto PRIORITY$

$schedule' : JOB \mapsto PRIORITY$

$pending : \mathbb{P} JOB$

$pending' : \mathbb{P} JOB$

$pending = dom\ schedule$

$pending' = dom\ schedule'$

# S

## Initial Z Specification (Cont'd)

### AddJob

$\Delta$ Scheduler

$job? : JOB$

$priority? : PRIORITY$

$job? \notin pending$

$priority? \notin ran\ schedule$

$schedule' = schedule \cup \{(job? \mapsto priority?)\}$

### RemoveJob

$\Delta$ Scheduler

$job? : JOB$

$job? \in pending$

$schedule' = \{job?\} \triangleleft schedule$

### ScheduleJob

$\Delta$ Scheduler

RemoveJob[job!/job?]

$job! : JOB$

$schedule \neq \emptyset$

$job! = schedule \sim (\min(ran\ schedule))$

$\Delta$ Scheduler

Scheduler

Scheduler'

$\Delta$ Scheduler

$schedule : JOB \mapsto PRIORITY$

$schedule' : JOB \mapsto PRIORITY$

$pending : \mathbb{P}\ JOB$

$pending' : \mathbb{P}\ JOB$

$pending = dom\ schedule$

$pending' = dom\ schedule'$

### AddJob

$schedule : JOB \mapsto PRIORITY$

$schedule' : JOB \mapsto PRIORITY$

$pending : \mathbb{P}\ JOB$

$pending' : \mathbb{P}\ JOB$

$job? : JOB$

$priority? : PRIORITY$

$pending = dom\ schedule$

$pending' = dom\ schedule'$


$job? \notin pending$

$priority? \notin ran\ schedule$

$schedule' = schedule \cup \{(job? \mapsto priority?)\}$

# S


## How to Write Z Specification?



- Most supported format is LaTeX
- Tools do exist to provide WYSIWYG editing:
  - Z/EVES Specification Editor
    - Specification
    - Type Checker
    - Proof System
  - Word Z Tools (Microsoft Word)
    - Type Checker
    - Doesn't work well with the ACM template ☹



# Z Specification Type Checking

A thick green arrow points horizontally from the end of the title text towards the right edge of the slide.

- Check the consistency of the specification
- First step of analysis of a Z specification
- Provides a level of correctness not provided by informal specifications.
- Many Z Type Checkers exist
  - Examples: FUZZ, ZTC, Z/EVES, etc.



# Z Specification Proof

---

- A number of tools available for proof:
  - Isabelle Theorem Prover
  - ProofPower
  - CADiZ
  - Z/EVES
- Some are General Theorem Provers
- Selected Z/EVES for its direct support of Z
- Many systems use Proof Scripts which obscure the actual proof

# S

## Z Specification Proof (Cont'd)



### Initialization Theorem

- Can the system be initialized as specified?

$$\begin{array}{|l} \textit{Line} \\ \hline y, m, x, b : \mathbb{Z} \\ \hline y = m * x + b \end{array}$$

$$\begin{array}{|l} \textit{Init} \\ \hline \textit{Line}' \\ \hline m' = 1 \\ x' = 1 \\ b' = 1 \\ y' = 0 \end{array}$$

# S Z Specification Proof (Cont'd)

## Precondition Investigation/Calculation

- Determine which states of the system are valid for the current specification.
- Generate operation preconditions systematically instead of intuitively.
- Formal reasoning to check intuition.

<i>AddJob</i>
<i>job? : JOB</i> <i>priority? : PRIORITY</i>
<i>job? ∉ pending</i> <i>priority? ∉ ran schedule</i> <i>schedule' = schedule ∪ {(job? ↦ priority?)}</i>

Preconditions are needed to prevent a partial injection property violation

<i>Scheduler</i>
<i>schedule : JOB ↦ PRIORITY</i> <i>pending : P JOB</i>
<i>pending = dom schedule</i>

# S

## Error Handling

Operations are not fully defined

- What happens when preconditions are not met?

<i>RemoveJob</i>
$\Delta$ Scheduler
<i>job?</i> : JOB
<i>job?</i> $\in$ pending
<i>schedule'</i> = { <i>job?</i> } $\triangleleft$ <i>schedule</i>

*STATUS* ::= *Ok*

<i>Job_Priority_Already_Exists</i>
<i>Job_Already_Exists</i>
<i>Priority_Already_Exists</i>
<i>No_Schedulable_Job</i>
<i>Job_Does_Not_Exist</i>

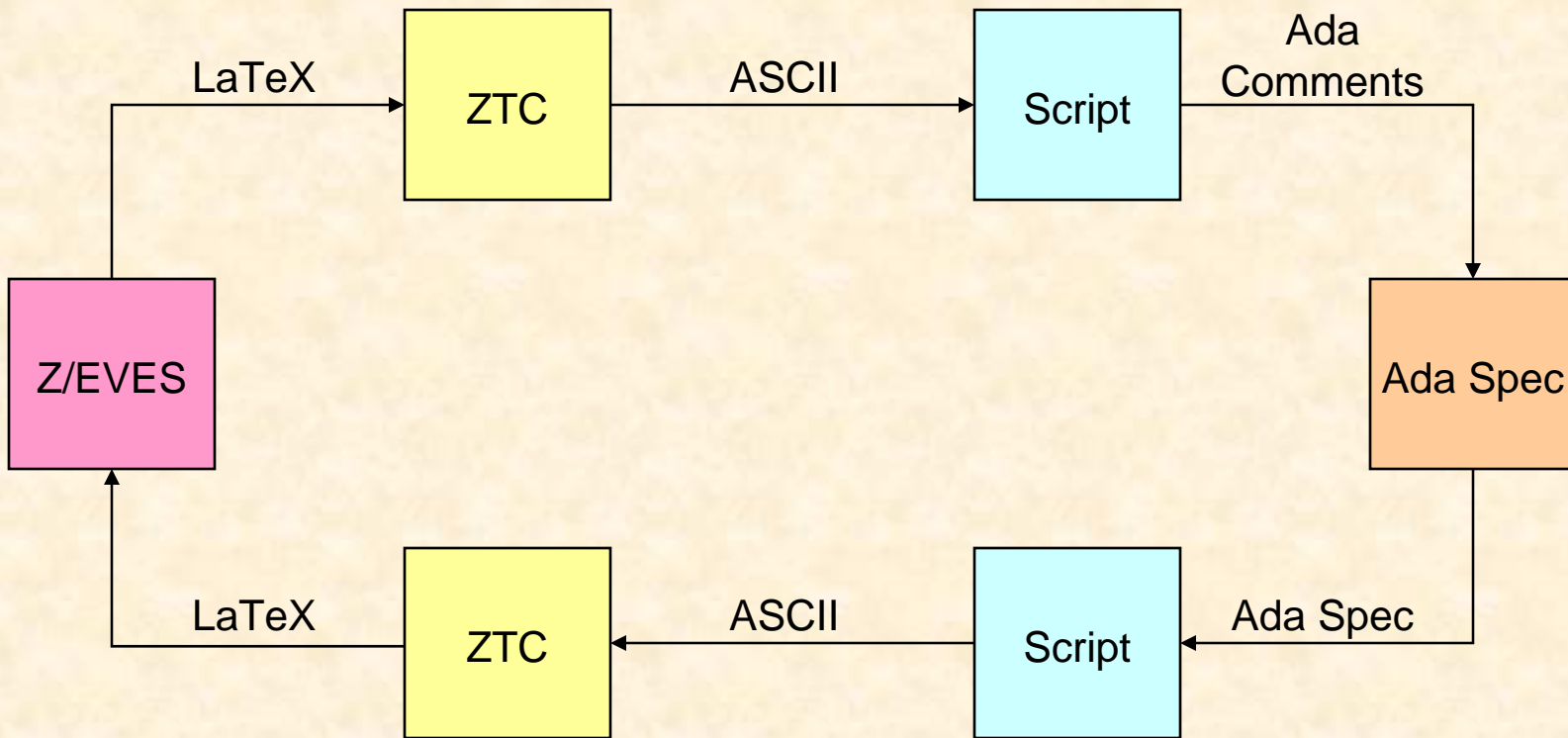
<i>Success</i>
<i>status!</i> : <i>STATUS</i>
<i>status!</i> = <i>Ok</i>

<i>JobDoesNotExist</i>
$\exists$ Scheduler
<i>job?</i> : JOB
<i>status!</i> : <i>STATUS</i>
<i>job?</i> $\notin$ pending
<i>status!</i> = <i>Job_Does_Not_Exist</i>

*RemoveJobTotal*  $\hat{=}$  (*RemoveJob*  $\wedge$  *Success*)  $\vee$  *JobDoesNotExist*

# S

# Rudimentary Round-Tripping





# Z Specification to SPARK

```
--Z specification
--Z
--Z [ JOB ]
--Z
--Z PRIORITY == N
--Z
--Z --- Scheduler -----
--Z | schedule : JOB >+> PRIORITY;
--Z | pending   : P JOB
--Z |-----
--Z | pending = dom schedule
--Z -----
--Z
...

```

# S

## Scheduler in SPARK



- Static (SPARK) vs. Dynamic (Z)
- Reserved words (SPARK)
- State Invariant

### *Scheduler*

*schedule* :  $JOB \leftrightarrow PRIORITY$

*pending* :  $\mathbb{P} JOB$

*pending* = dom *schedule*

```
type Schedule_Type      is array (Jobs.Job_Type) of Priorities.Priority_Type;  
type Pending_Jobs_Type is array (Jobs.Job_Type) of Boolean;
```

```
Schedule      : Schedule_Type;  
Pending_Jobs : Pending_Jobs_Type;
```

# S

# Z Specification to SPARK

```
procedure Remove_Job
( Job      : in      Jobs.Job_Type;
  Status   : out    Status_Type
);
--# global in out Pending_Jobs;
--# derives Pending_Jobs from Pending_Jobs, Job &
--#      Status           from Pending_Jobs, Job;
--# post (Status = Ok      ->
--#      (Pending_Jobs~(Job) and
--#      (Pending_Jobs =
--#      Pending_Jobs~[Job => False]))) and
--#
--#      (Status = Job_Does_Not_Exist ->
--#      (not Pending_Jobs~(Job) and
--#      (Pending_Jobs = Pending_Jobs~)));
```

RemoveJob

$\Delta$ Scheduler

job? : JOB

job?  $\in$  pending

schedule' = {job?}  $\triangleleft$  schedule

Success

status! : STATUS

status! = Ok

JobDoesNotExist

$\exists$ Scheduler

job? : JOB

status! : STATUS

job?  $\notin$  pending

status! = Job\_Does\_Not\_Exist

$RemoveJobTotal \hat{=} (RemoveJob \wedge Success) \vee$   
 $JobDoesNotExist$



# Results (Pros of Z)

---

- Language Agnostic Specification
- Reasoning at a Higher Level of Abstraction
- Higher Confidence than Informal Specification
- Higher Confidence than Model Checking



# Results (Cons of Z)

- Need to Learn/Have Math Background
- Non-ASCII Characters
- Lacking Good Visualization/Animation
- Lacking Proof Support for Novice



# Conclusion

- Learned Abstract Specification Language
- Reasoned Formally about Specification
- Refined Abstract Specification to Concrete Specification.



# Further Investigations

---

- Formal Refinement in Z
- Object-Z (Object Oriented specification)
- Alloy  $\rightarrow$  Z  $\rightarrow$  SPARK
  - Use Alloy for exploration
  - Use Z for formal specification/proof
  - Use SPARK for language specific application
- B Method (from the inventor of Z)
  - Lower-level than Z
  - More emphasis on refinement
- Vienna Development Method (VDM)