4/25/2020

Jason Arrabito, Divitha Kusupati, Ismael Resendiz Robles, Jennifer Shelby

Cleanroom with Little-JIL

CS 5396 Spring 2020

Table of Contents

[Overall Description 3](#_Toc40104175)

[Introduction to Little-Jil 3](#_Toc40104176)

[Introduction to Cleanroom Software Engineering 3](#_Toc40104177)

[Creation of Little-Jil Project 5](#_Toc40104178)

[Download Eclipse and Little-Jil 5](#_Toc40104179)

[Downloading and Installing Eclipse 5](#_Toc40104180)

[Downloading and Installing Little-JIL Plugin 5](#_Toc40104181)

[Create Little-Jil Project 5](#_Toc40104182)

[Modeling Cleanroom Process with Little-JIL 6](#_Toc40104183)

[Overall Coordination Diagram 6](#_Toc40104184)

[Resources 6](#_Toc40104185)

[Simulation 7](#_Toc40104186)

[Output 7](#_Toc40104187)

Figure 1. Cleanroom Lifecycle 4

Figure 2. Coordination Diagram 6

# Overall Description

This paper describes how we used Eclipse and Little-JIL to model the Cleanroom Software Engineering process. The purpose of this paper is to give an introduction to Little-JIL, Cleanroom Software Engineering, and to describe how our project was created. Section 2 will introduce Little-JIL, section 3 will provide an introduction to Cleanroom Software Engineering, section 4 will describe how we created our project using Little-JIL, section 5 will describe the implementation of a Cleanroom Software Engineering process using Little-JIL, and section 6 will discuss the results of using Little-JIL to implement the Cleanroom Software Engineering process.

# Introduction to Little-Jil

Software engineering involves agents (human or automated) and tools that must coordinate to produce a complex deliverable. Little-JIL is a graphical process language that tries to resolve conflicting objectives of supporting a wide variety of process abstractions and creating a language that is understandable and easy to use.

Little-JIL focuses on the coordination of agents and activities by separating coordination from other language elements. It also has a single higher-level coordination paradigm that includes a small set of computational constructs that allows programmers to refine interactions between major elements.

It allows programmers to focus on coordination and the ability to add additional layers of complexity incrementally can lead to benefits in process analysis, understanding, adaptation, and execution.

The Little-JIL process consists of the following:

* **The Coordination Paradigm.** Consists of five elements:
  + Multiple agents (human or automated) that can perform one or more tasks
  + Communication allowing agents to share information
  + A distribution mechanism allowing agents to operate on separate machines
  + Assignment of tasks to agents
  + A mechanism to tie agents and their tasks together in a way that accomplishes the activity
* **The Resources.** Resources or data necessary to complete an activity

The process of using Little-JIL results in a coordination diagram that depicts the steps of the process and specifies their order. It illustrates the sub-steps, any resources, and pre-/post-requisites to each step, if any exist. It also shows any exception or error handling necessary.

# Introduction to Cleanroom Software Engineering

Cleanroom software engineering combines formal methods of object-based box structure specifications and design, function-theoretic correctness verification, and statistical usage testing for quality certification to produce zero defect with high probability software while being team oriented.

There is a development team who build in correctness using formal specifications, design, and verification. A certification team certifies the quality of software with respect to the specification rather than testing in quality. Certification team employs statistical usage testing to assess the product quality. If any errors are found, they are returned to the development team for resolution.

This is an incremental process with continuous integration to ensure functionality growth. Because testing is done at the system level, there are fewer bugs introduces when attempting to fix a known bug.

Cleanroom development starts with customer requirements and functional specification created by the development team, or in a scaled environment, by a specification team. Usage specifications are created by the certification team. Planning between the two teams occurs to define the initial incremental development and certification plan. The development team then carries out a design and verification cycle. This design and verification have corresponding statistical test cases created by the certification team based on the usage specification. The development team delivers completed increments to the certification team for statistical testing and MTTF estimates. Errors are returned to the development team.

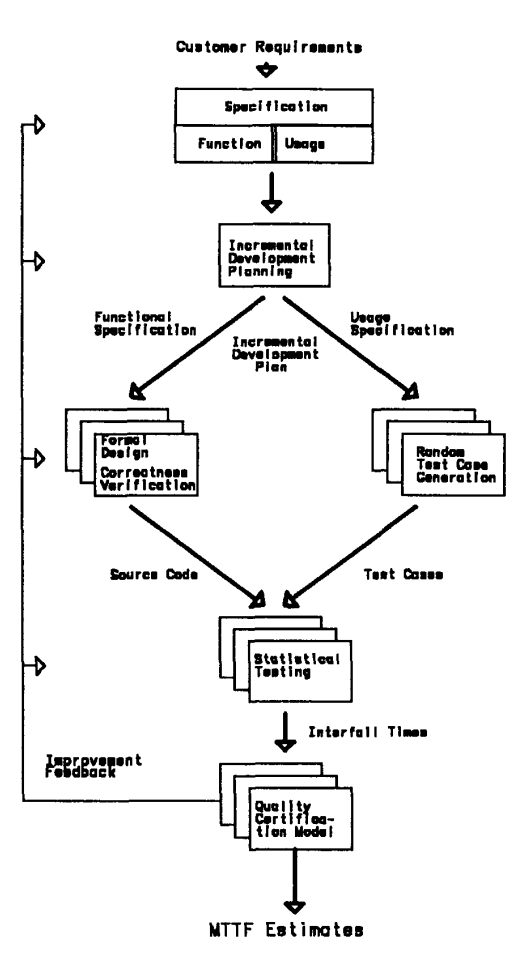


Figure . Cleanroom Lifecycle

# Creation of Little-Jil Project

## Download Eclipse and Little-Jil

Two critical pieces of software were needed for this project: Eclipse IDE for Java Developers and the Little-JIL plugin.

### Downloading and Installing Eclipse

Eclipse IDE for Java Developers can be downloaded for free at *http://www.eclipse.org/downloads*. We just installed the latest version (Eclipse 2020-03) using the prepackaged installer and followed the prompts.

### Downloading and Installing Little-JIL Plugin

Once Eclipse was installed, we launched the IDE and navigated to Help > Install New Software. A window to install Available Software popped up and in the “Work with” textbox we entered, *http://laser.cs.umass.edu/eclipse/visualjil/.* Weselected the packages we wanted to install (in this case, Visual-JIL), and clicked Add. We then followed the prompts to complete installation of the Little-JIL plugin.

## Create Little-Jil Project

Once installation completes, a new project can be opened using the Little-JIL perspective. This is found in the Eclipse IDE at Window > Perspective > Open Perspective > Other > Little-JIL > Open. A new project is created using the New Project button, selecting Little-JIL Project. This will provide a Stub Resource Model and Stub Artifact Model.

# Modeling Cleanroom Process with Little-JIL

## Overall Coordination Diagram

A picture containing text

Description automatically generated

Figure . Coordination Diagram

Figure 2 shows the coordination diagram for modeling the cleanroom process with Little-JIL. Parent steps are either sequential steps or parallel steps. Sequential steps are performed left to right, while parallel steps can be performed in tandem. The [FILE NAME HERE] file is used as the input for the Little-JIL simulator.

## Resources

Some steps require resources, denoted by the filled circle above each step name. These resources can be data artifacts, participants, or equipment. An agent is either human or automated, and the agent is responsible for execution of that process. Resources used for this model are listed below:

CleanRoomAgent:

cleanroomAgent

SpecificationTeam:

specification

Developer:

developer

Designer:

designer

tester:

tester

qualityCheck:

qualityteam

## Simulation

An XML file written in JSim Oracle Language allows us to use the simulator to view the process as it runs. We used a very simple file:

<simulation>

<rules>

<posted>

<start>

<fixed value=”1” />

</start>

</posted>

<started>

<complete>

<fixed value=”1” />

</complete>

<started>

</rules>

</simulation>

A Run Configuration also need to be created within Eclipse.

## Output

The output for our simulation is as follows:

CleanRoomProcess(1) was assigned to cleanroominit at 0

cleanroominit requested that CleanRoomProcess(1) start at 1

cleanroominit started performing CleanRoomProcess(1) at 1

Specification Stage(2) was assigned to specification1 at 1

specification1 requested that Specification Stage(2) start at 2

specification1 started performing Specification Stage(2) at 2

Usage Specification(3) was assigned to specification1 at 2

Functional Specification(4) was assigned to specification1 at 2

specification1 requested that Usage Specification(3) start at 3

specification1 started performing Usage Specification(3) at 3

specification1 requested that Functional Specification(4) start at 3

specification1 started performing Functional Specification(4) at 3

specification1 requested that Usage Specification(3) complete at 4

Usage Specification(3) completed successfully at 4

specification1 requested that Functional Specification(4) complete at 4

Functional Specification(4) completed successfully at 4

specification1 requested that Specification Stage(2) complete at 4

Specification Stage(2) completed successfully at 4

Incremental Planning(5) was assigned to cleanroominit at 4

cleanroominit requested that Incremental Planning(5) start at 5

cleanroominit started performing Incremental Planning(5) at 5

Development Stage(6) was assigned to cleanroominit at 5

cleanroominit requested that Development Stage(6) start at 6

cleanroominit started performing Development Stage(6) at 6

Design Stage(7) was assigned to designer1 at 6

Random Test Case Generation(8) was assigned to cleanroominit at 6

designer1 requested that Design Stage(7) start at 7

designer1 started performing Design Stage(7) at 7

Formal Design Creation(9) was assigned to designer1 at 7

cleanroominit requested that Random Test Case Generation(8) start at 7

cleanroominit started performing Random Test Case Generation(8) at 7

cleanroominit requested that Random Test Case Generation(8) complete at 7

Random Test Case Generation(8) completed successfully at 7

designer1 requested that Formal Design Creation(9) start at 8

designer1 started performing Formal Design Creation(9) at 8

designer1 requested that Formal Design Creation(9) complete at 8

Formal Design Creation(9) completed successfully at 8

Verification Correction(10) was assigned to designer1 at 8

designer1 requested that Verification Correction(10) start at 9

designer1 started performing Verification Correction(10) at 9

designer1 requested that Verification Correction(10) complete at 9

Verification Correction(10) completed successfully at 9

designer1 requested that Design Stage(7) complete at 9

Design Stage(7) completed successfully at 9

cleanroominit requested that Development Stage(6) complete at 9

Development Stage(6) completed successfully at 9

Statistical Testing(11) was assigned to cleanroominit at 9

cleanroominit requested that Statistical Testing(11) start at 10

cleanroominit started performing Statistical Testing(11) at 10

cleanroominit requested that Statistical Testing(11) complete at 10

Statistical Testing(11) completed successfully at 10

Certification Stage(12) was assigned to qualityteam at 10

qualityteam requested that Certification Stage(12) start at 11

qualityteam started performing Certification Stage(12) at 11

qualityteam requested that Certification Stage(12) complete at 11

Certification Stage(12) completed successfully at 11

cleanroominit requested that Incremental Planning(5) complete at 11

Incremental Planning(5) completed successfully at 11

Incremental Planning(13) was assigned to cleanroominit at 11

cleanroominit requested that Incremental Planning(13) start at 12

cleanroominit started performing Incremental Planning(13) at 12

Development Stage(14) was assigned to cleanroominit at 12

cleanroominit requested that Development Stage(14) start at 13

cleanroominit started performing Development Stage(14) at 13

Design Stage(15) was assigned to designer1 at 13

Random Test Case Generation(16) was assigned to cleanroominit at 13

designer1 requested that Design Stage(15) start at 14

designer1 started performing Design Stage(15) at 14

Formal Design Creation(17) was assigned to designer1 at 14

cleanroominit requested that Random Test Case Generation(16) start at 14

cleanroominit started performing Random Test Case Generation(16) at 14

cleanroominit requested that Random Test Case Generation(16) complete at 14

Random Test Case Generation(16) completed successfully at 14

designer1 requested that Formal Design Creation(17) start at 15

designer1 started performing Formal Design Creation(17) at 15

designer1 requested that Formal Design Creation(17) complete at 15

Formal Design Creation(17) completed successfully at 15

Verification Correction(18) was assigned to designer1 at 15

designer1 requested that Verification Correction(18) start at 16

designer1 started performing Verification Correction(18) at 16

designer1 requested that Verification Correction(18) complete at 16

Verification Correction(18) completed successfully at 16

designer1 requested that Design Stage(15) complete at 16

Design Stage(15) completed successfully at 16

cleanroominit requested that Development Stage(14) complete at 16

Development Stage(14) completed successfully at 16

Statistical Testing(19) was assigned to cleanroominit at 16

cleanroominit requested that Statistical Testing(19) start at 17

cleanroominit started performing Statistical Testing(19) at 17

cleanroominit requested that Statistical Testing(19) complete at 17

Statistical Testing(19) completed successfully at 17

Certification Stage(20) was assigned to qualityteam at 17

qualityteam requested that Certification Stage(20) start at 18

qualityteam started performing Certification Stage(20) at 18

qualityteam requested that Certification Stage(20) complete at 18

Certification Stage(20) completed successfully at 18

cleanroominit requested that Incremental Planning(13) complete at 18

Incremental Planning(13) completed successfully at 18

cleanroominit requested that CleanRoomProcess(1) complete at 18

CleanRoomProcess(1) completed successfully at 18

In the output above, you can see the agent assigned to a step, when the step is completed. Sequential steps are competed when their last substep is completed.