2023

WSC 2023
Simulation Challenge
Tech Document

NUS National University of Singapore
Centre of Excellence in Modelling and Simulation for Next Generation Ports
Industrial Systems Engineering and Management
College of Design and Engineering

University of Exeter

KAIST

北京大学 PEKING UNIVERSITY

SUMMARY

The “Simulation Challenge” (previously named the “Case Study Competition”) in Winter Simulation Conference (WSC) was initially launched in 2022, with the aim to delve into the significance of simulation and pave the way for increased collaboration between academia and industry. Through this initiative, we target to foster an environment of interdisciplinary cooperation that encourages innovation and fosters the creation of cutting-edge solutions to real-world problems.

With the theme “Simulation for Resilient Systems”, the Simulation Challenge in WSC2023 will continue to encourage participants to explore the latest advances in simulation technology and their application to next-generation industrial systems. Participants will be required to present case studies that demonstrate the use of simulation to design, optimize, and evaluate systems that are resilient, sustainable, and adaptable to changing circumstances. The competition will provide a unique opportunity to engage with leading experts in the field, gain recognition for their work, and potentially win prizes.

Using the gating control in semiconductor fabrication as the case study for this competition, we have built a basic simulation system model with discrete-event modeling methods. The model is implemented with both C# (O2DES.NET) and Python (O2DESpy), which are frameworks for object-oriented discrete event simulation. Participants are required to choose either of them to build, design, and implement your own strategy algorithm, competing with other players. Participants are expected to embed their own algorithms in certain parts of the model, using their skills in model training and large-scale search, to improve the gating control performance. The teams that generate the best overall performance stand the best chance to win.

The purpose of this document is to provide participants with detailed descriptions of the model structure, and to guide them on the process of file downloading and submission procedures.

CONTENTS

Chapter 1

Overview

1.1 Problem Description

Welcome to this challenge. In this endeavor, we will leverage your simulation skills in concert with optimization techniques and data learning to enhance the performance of gating control within semiconductor fabrication.

During the fabrication process, Lots undergo multiple steps for completion, each of which involves being transported to different workstations equipped with the necessary tools. The Lots in a step are referred to as Work-in-Progress (WIP). It's important to note that different types of Lots have distinct process steps and Queue Time (QT) Loops. The quality of Lots is significantly influenced by the length of their queueing times. To regulate this, Maximum Queue Times (Max QTs) are assigned to Lots during specific process steps. A breach occurs when a Lot's queue time exceeds its Max QT, in which case the Lot is deemed defective.

To manage the flow of Lots downstream, we impose Gating Factors (GFs) on workstations. The GF, a ratio ranging from 0 to 1, is compared with the current and maximum workload capacity of the workstation. This helps minimize unnecessary queueing time, thereby reducing the breach rate.

Currently, the adjustment of the gating factor is manually performed based on observations and feedback from the technicians and engineers assigned to the workstation. This ratio is often a conservative estimate as minimizing the breach rate is a priority. However, this method has proven to be unstable and labor-intensive, requiring constant feedback and adjustments to the gating factor in response to the current queueing situation. Furthermore, the conservative ratios might result in the underutilization of some Tools, leading to sub-optimal throughput rates.

Underestimating the gating factor leads to reduced throughput, while overestimating it increases the likelihood of breaches. The gating factor is a critical decision variable that affects the workstation's throughput and breach rate. Therefore, the objective of this competition is to develop a mechanism to optimize the gating factor for each workstation, with the aim of minimizing the maximum WIP and the penalties incurred by breaches among the steps. We encourage contestants to employ their own strategies and methodologies, such as simulation, deep learning, or others, to identify the optimal gating factor value under the given scenarios.

This competition is not only about winning but also about learning and discovery. We hope that participants will gain fascinating insights into the complexities of semiconductor fabrication and enjoy the process of finding creative solutions to real-world challenges. This is an opportunity to apply your skills in a practical setting, leading to not only personal growth but potentially transformative changes in the industry. We look forward to your participation and the unique perspectives you will bring to this competition.

Good luck and have fun in WSC simulation challenge 2023!

1.2 Competition Rules

As shown in Figure 1, the given model contains data and source code according to the following three aspects of information:

(A) Input: Example simulation scenarios with specific parameters.
(B) Strategy: The strategy module where users can modify the code for their own decision algorithms for gating factor. The default algorithms will be provided too.
(C) Output: Performance indicators to measure the efficiency and quality of the semiconductor manufacturing process.

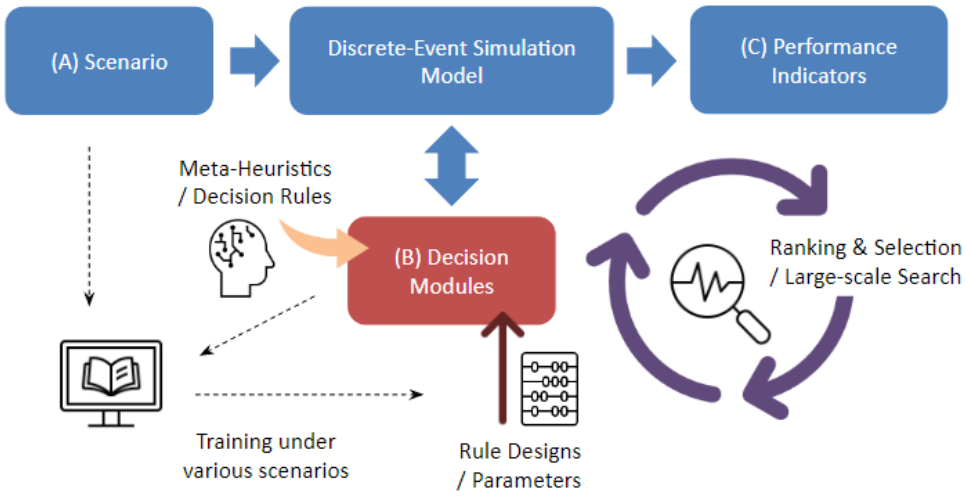


Figure 1 - Traditional Warehouse

You are expected to rewrite and replace the existing decision modules (B), to maximize the performance (C) of the system under different scenarios (A).

You can generate the logic rules in Part (B) in various ways, including but not limited to:

a. Writing rule-based scripts or heuristic algorithms embedded in decision events
b. Embedding simulation model into external optimization search algorithm
c. Using a machine learning model to identify and conclude the best rule parameters and embed them in decision events

You only need to provide part (B) of the program code and required data, and there is no need to submit optimization and training program.

Note: all other source codes besides those for Part (B) will not be evaluated, nor will they be run.

1.3 General Evaluation Guide

Your data and program code will be embedded in the discrete-event simulation model provided in advance. It will overwrite the corresponding original code, and compile and generate an executable simulation program. Your program will run under a variety of scenarios and random seeds. The winner will be the one whose model generates the top average performance index for each case.

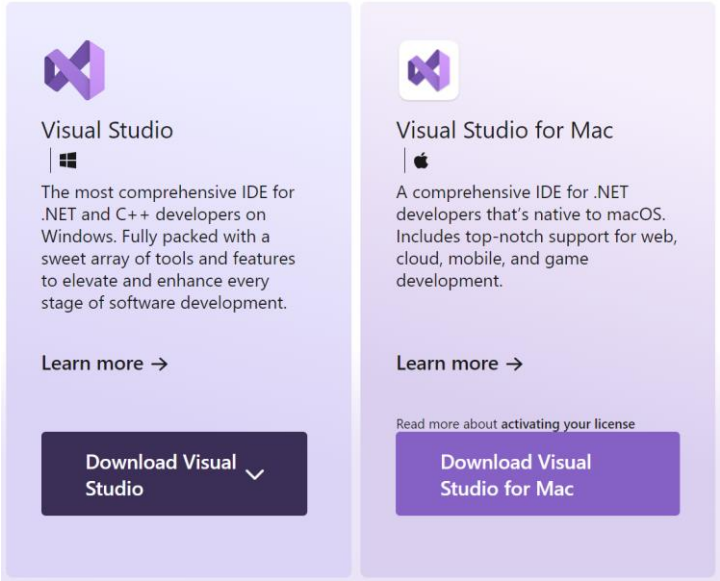
For further instructions regarding file downloads, please see Chapter 2. For elaborations on model structures, please see Chapter 3. For detailed evaluation criteria, please see Chapter 4.

Chapter 2

User Instruction

2.1 Install Visual Studio (C#)

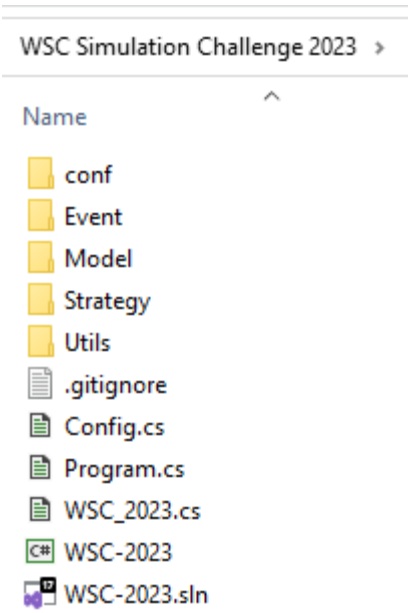
1) Go to website https://visualstudio.microsoft.com/, choose the version that suits your computer and click **Download** button to download the Visual Studio.



2) Refer to link https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022 to install Visual Studio on windows.
3) Refer to link https://learn.microsoft.com/en-us/visualstudio/mac/installation?view=vsmac-2022 to install Visual Studio for Mac.
4) Note that select the workload to run the model with C#.

2.2 Source Code (C#)

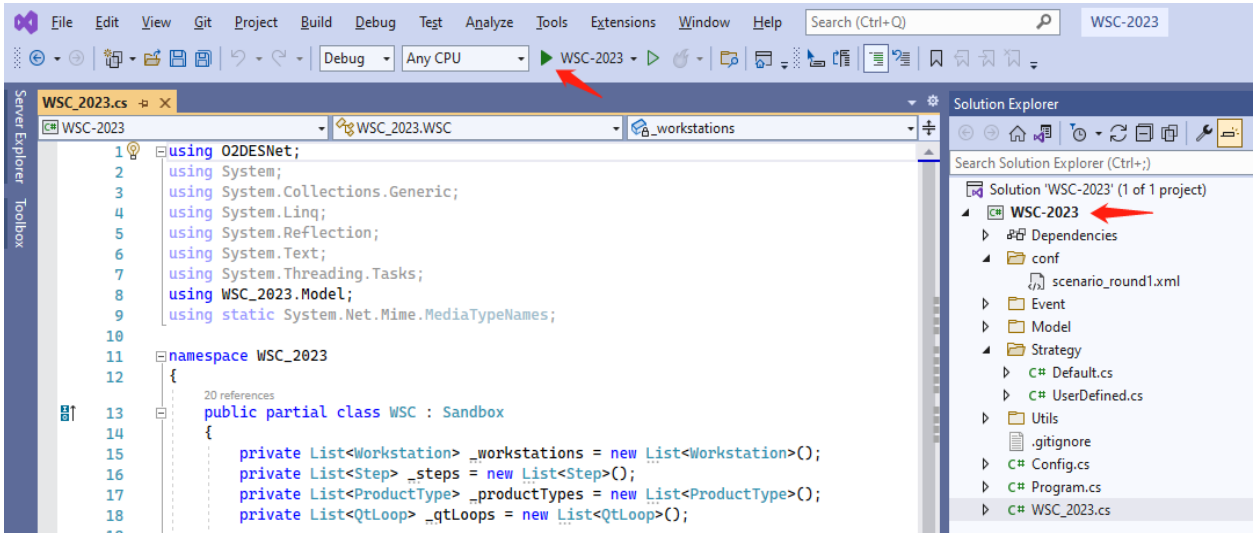
1) After registration and joining in a team, you will receive the zip package of source code by email: "**WSC Simulation Challenge 2023.zip**".
2) Decompose zip package.
3) Source code structure of '**WSC Simulation Challenge 2023**'.



conf folder: includes scenario files (XML Document)

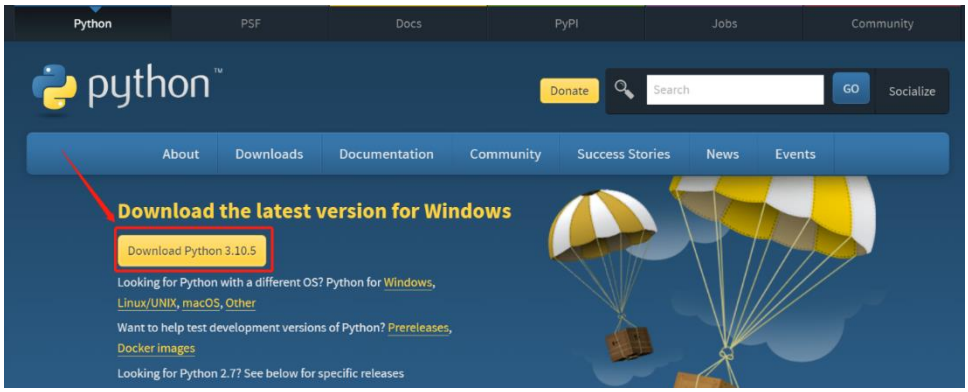
Strategy folder: includes Default and UserDefined files (C# Source File)

4) Click WSC-2023.sln to open and run the project.



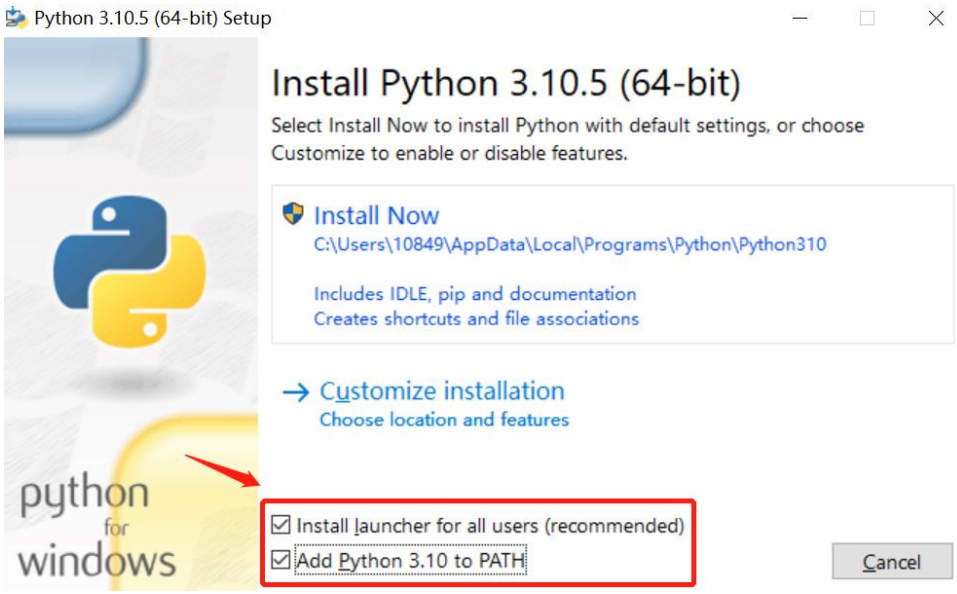
2.3 Install Python (Python)

1) Go to website https://www.python.org/downloads/ and click **Download** button to download the python executable installer

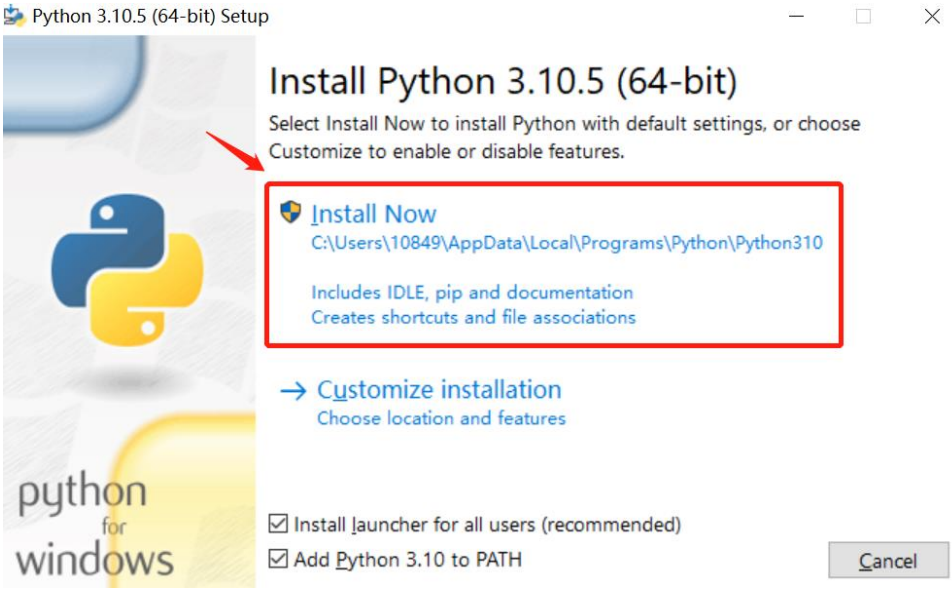


2) Double click the downloaded .exe file and start the installation process

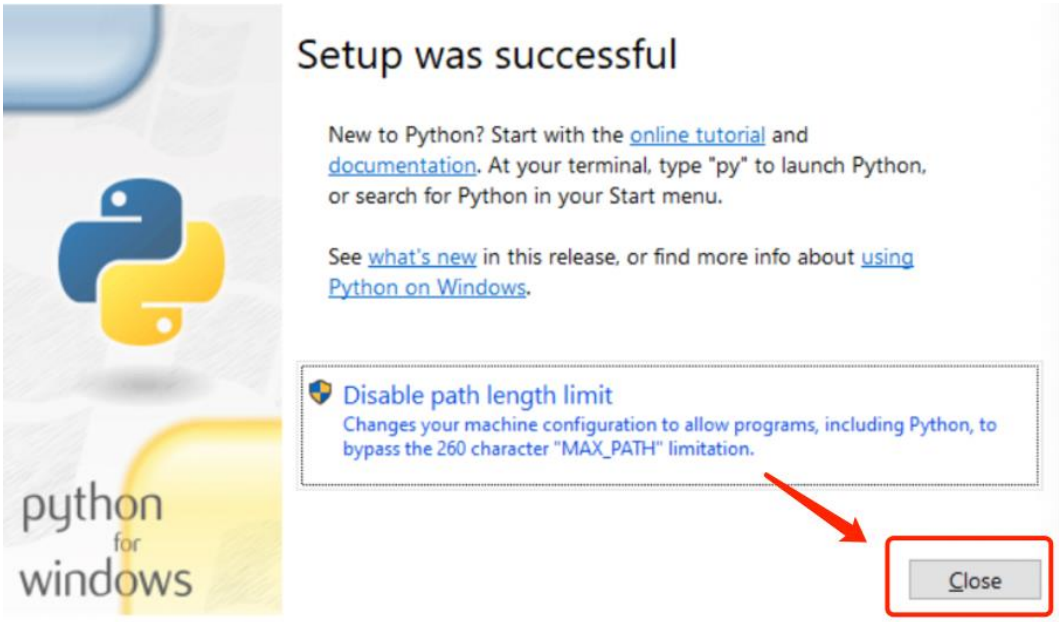
3) Tick both the **Install launcher for all users** and **Add Python 3.10 to PATH** checkboxes



4) Click **Install Now**

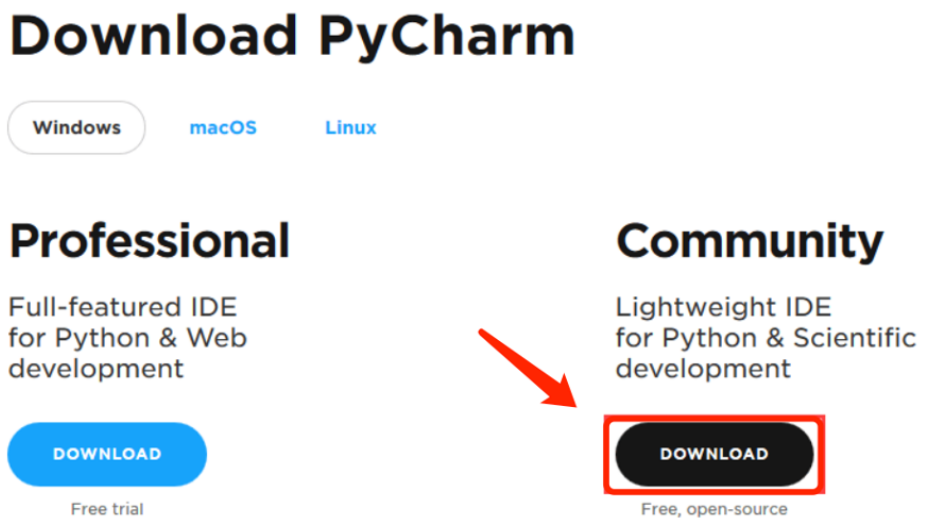


5) Wait for a while to complete the installation process and after successful setup, close the dialogue box

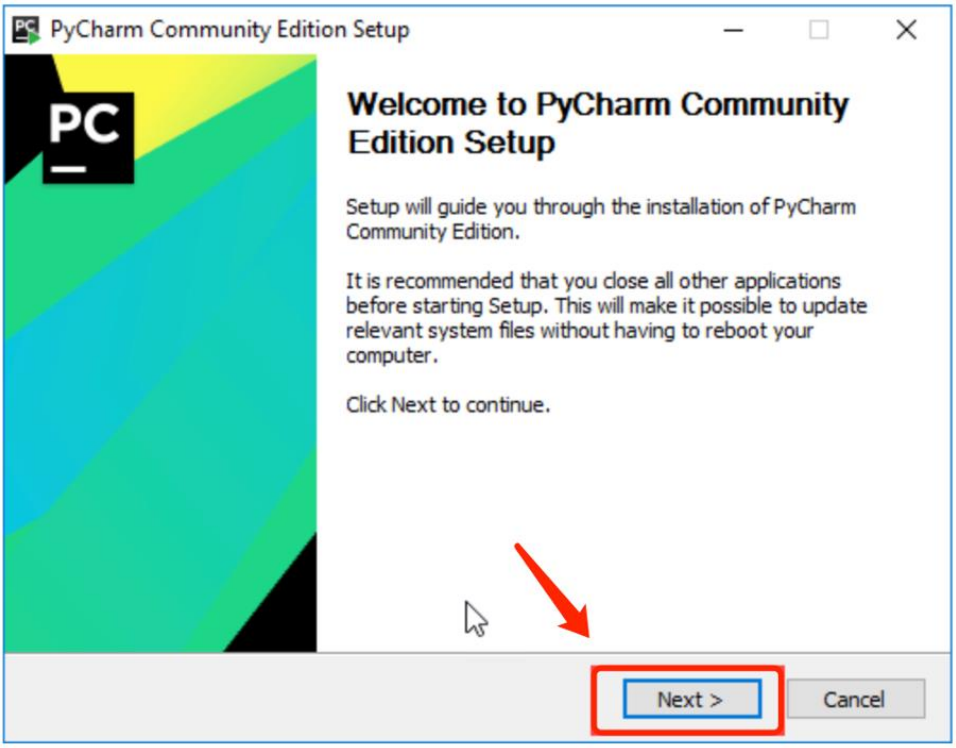


2.4 Install PyCharm (Python)

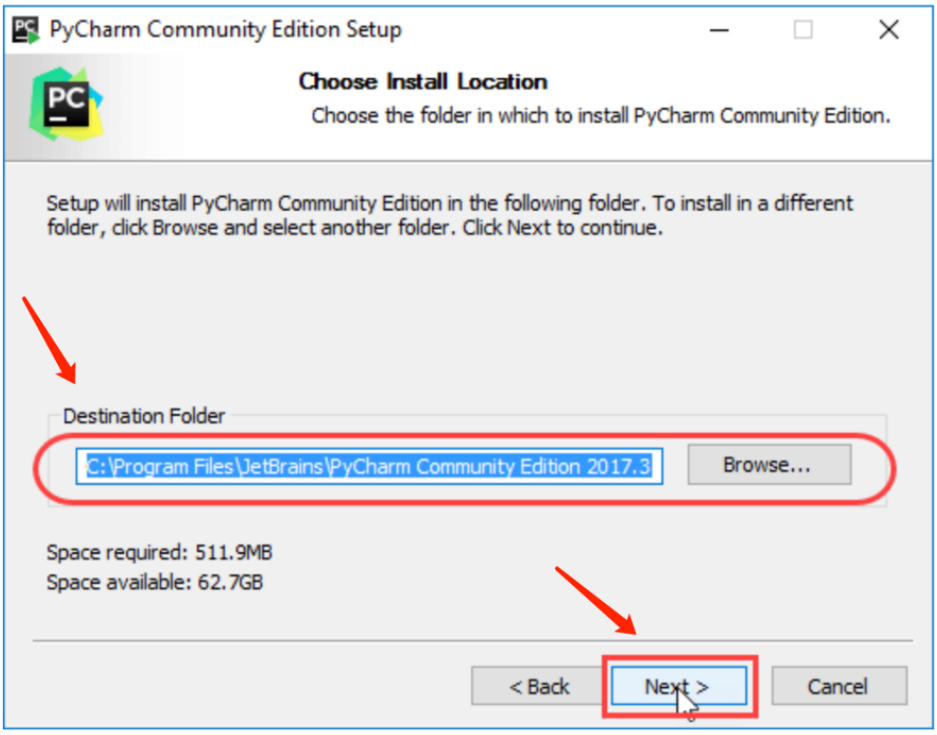
1) Go to the website https://www.jetbrains.com/pycharm/download/ and click the **DOWNLOAD** button under the Community Section.



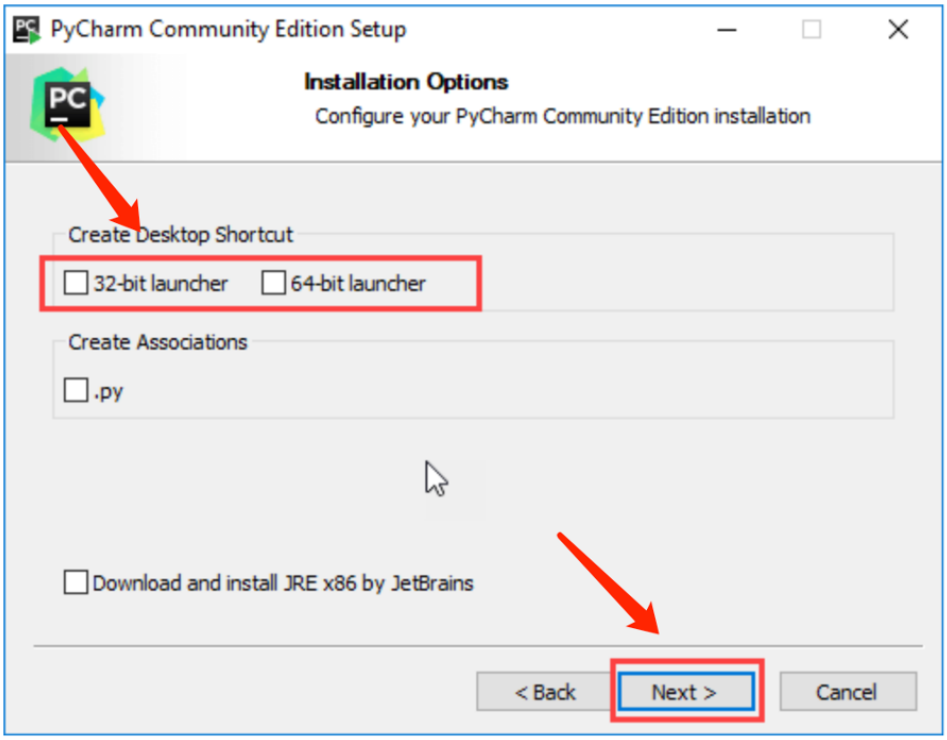
2) Double click the downloaded .exe file to start the installation process and click **Next**



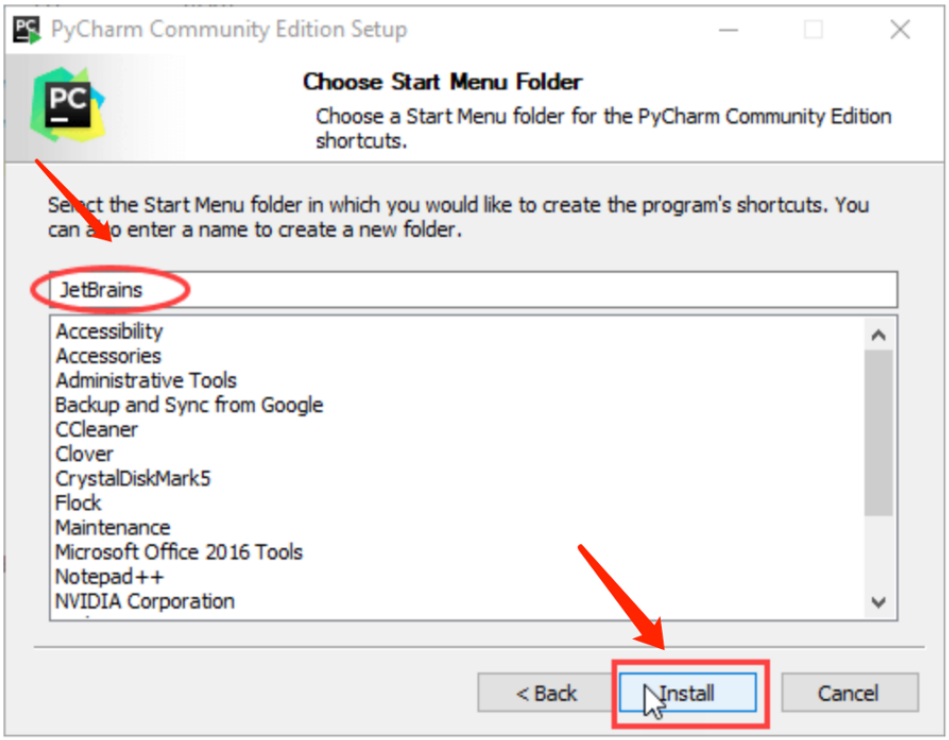
3) Modify the installation location if needed and click **Next**



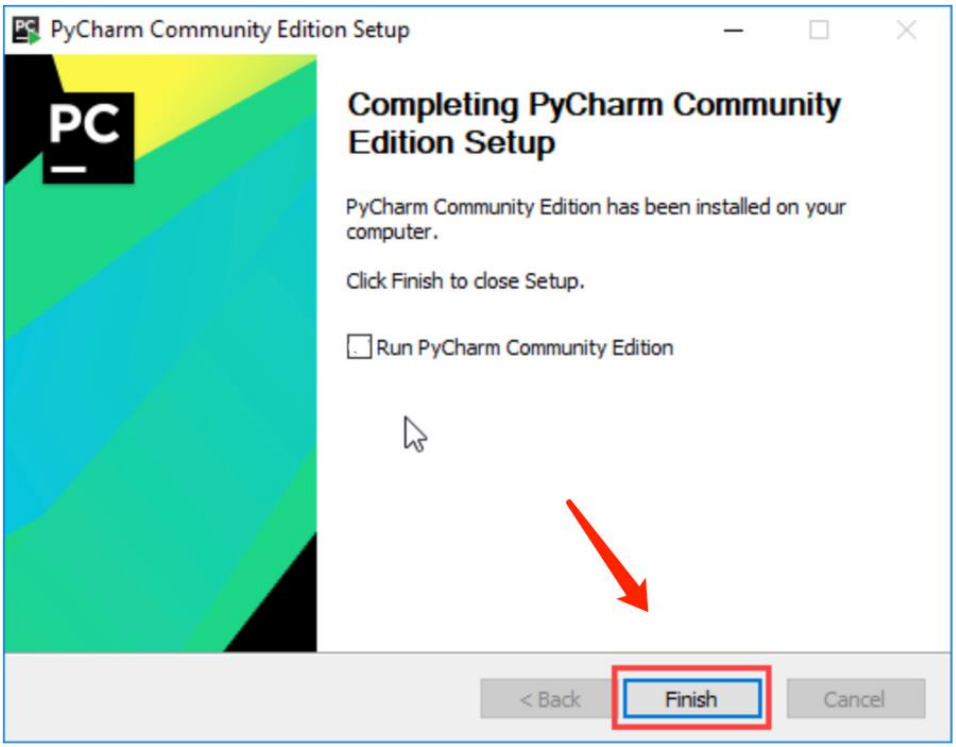
4) Choose 64-bit launcher and click **Next**



5) Modify the default start menu folder if needed and click **Install**

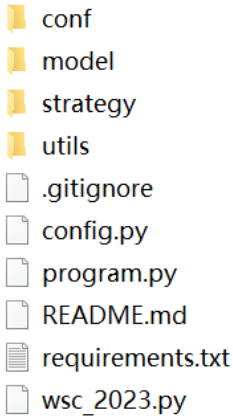


6) Wait for the installation process to finish and click **Finish**



2.5 Source Code (Python)

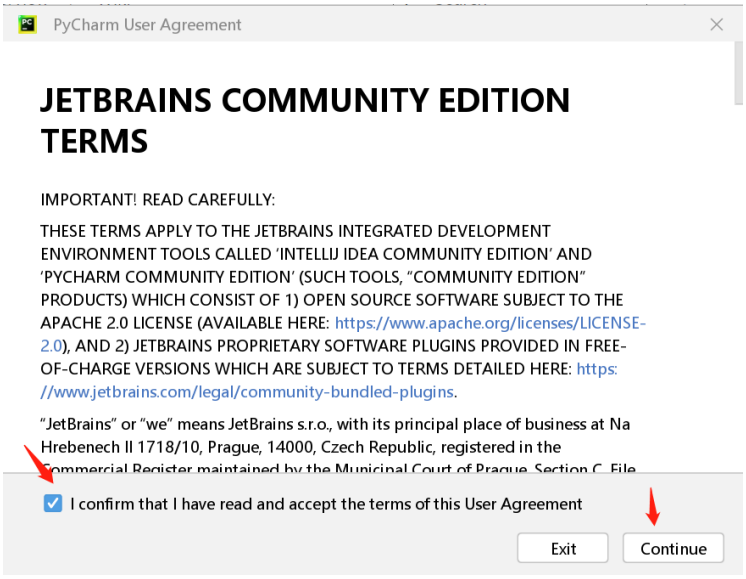
1) Decompose zip package.
2) Source code structure of '**WSC Simulation Challenge 2023 (Python)**'.



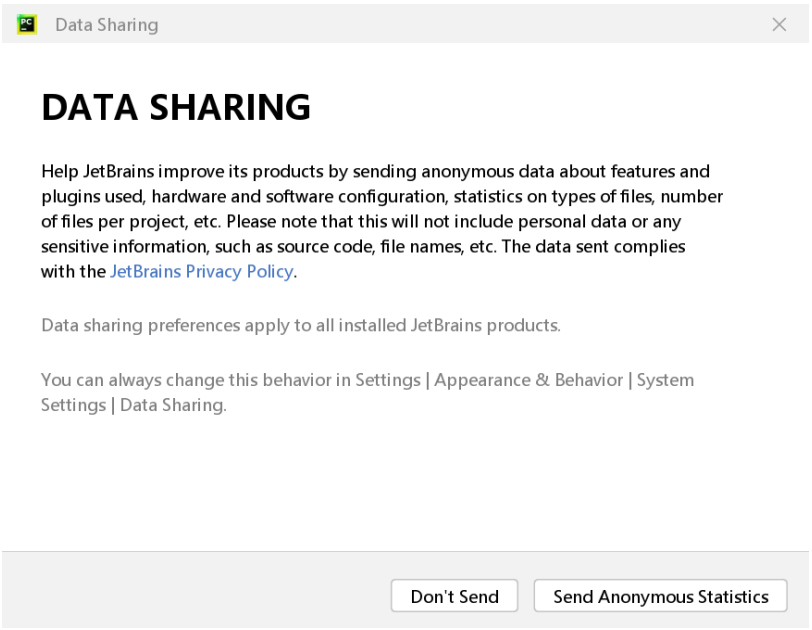
conf folder: includes scenario files (XML Document)
strategy folder: includes strategy file (Python File)
program.py: run file (Python File)
requirements.txt: required python packages

2.6 Add Python Interpreter (Python)

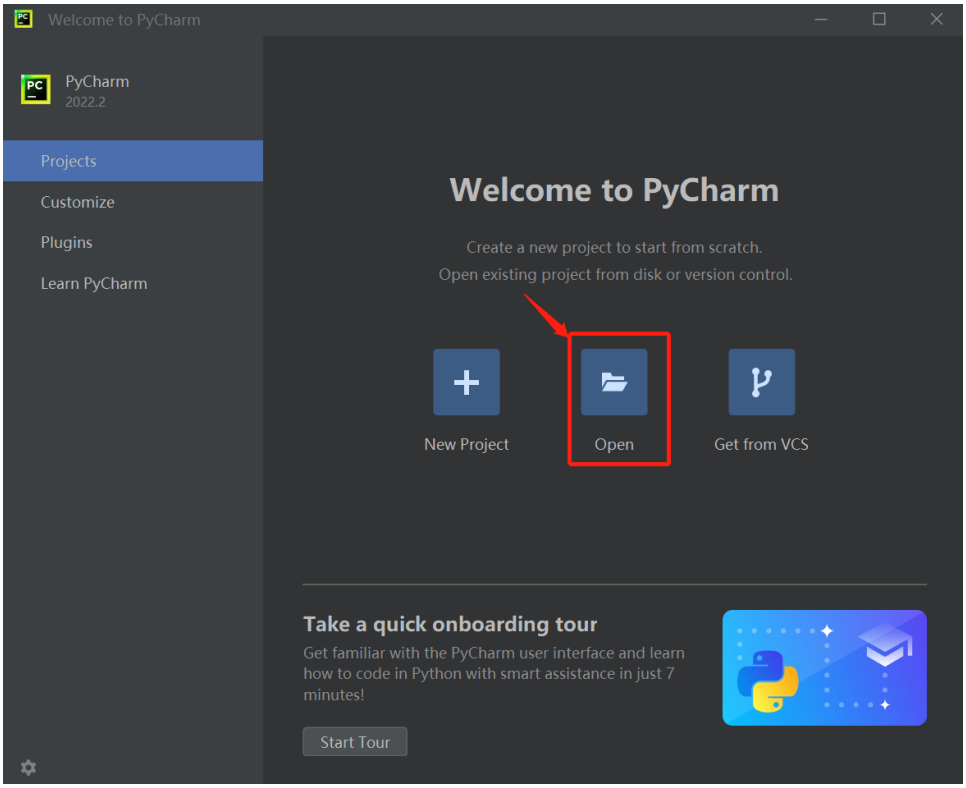
1) Open PyCharm, confirm user agreement and click **Continue**



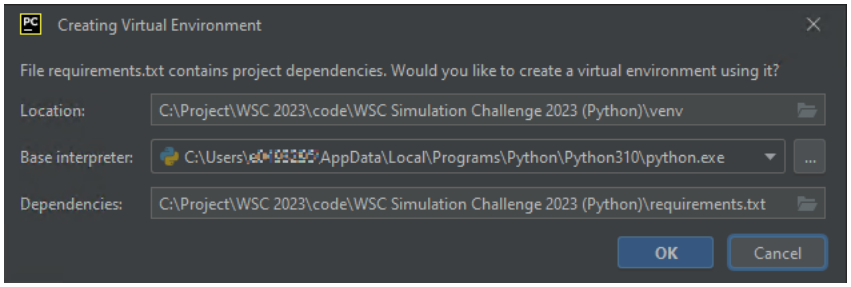
2) Click either option of your choice



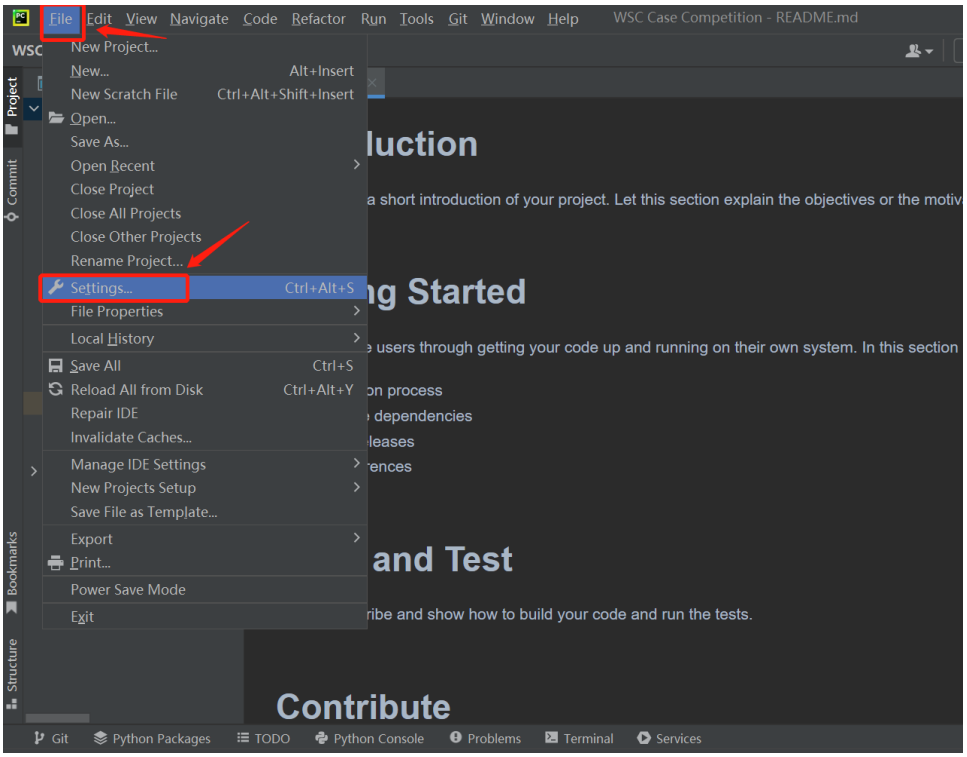
3) Click **Open**, and open "WSC Simulation Challenge 2023 (Python)" folder directly



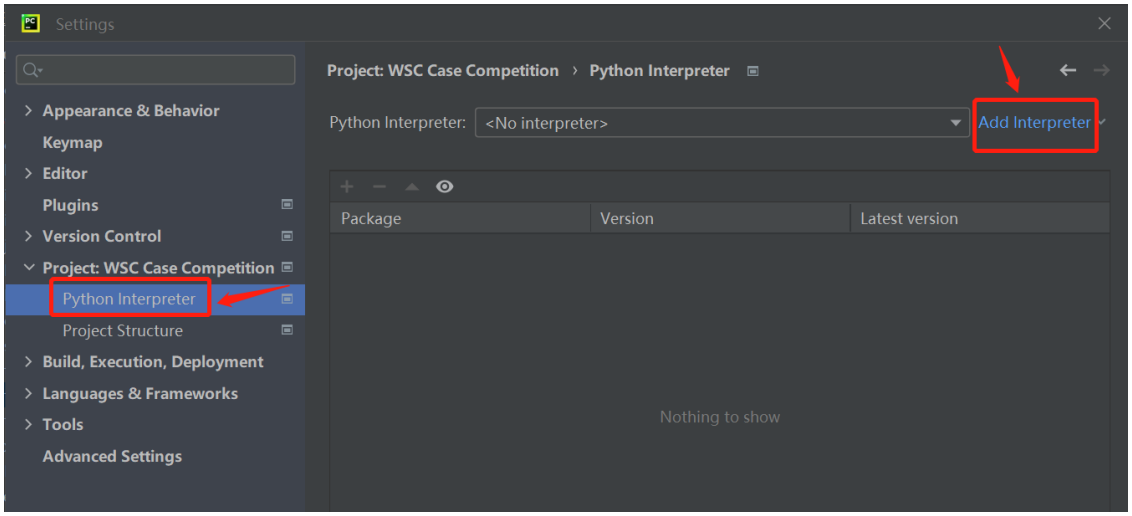
4) If PyCharm shows the following option, you can click **OK** to automatically create a virtual environment and install required python packages. This process may take some time. Otherwise, continue with the following steps.



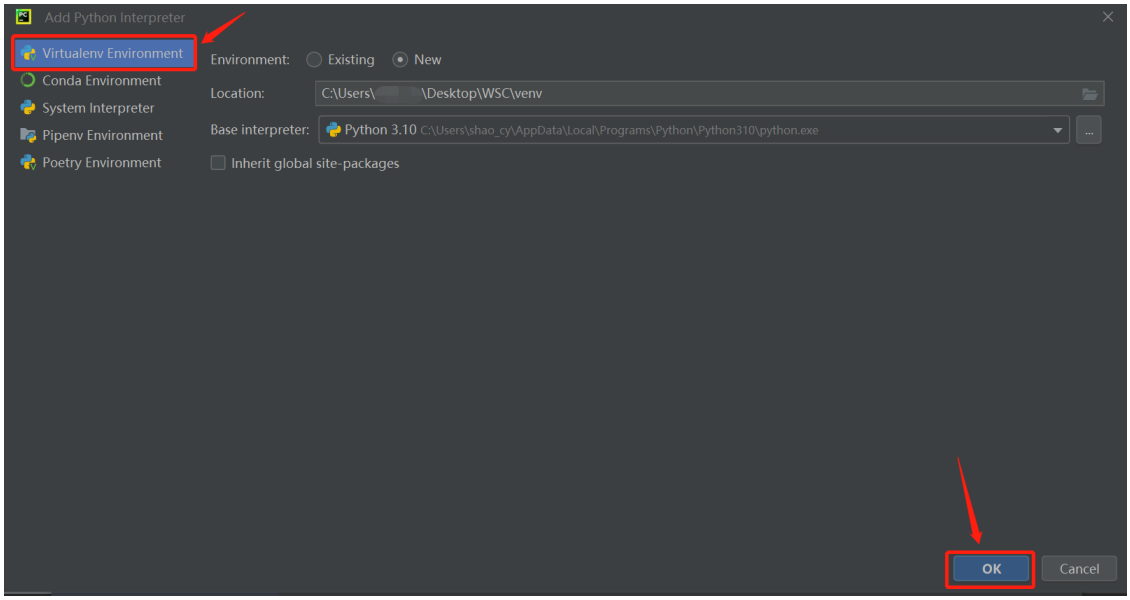
5) Click **File** and click **Settings**



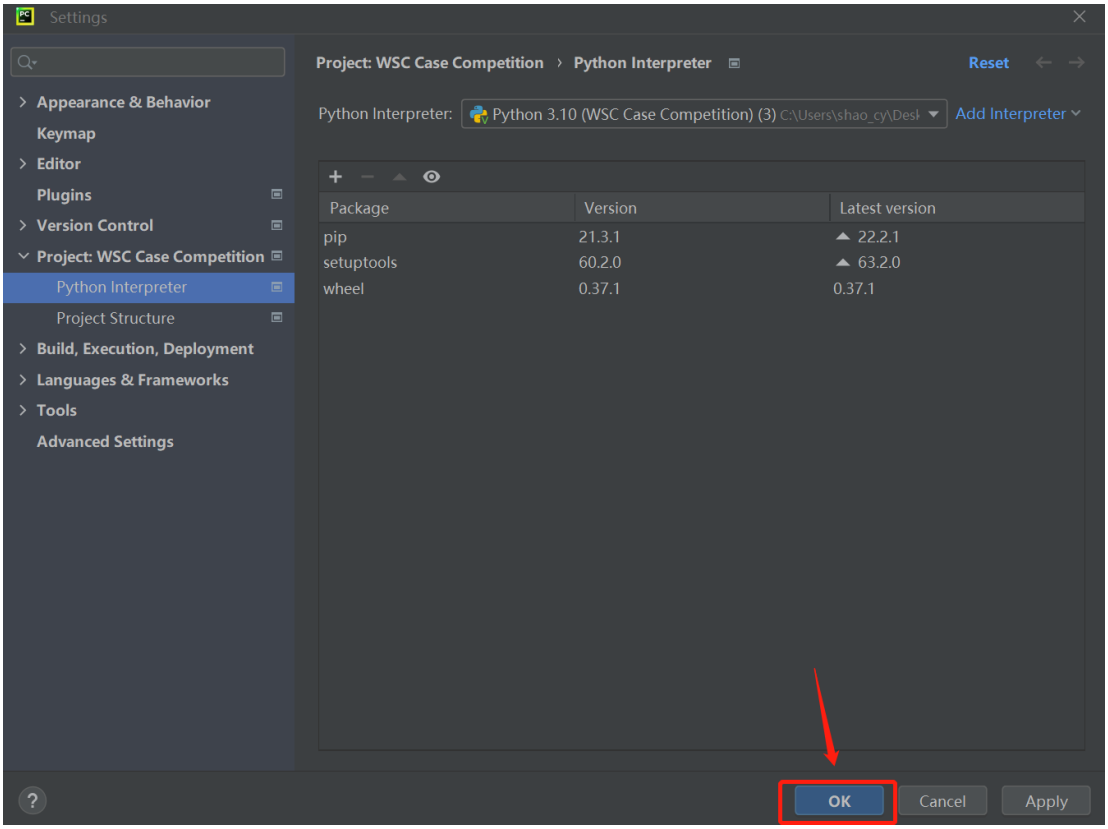
6) Open **Project** tab and click **Python Interpreter**, and click **Add Interpreter**



7) Choose **Virtualenv Environment**, click **OK**

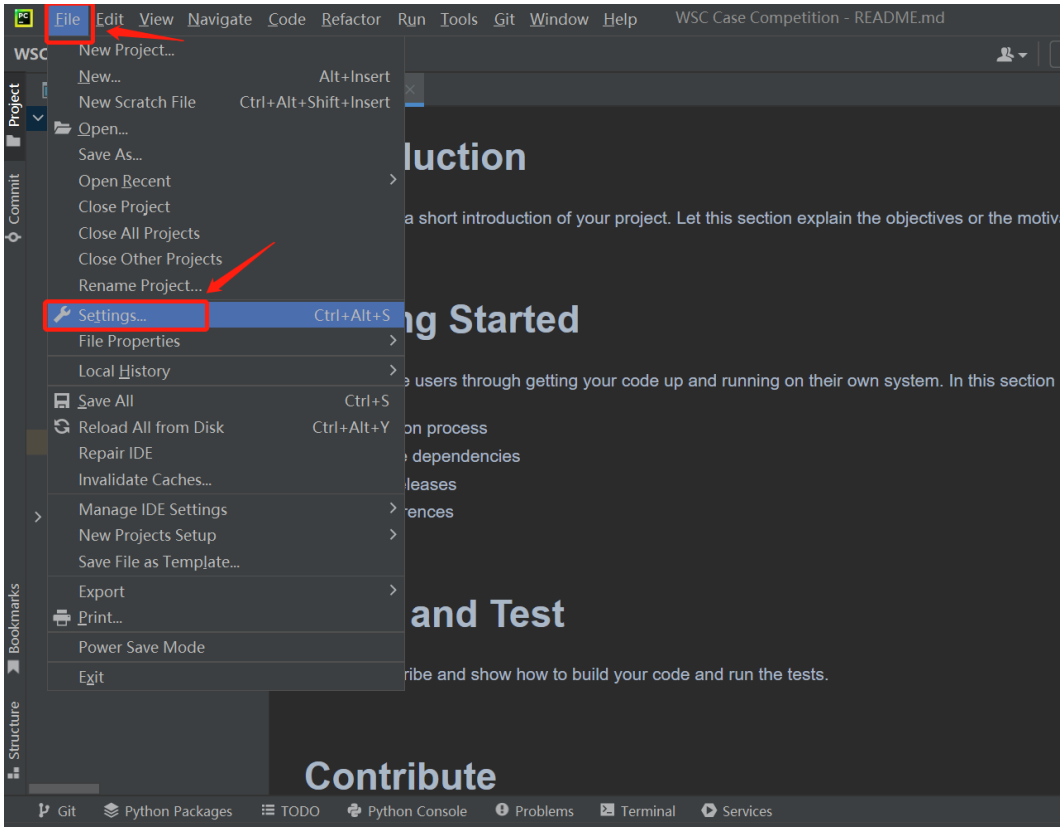


8) Click **OK**

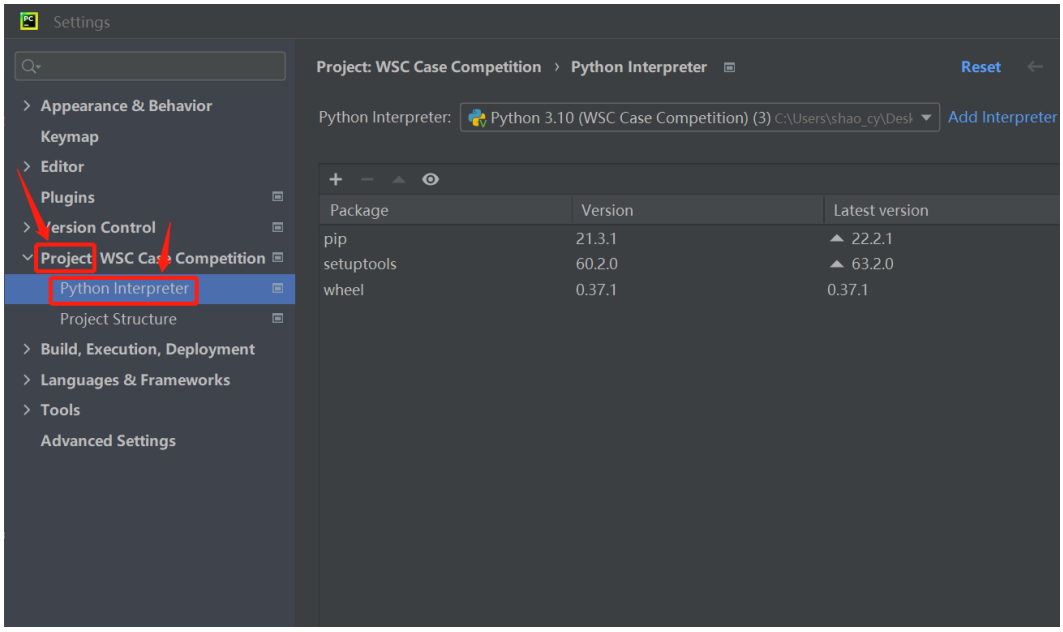


2.7 Add Python Packages Installation (Python)

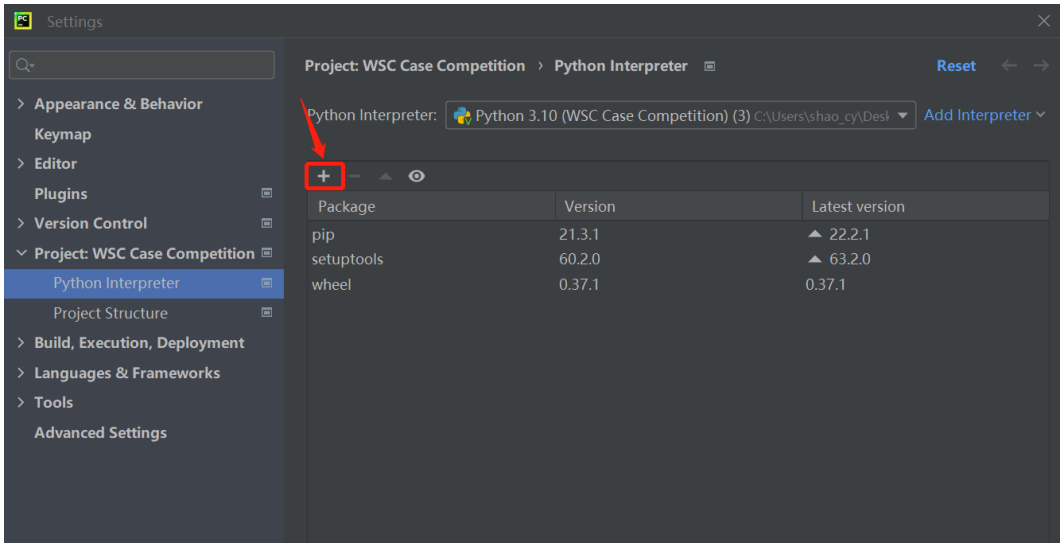
1) Click **File** and click **Settings**



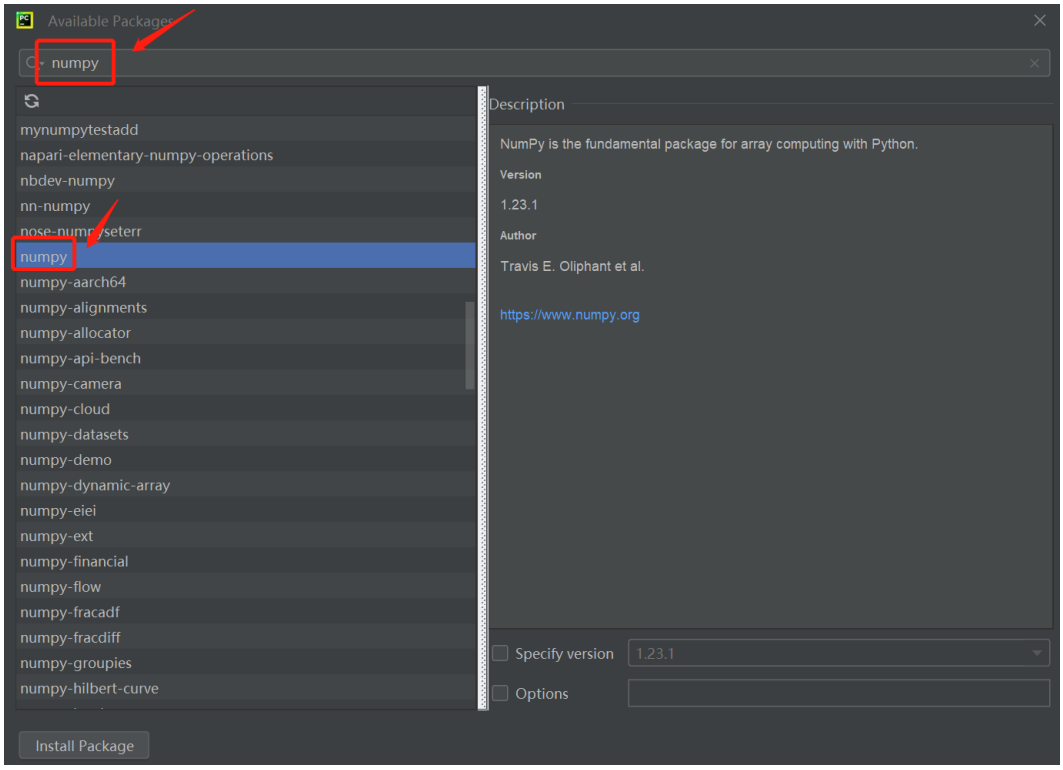
2) Open **Project** tab and click **Python Interpreter**



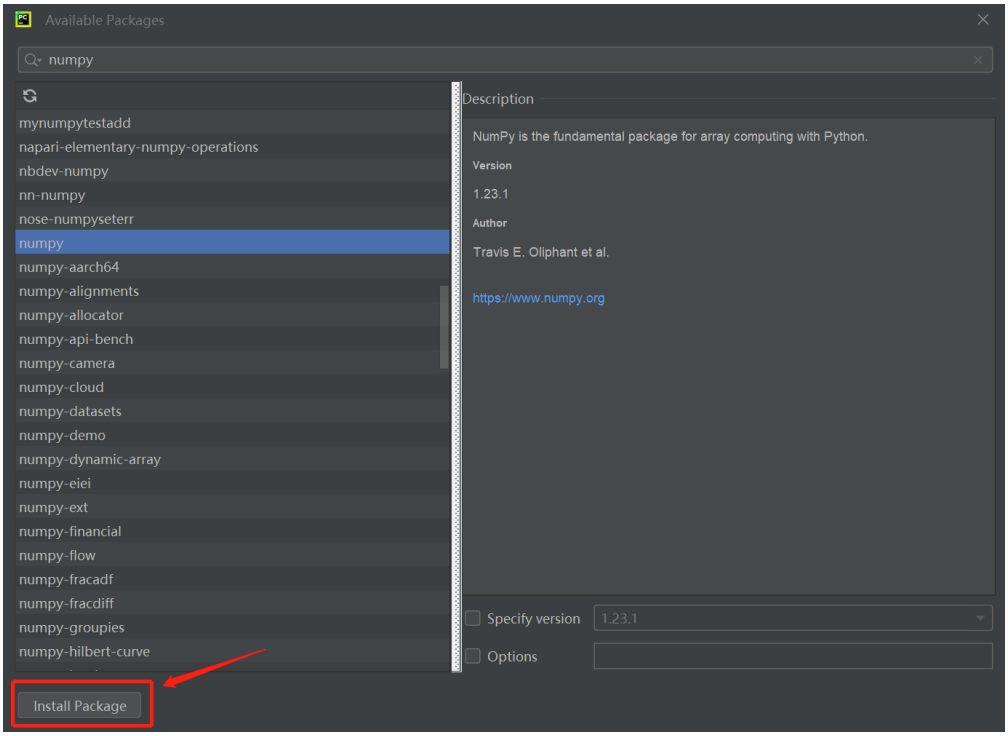
3) Click + button



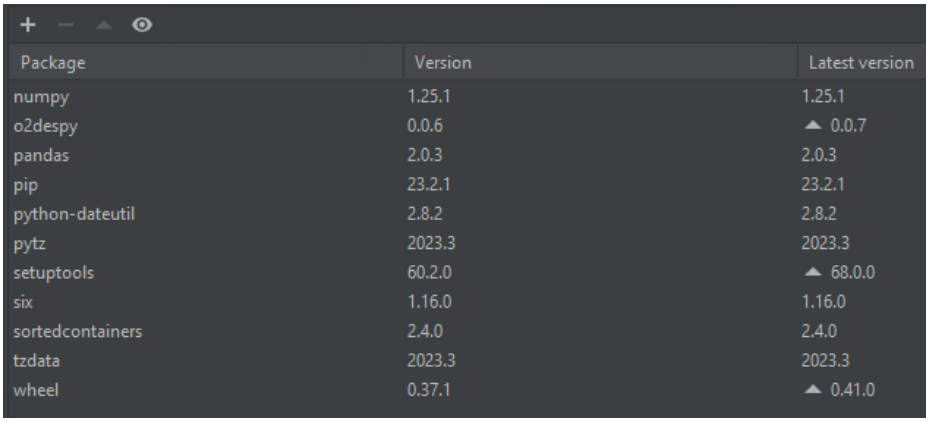
4) Search for the packages to be installed



5) Click **Install Package** button to start the installation process



6) For this competition, repeat step 4 - step 5 to install packages included in "requirements.txt" correspondingly
7) Click **OK** to finish all packages installation



Package	Version	Latest version
numpy	1.25.1	1.25.1
o2despy	0.0.6	▲ 0.0.7
pandas	2.0.3	2.0.3
pip	23.2.1	23.2.1
python-dateutil	2.8.2	2.8.2
pytz	2023.3	2023.3
setuptools	60.2.0	▲ 68.0.0
six	1.16.0	1.16.0
sortedcontainers	2.4.0	2.4.0
tzdata	2023.3	2023.3
wheel	0.37.1	▲ 0.41.0

Chapter 3

Data in the Discrete-Event Simulation Model

3.1 Entities

There are seven entities involved in this model, Fab, Workstation, Product Type, Step, Qt Loop, Lot and Running QTL (full description of these terms will be provided in subsections 3.1.1 and 3.1.2). Each is represented by a class and has its own set of attributes. As shown in Figure2, a fabrication consists of multiple workstations. During fabrication, Lots undergo multiple steps for completion and are transported to workstations that contain multiple tools performing the required process steps. The Lots that are being processed in each step are referred to as the work-in-process (WIP) of that step. Different types of Lots have different steps and Qt Loops. Note that re-entry of Lots to previous workstations is possible due to the complex nature of semiconductor manufacturing. An Entity Relationship Diagram (ERD) is shown in Figure 3 to give an overview of the relationships among the entities and the attributes associated with them.

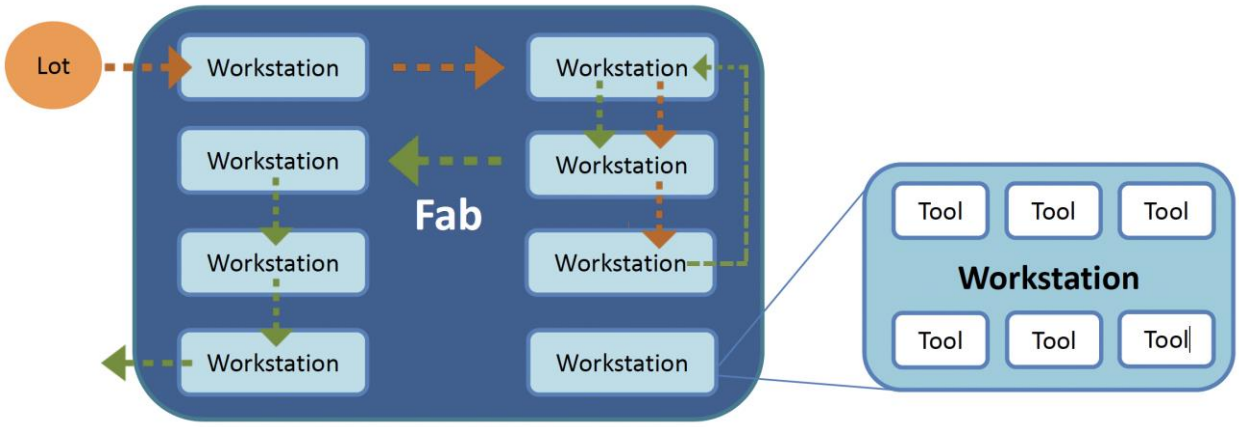


Figure 2 - Lot Process Flow

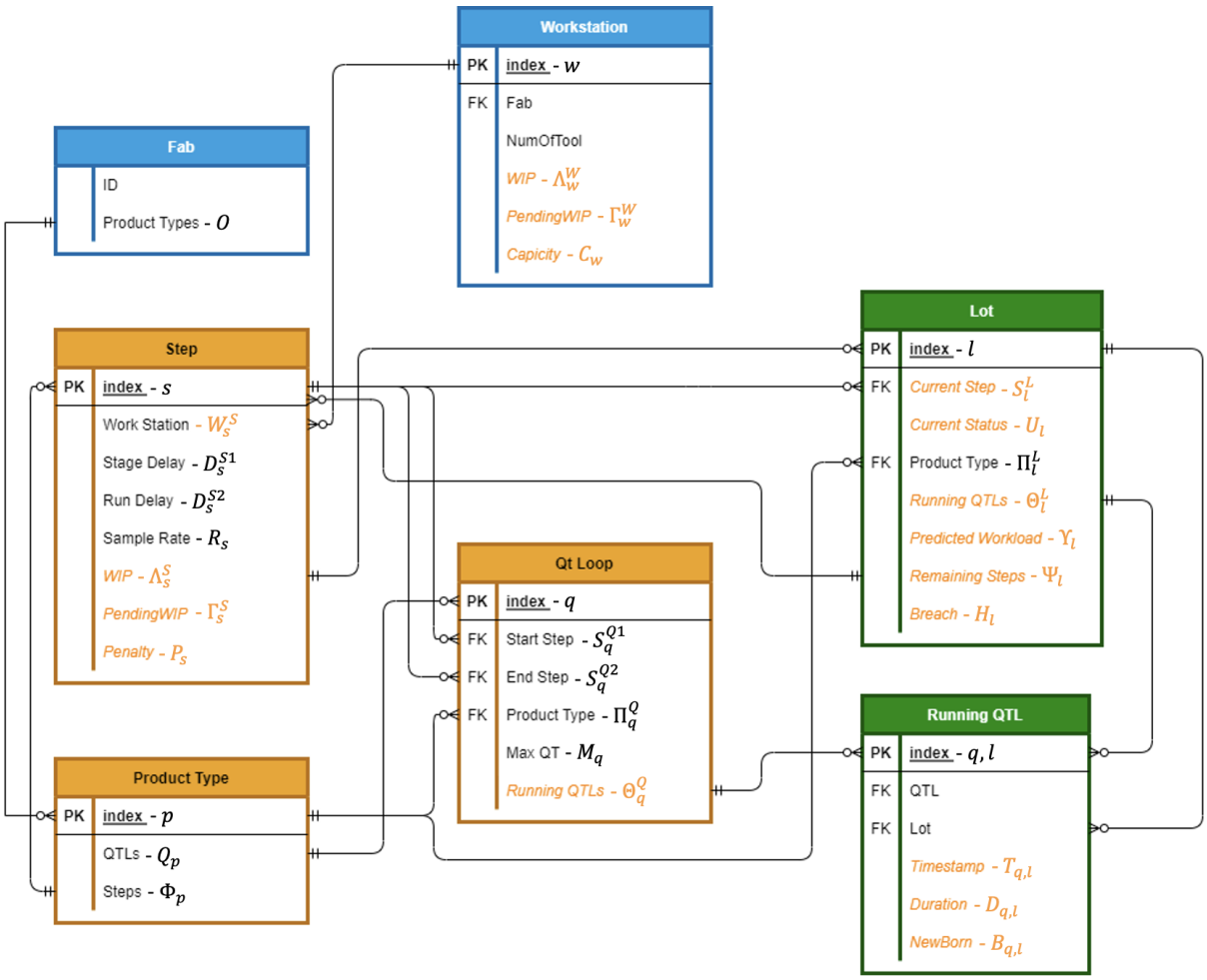


Figure 3 - Entity Relationship Diagram (ERD)

Note:

1) PK: short term for primary key, which uniquely identifies the entity object.
2) Static attribute: entity attributes that do not change overtime. Static attributes are owned by the class itself (in black color).
3) Dynamic attribute (italic): entity attributes that can change overtime (in orange color).

3.1.1 Entity Relationship

1. The Fab can handle zero to many Product Types.

2. A Product Type needs to go through zero to many Steps.

   A Step corresponds to one Product Type.

3. A Workstation can handle zero to many Steps.

   A Step corresponds to one Workstation.

4. A Product Type follows zero to many Qt Loops.

   A Qt Loop corresponds to one Product Type.

5. A Qt Loop has one start Step and one end Step.

   A Step can be the start or end step of zero to many Qt Loops.

6. A Product Type corresponds to zero to many Lots.

   A Lot belongs to one Product Type.

7. In a Step, there are zero to many Lots (WIP).

   A Lot can only be in one Step at any point in time.

   A Lot has zero to many remaining Steps to go through.

8. A Lot has zero to many Running QTLs to record the corresponding Qt Loops.

   A Running QTL corresponds to One Lot.

9. A Qt Loop corresponds to zero to many Running QTLs.

   A Running QTLs corresponds to one Qt Loop.

3.1.2 Entity Attributes and Definitions

The details about the entity attributes and their definitions are explained below.

Table 1 lists the attributes of Fab which are all static.

Table 1 - Attributes of Fab

Attribute Name	Type	Description
ID	string	The ID of the Fab.
Product Types	List<Product Type>	The list of product types the Fab handles. Note that in the code, Product Types are controlled by the input file, so this attribute is omitted from the Fab itself.

Table 2 summarizes the attributes of Workstation.

Table 2 - Attributes of Workstation

Attribute Name	Type	Description
index	string	The primary key of the Workstation.
NumOfTool	int	The number of tools in the Workstation.
WIP	List<WIP>	The list of current WIP in the Workstation.
PendingWIP	List<WIP>	The list of WIP whose status has changed from waiting to stage, but has not yet occupied a tool in the Workstation.
Capicity	int	The capacity of the Workstation. Its initial value is equal to NumOfTool.

Table 3 gives information on the attributes of Product Type, which are all static.

Table 3 - Attributes of Product Type

Attribute Name	Type	Description
index	string	The primary key of the Product Type
QTLs	List<Qt Loop>	The list of the Qt Loops that the Product Type needs to follow.
Steps	List<Step>	The list of the Steps that the Product Type needs to go through.

Table 4 summarizes the attributes of Step.

Table 4 - Attributes of Step

Attribute Name	Type	Description
index	string	The primary key of Step.
Work Station	Workstation	The Workstation where the Step is processed.

Stage Delay	double	The time required for WIP to occupy a tool and end the Stage status in this Step.
Run Delay	double	The time required for WIP to complete the Run in this Step.
Sample Rate	double	The probability that the Step will be executed. When a Lot reaches the Step, a random number is generated between 0 and 1. If the random number is less than Sample Rate, the Lot needs to go through the step, otherwise the Step is skipped.
WIP	List<WIP>	The list of current WIP in the Step.
PendingWIP	List<WIP>	The list of WIP whose status is waiting in the Step.
Penalty	double	If a breach happens at a certain step, the penalty will be incurred where the one unit of WIP will be evenly distributed and anchored permanently across the current and all subsequent steps.

Table 5 summarizes the attributes of Qt Loop.

Table 5 - Attributes of Qt Loop

Attribute Name	**Type**	**Description**
index	string	The primary key of Qt Loop.
Start Step	Step	The start Step of the Qt Loop.
End Step	Step	The end Step of the Qt Loop.
Product Type	Product Type	The Product Type follows the Qt Loop.
Max QT	double	The maximum time that WIP can spend between the R step status of the source Workstation and the R step status of the destination Workstation in the Qt Loop.
Running QTLs	List<Running QTL>	The list of current Running QTLs corresponding to the Qt Loop.

Table 6 summarizes the attributes of Lot. Lots arrive to the system following an exponential distribution.

Table 6 - Attributes of Lot

Attribute Name	Type	Description
index	string	The primary key of Lot.
Product Type	Product Type	The Product Type to which the Lot belongs.
Current Step	Step	The current Step where the Lot is.
Current Status	enum {None, Waiting, Stage, Run}	The current status of the Lot.
Running QTLs	List<Running QTL>	The list of current Running QTLs used to record the situation of the Qt Loops currently followed by the Lot.
Predicted Workload	List<double>	The list of predicted workloads of the Lot. Actually, since there is more than one Qt Loop that needs to be followed, there is more than one predicted workload. In the code, a dictionary is used to record these predicted workloads. This attribute is not used.
Remaining Steps	Queue<Step>	The remaining steps that the Lot needs to go through.
Breach	bool	It records whether the Lot has breached.

Table 7 summarizes the attributes of Running QTL.

Table 7 - Attributes of Running QTL

Attribute Name	Type	Description
index	string	The primary key of Running QTL.
QTL	Qt Loop	The Qt Loop to which the Running QTL corresponding.
Lot	Lot	The Lot to which the Running QTL corresponding.

Timestamp	DateTime	The moment when the corresponding Lot occupies the tool in the source workstation of the corresponding Qt Loop and officially starts the stage.
Duration	double	For the corresponding Lot in the corresponding Qt Loop, the duration of time from the end of the Run status of the source workstation.
NewBorn	bool	It records whether the Running QTL is newly generated. Timestamp can be updated only when NewBorn is true.

3.2 Event

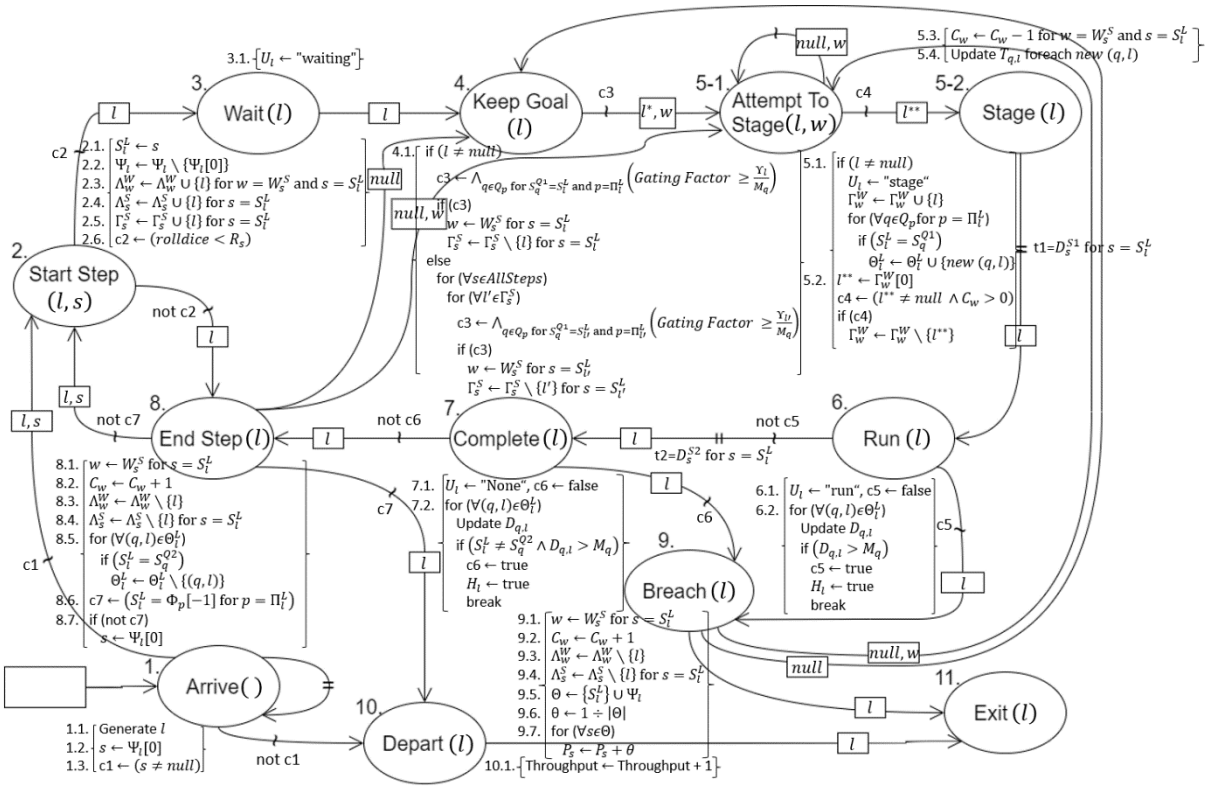


Figure 4 - Event Graph for the Model

The simulation model uses 12 events to describe the entire process, as shown in the event graph (EG). Each event is scheduled with or without a parameter, whose primary key index is shown in the bracket beside the event name. The primary key index can be found in the entity tables in Section 3.1. Take “Arrive()” as an example, Arrive() event is scheduled without a parameter. Conversely, Start Step (l, s) event is scheduled

with two parameter a Lot l and a Step s. The arrow "→" indicates the triggering relationship between events, i.e., an event is scheduled by another event. "cx ~" indicates the condition required to schedule the next event. For example, Start_Step (l, s) event schedules Wait(l) event under condition c2 and passes in parameter l. What's more, "||" represents a time delay in scheduling the next event. For example, Stage(l) event starts at time t, and it schedules Run(l) event at time t + t1. In addition, in each event, certain system states or entity attributes might be changed. The following sections will describe each event in detail.

Our system's events are: Arrive, Start Step, Wait, Keep Goal, Attempt to Stage, Stage, Run, Complete, End Step, Breach, Depart, and Exit. A Lot needs to go through Start Step, Wait, Keep Goal, Attempt to Stage, Stage, Run, Complete, and End Step to complete a step. When the Lot reaches the Run and Complete events, we will respectively calculate whether the current time spent on this Lot exceeds its Max QTs. If so, it will enter the Breach event.

The gating factor is involved in the event Keep Goal. This event determines whether Lots can go from the Waiting status to the Stage status. This is where our competitors need to focus.

3.2.1 Arrive

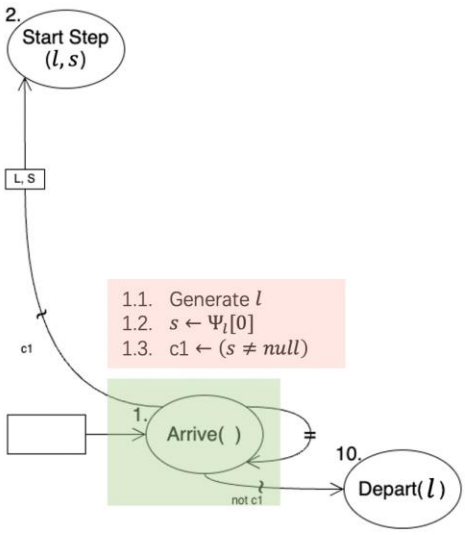


Figure 5 - EG for Arrive

1. In this event, Lot $\boldsymbol{l}$ is generated, i.e., arrives to the system.

2. Obtain the first step $\boldsymbol{s}$ from the remaining step set $\boldsymbol{\Psi_l}$ of the Lot $\boldsymbol{l}$.

3. Check if $\boldsymbol{s}$ is a null value. If $\boldsymbol{s}$ is not null, Start Step event is scheduled; otherwise, it means that the Lot does not need to go through any steps, then Depart event is scheduled.

The Arrive event schedules itself at different time intervals, and each schedule means that a new Lot will arrive to the system at a later time. In the code, the inter-arrival time follows an exponential distribution. Note that different seeds can lead to different inter-arrival times.

3.2.2 Start Step

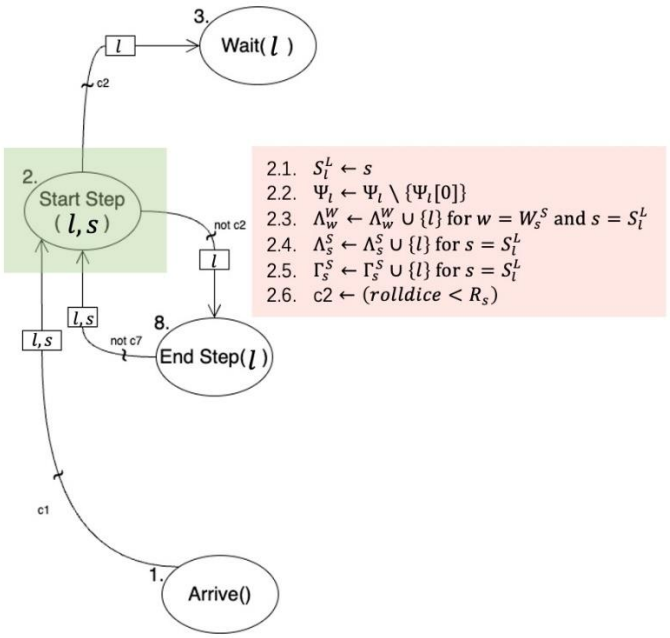


Figure 6 - EG for Start Step

1. Update current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$ with the step $\boldsymbol{s}$.

2. Remove the first step from the remaining step set Ψ_l of the Lot l.

3. Adds the lot $\boldsymbol{l}$ to the list of WIP $\boldsymbol{\Lambda_w^W}$ of the workstation $\boldsymbol{W_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

4. Add the Lot $\boldsymbol{l}$ to the list of WIP $\boldsymbol{\Lambda_s^S}$ of the current step $\boldsymbol{s}$.

5. Add the Lot $\boldsymbol{l}$ to the list of pending WIP $\boldsymbol{\Gamma_s^S}$ of the current step $\boldsymbol{s}$.

6. Determine whether to schedule Wait event based on a comparison between a random number and the sample rate $\boldsymbol{R_s}$ of the step $\boldsymbol{s}$. If the random number (between 0 and 1) is less than the sample rate $\boldsymbol{R_s}$, it will schedule Wait event; otherwise, it will schedule End Step event.

3.2.3 Wait

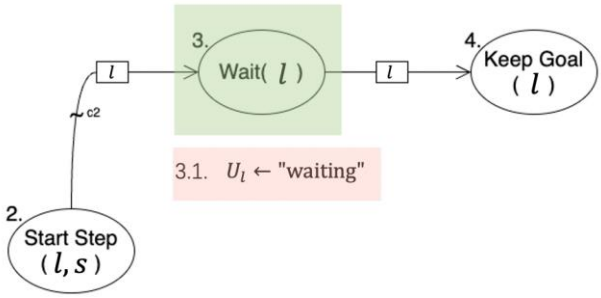


Figure 7 - EG for Wait

1. Update the status $\boldsymbol{U_l}$ of the Lot $\boldsymbol{l}$ with "waiting". In the end, Keep Goal event is scheduled.

3.2.4 Keep Goal

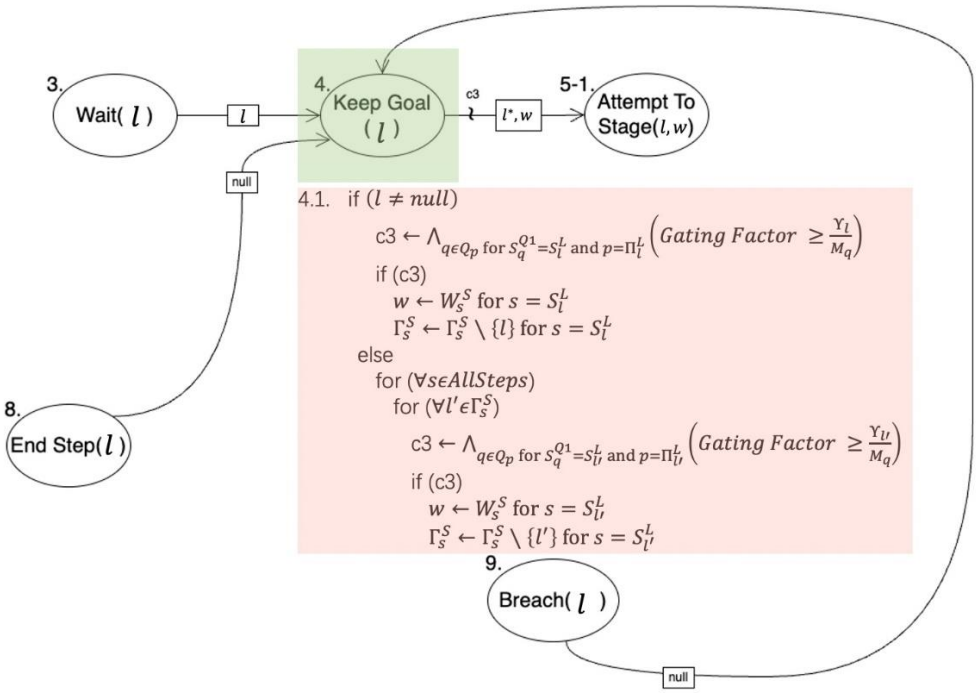


Figure 8 - EG for Keep Goal

1. (1) If the parameter Lot $\boldsymbol{l}$ is not null, based on the Product Type $\boldsymbol{\Pi_l^L}$ corresponding to the Lot $\boldsymbol{l}$, find each Qt Loop $\boldsymbol{q}$ in the Qt Loop set $\boldsymbol{Q_p}$ corresponding to $\boldsymbol{\Pi_l^L}$, with the same start step $\boldsymbol{S_q^{Q1}}$ as the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$. The value of **c3** is only true when the gating factor logic allows the Lot to enter every Qt Loop found. If **c3** is true, assign $\boldsymbol{w}$ a value of the workstation $\boldsymbol{W_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$. $\boldsymbol{w}$ will be passed as a parameter to the next event. And remove the Lot $\boldsymbol{l}$ from the list of pending WIP $\boldsymbol{\Gamma_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$. Then Attempt To Stage event is scheduled.

(2) If the parameter Lot $\boldsymbol{l}$ is null, it means that Keep Goal event is scheduled by End Step event or Breach event. Due to changes in the current workload of the system, we need to make the same judgments and operations as Lot $\boldsymbol{l}$ in (1) for all Lots in all steps.

3.2.5 Stage

3.2.5.1 Attempt to Stage

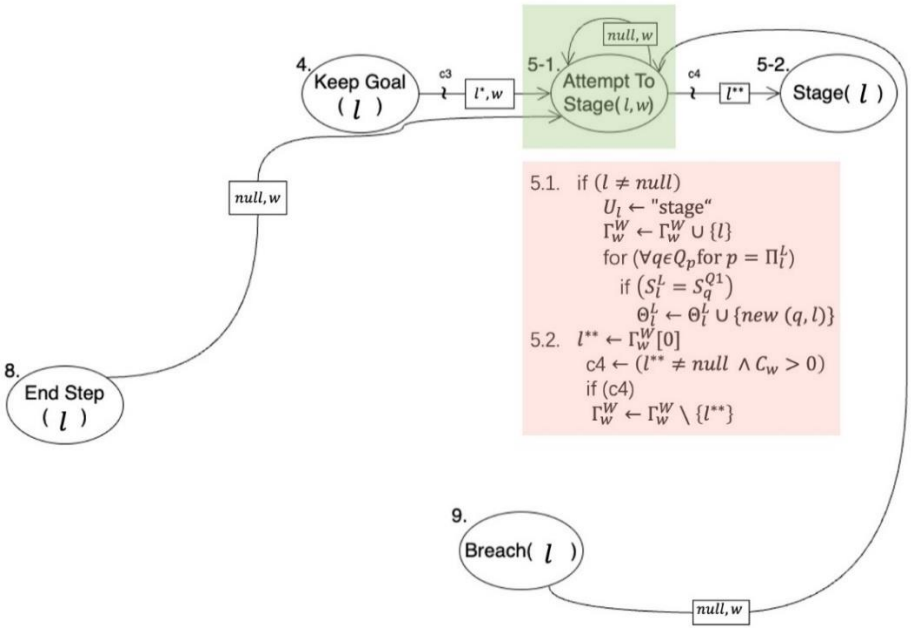


Figure 9 - Event Graph for Attempt to Stage

1. If the parameter Lot $\boldsymbol{l}$ is not null, update the status $\boldsymbol{U_l}$ of the Lot $\boldsymbol{l}$ with "stage" and add the Lot $\boldsymbol{l}$ to the Pending WIP list $\boldsymbol{\Gamma_w^W}$ of the parameter workstation $\boldsymbol{w}$. Then based on the Product Type $\boldsymbol{\Pi_l^L}$ corresponding to the Lot $\boldsymbol{l}$, find each Qt Loop $\boldsymbol{q}$ in the Qt Loop set $\boldsymbol{Q_p}$ corresponding to $\boldsymbol{\Pi_l^L}$, with the same start step $\boldsymbol{S_q^{Q1}}$

as the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$. Create a new Running QTL based on each found Qt Loop $\boldsymbol{q}$ and Lot $\boldsymbol{l}$ and add it to the Running QTL list $\boldsymbol{\Theta_l^L}$ of Lot $\boldsymbol{l}$.

2. Obtain the first step $\boldsymbol{l}^{**}$ from the Pending WIP list $\boldsymbol{\Gamma_w^W}$ of the parameter workstation $\boldsymbol{w}$. If $\boldsymbol{l}^{**}$ is not null and the capacity $\boldsymbol{C_w}$ of workstation $\boldsymbol{w}$ is greater than 0, then remove $\boldsymbol{l}^{**}$ from the Pending WIP list $\boldsymbol{\Gamma_w^W}$ and schedule Stage event.

3.2.5.2 Stage

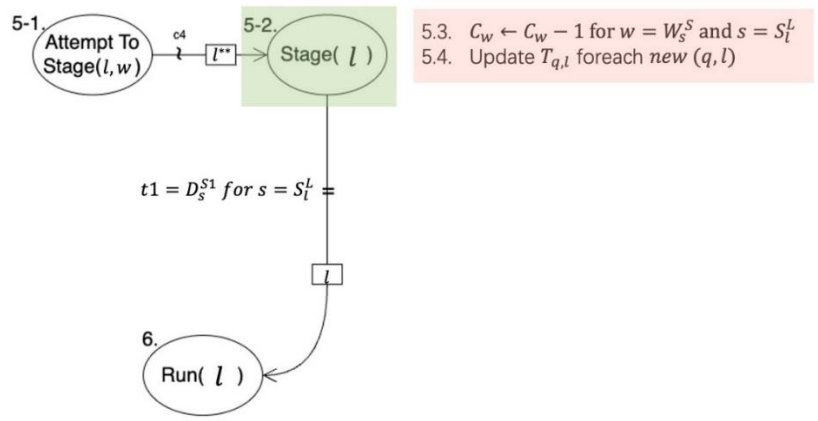


Figure 10 - EG for Stage

1. Decrease the capacity $\boldsymbol{C_w}$ of the workstation $\boldsymbol{W_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the parameter Lot $\boldsymbol{l}$ by one.

2. Update the Timestamp $\boldsymbol{T_{q,l}}$ for each new running QTL $(\boldsymbol{q}, \boldsymbol{l})$.

Run event is scheduled after $t1$ time which is equal to the stage delay $\boldsymbol{D_s^{S1}}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

3.2.6 Run

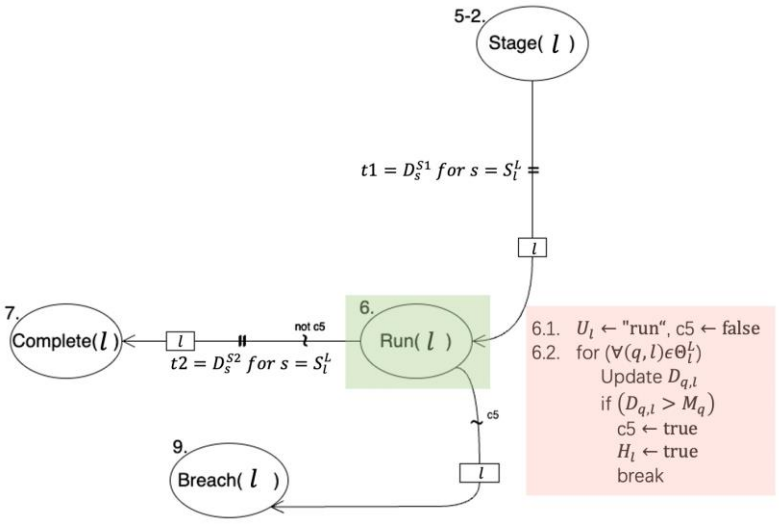


Figure 11 - Flow Chart for Run

1. Update the status $\boldsymbol{U_l}$ of the Lot $\boldsymbol{l}$ with "run" and initialize c5 to false.

2. For each Running QTL $(\boldsymbol{q}, \boldsymbol{l})$ in the Running QTL list $\boldsymbol{\Theta_l^L}$ of Lot $\boldsymbol{l}$, update the duration $\boldsymbol{D_{q,l}}$. If $\boldsymbol{D_{q,l}}$ is greater than the Max QT $\boldsymbol{M_q}$ of Qt Loop $\boldsymbol{q}$, then update c5 to true and the breach $\boldsymbol{H_l}$ of Lot $\boldsymbol{l}$ to true and break the loop.

If c5 is true, then Breach event is scheduled. Otherwise, Complete event is scheduled after t2 time which is equal to the run delay $\boldsymbol{D_s^{S2}}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

3.2.7 Complete

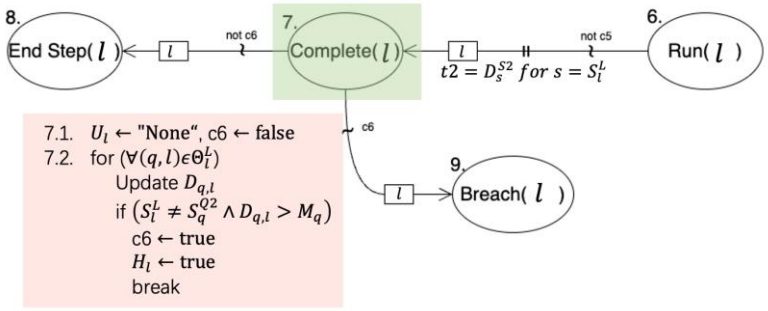


Figure 12 - EG for Complete

1. Update the status $\boldsymbol{U_l}$ of the Lot $\boldsymbol{l}$ with "None" and initialize c6 to false.

2. For each Running QTL $(\boldsymbol{q}, \boldsymbol{l})$ in the Running QTL list $\boldsymbol{\Theta_l^L}$ of Lot $\boldsymbol{l}$, update the duration $\boldsymbol{D_{q,l}}$. If the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$ is not equal to the end step $\boldsymbol{S_q^{Q2}}$ of Qt Loop $\boldsymbol{q}$ and $\boldsymbol{D_{q,l}}$ is greater than the Max QT $\boldsymbol{M_q}$ of Qt Loop $\boldsymbol{q}$, then update c6 to true and the breach $\boldsymbol{H_l}$ of Lot $\boldsymbol{l}$ to true and break the loop.

If c6 is true, then Breach event is scheduled. Otherwise, End Step event is scheduled.

3.2.8 End Step

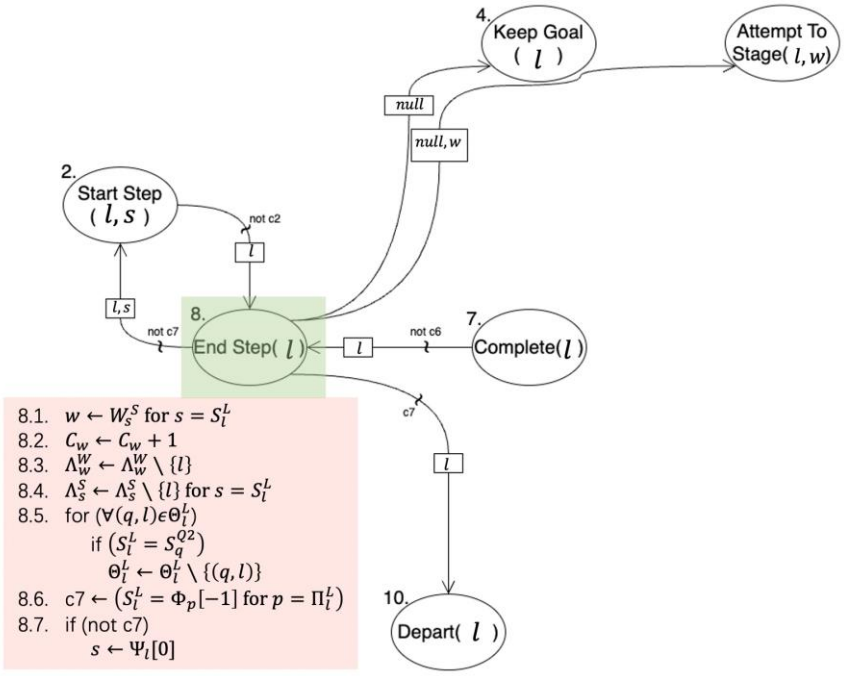


Figure 13 – EG for End Step

1. Use $\boldsymbol{w}$ to represent the workstation $\boldsymbol{W_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

2. Increase the capacity $\boldsymbol{C_w}$ of the workstation $\boldsymbol{w}$ by one.

3. Remove the Lot $\boldsymbol{l}$ from the WIP list $\boldsymbol{\Lambda_w^W}$ of the workstation $\boldsymbol{w}$.

4. Remove the Lot $\boldsymbol{l}$ from the WIP list $\boldsymbol{\Lambda_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

5. For each Running QTL $(\boldsymbol{q}, \boldsymbol{l})$ in the Running QTL list $\boldsymbol{\Theta_l^L}$ of Lot $\boldsymbol{l}$, if the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$ is equal to the end step $\boldsymbol{S_q^{Q2}}$ of Qt Loop $\boldsymbol{q}$, then remove Running QTL $(\boldsymbol{q}, \boldsymbol{l})$ from the Running QTL list $\boldsymbol{\Theta_l^L}$ of Lot $\boldsymbol{l}$.

6. Use c7 to indicate whether the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$ is the same as the last step in the step list $\boldsymbol{\Phi_p}$ of the product type $\boldsymbol{\Pi_l^L}$ of the Lot $\boldsymbol{l}$.

7. If c7 is false, obtain the first step $\boldsymbol{s}$ from the remaining step set $\boldsymbol{\Psi_l}$ of the Lot $\boldsymbol{l}$ and schedule Start Step event. Otherwise, Depart event is scheduled.

In the end, Keep Goal and Attempt to Stage events are scheduled.

3.2.9 Breach

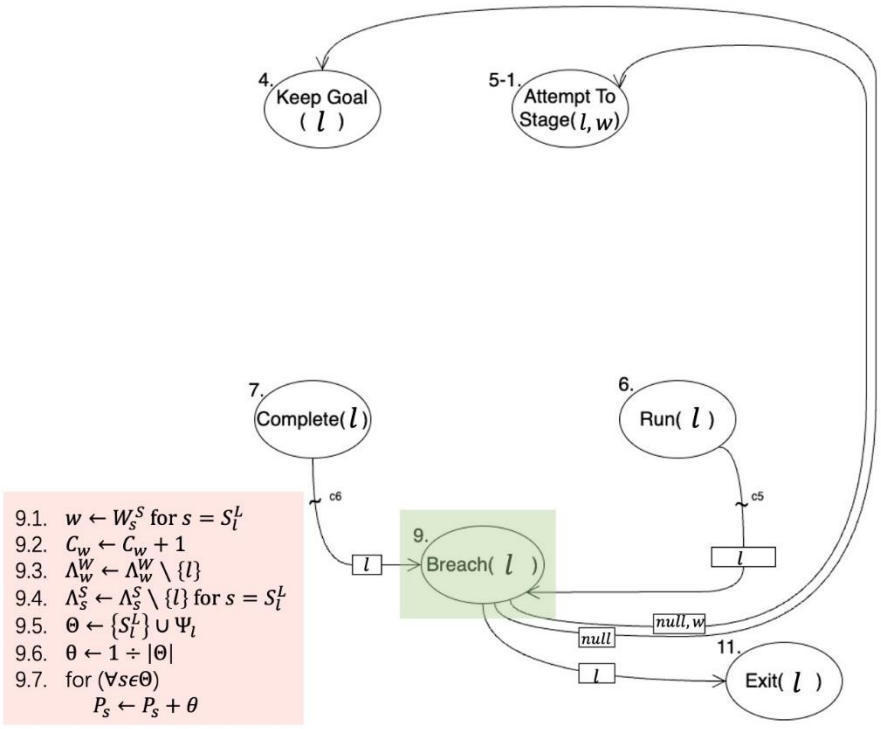


Figure 14 - EG for Breach

1. Use $\boldsymbol{w}$ to represent the workstation $\boldsymbol{W_s^S}$ of the current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

2. Increase the capacity $\boldsymbol{C_w}$ of the workstation $\boldsymbol{w}$ by one.

3. Remove the Lot $\boldsymbol{l}$ from the WIP list $\boldsymbol{\Lambda_w^W}$ of the workstation $\boldsymbol{w}$.

4. Remove the Lot $\boldsymbol{l}$ from the WIP list $\boldsymbol{\Lambda_s^S}$ of the current step current step $\boldsymbol{S_l^L}$ of the Lot $\boldsymbol{l}$.

5. Add the current step $\boldsymbol{S_l^L}$ to the remaining step set $\boldsymbol{\Psi_l}$ of the Lot $\boldsymbol{l}$ to obtain the list $\boldsymbol{\Theta}$.

6. Calculate the penalty value $\boldsymbol{\theta}$, which is the unit penalty divided by the number of elements in the list $\boldsymbol{\Theta}$.

7. For each step $\boldsymbol{s}$ in the list $\boldsymbol{\Theta}$, increase their penalty $\boldsymbol{P_s}$ by $\boldsymbol{\theta}$.

In the end, Keep Goal, Attempt to Stage and Exit events are scheduled.

3.2.10 Depart

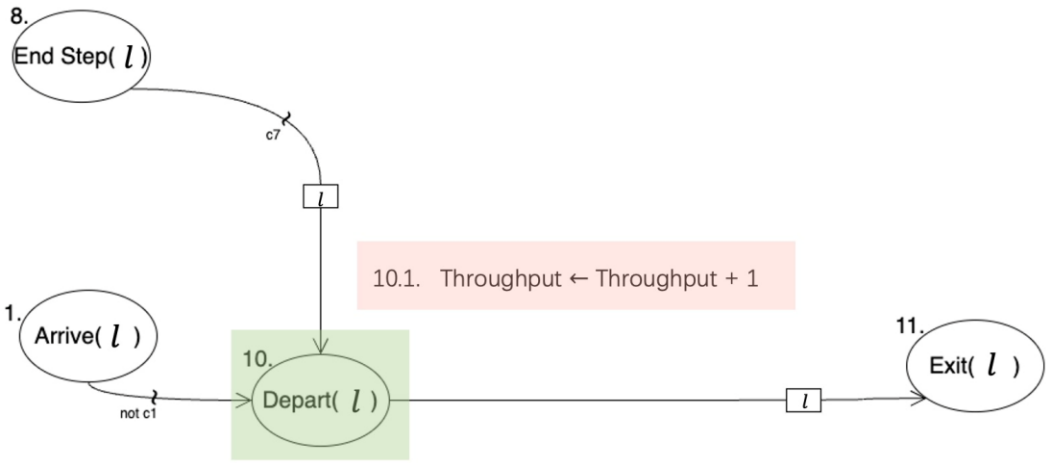


Figure 15 – EG for Depart

1. Throughput increase by 1.

In the end, Exit event is scheduled.

3.2.11 Exit

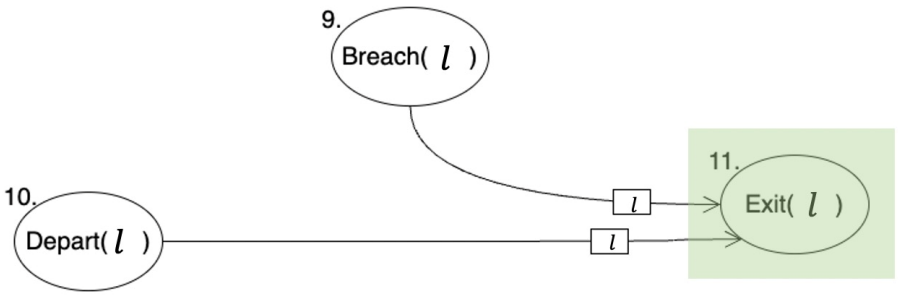


Figure 16 - EG for Exit

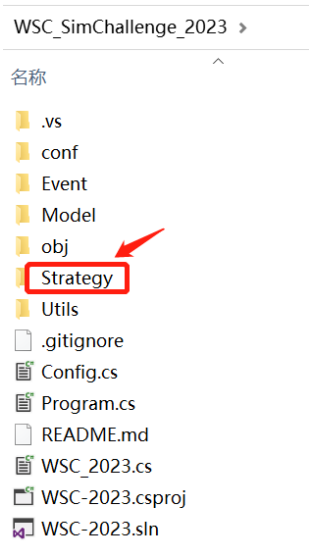
The parameter Lot ***l*** exits the system.

Chapter 4

Evaluation

4.1 File Submission Format

1) Zip package of **Strategy** folder.



Attention: Any modification should be put under the Strategy folder, including newly created files for algorithms purposes and explanatory documents if necessary. Changes in other folders will not be considered.

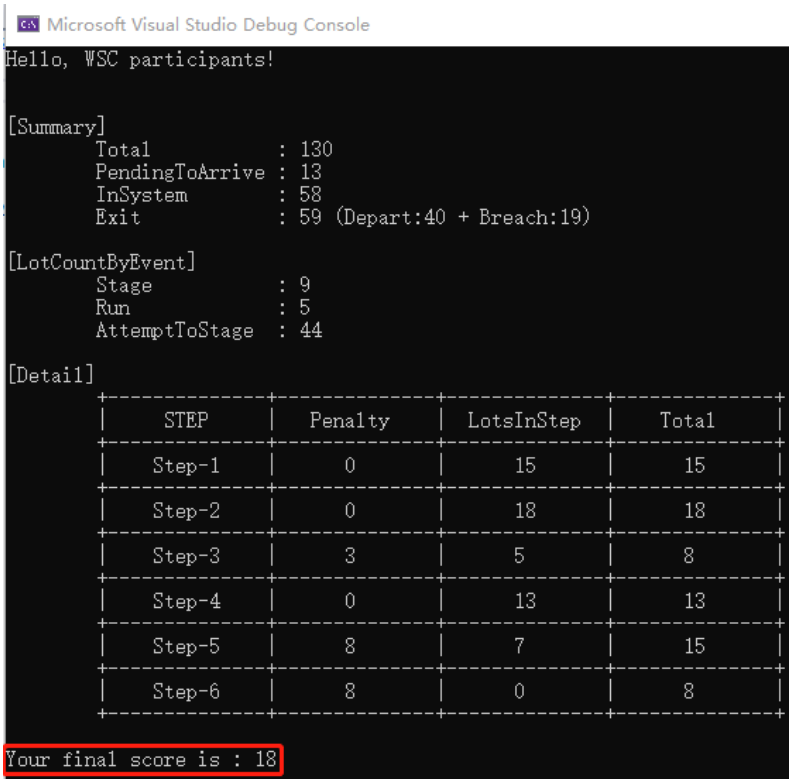
2) Save your code and name it in the following format: **Team Name_Round Number** (e.g. **SealTeam_Round1.zip**).

4.2 File Submission Method

Email address for submission: wsc2023SimChallenge@gmail.com

4.3 Evaluation Criteria

1) Competitors' strategy will be adopted to simulate the given scenario as well as the hidden scenario for the last round.
2) "**Your final result**" from the output is the only criteria to evaluate system performance.



3) In the evaluation stage, multiple random seeds will be used to calculate the average performance for each round. Note that only codes that run successfully will get scores.
4) Weightage and score of each round:

Weightage Table

Round Number	Round 1	Round 2	Round 3	Hidden Round
Weightage	5%	15%	30%	50%

The full score of each round is 100. The score for the top 10 teams will decrease by 5 from 100 according to team's rank (i.e. the number 1 ranking team will be scored as 100, the second ranking team 95, etc.). The rest of the teams after the top 10 teams will receive a score of 50.