# 6SENG006W Concurrent Programming

# FSP Process Composition Analysis & Design Form

| | |
|---|---|
| **Name** | Nethmi Mohotti |
| **Student ID** | 20200486 / w1830188 |
| **Date** | 11/01/2024 |

## 1. FSP Composition Process Attributes

| Attribute | Value |
|---|---|
| **Name** | TICKET_PRINTING_SYSTEM |
| **Description** | This is a composite process model of a printing system. It included two passenger processes, toner technician process and paper technician process. |
| **Sub-processes** (List them.) | TICKET_MACHINE, passenger1:PASSENGER(3), passenger2:PASSENGER(2), TONER_TECHNICIAN , TICKET_TECHNICIAN |
| **Number of States** | 54 |
| **Deadlocks** (yes/no) | no |
| **Deadlock Trace(s) (If applicable)** | N\A |

## 2. FSP "main" Program Code

The code for the parallel composition of all of the sub-processes and the definitions of any constants, ranges & process labelling sets used. (Do not include the code for the individual sub-processes.)

**FSP Program:**

```
const TICKET_COUNT_MAX = 3
range TICKET_RANGE = 0..TICKET_COUNT_MAX

const PAPER_COUNT_MAX = 3
range PAPER_RANGE = 0..PAPER_COUNT_MAX

set MACHINE_ACTIONS = {acquireMachine, printTicket, acquireRefill, release,
            refillPrinter, refillPaper, refillToner,   insufficientPapers}

|| PURCHASE_TICKET_SYSTEM=
    (passenger1 : PASSENGER(3)
    ||passenger2 : PASSENGER(2)
    ||tonerTechnician : TONER_TECHNICIAN
    ||ticketTechnician : TICKET_TECHNICIAN
    ||{passenger1, passenger2, tonerTechnician, ticketTechnician} :: TICKET_MACHINE).
```

## 3. Combined Sub-processes

(Add rows as necessary.)

| Process | Description |
|---|---|
| TICKET_MACHINE | The sub process models the behaviour of the printer. |
| passenger1:PASSENGER | The sub process is to model the behaviour of passenger to print the ticket. This instance of passenger wished to print 3 documents and take mutual exclusive access to the ticket machine to print the tickets. This process terminates after printing all the tickets. |
| passenger2:PASSENGER | The sub process is to model the behaviour of passenger to print the ticket. This instance of passenger wished to print 3 documents and take mutual exclusive access to the ticket machine to print the tickets. This process terminates after printing all the tickets. |
| TONER_TECHNICIAN | The sub process models the behaviour of the toner technician process. This checks if the machine is out of toner and refill the ticket machine with toner. |
| TICKET_TECHNICIAN | This sub process models the behaviour of the ticket technician process which repeatedly check if the machine is out of paper and refill the machine when necessary. |

## 4. Analysis of Combined Process Actions

- **Alphabets** of the combined processes, including the final process labelling.
- **Synchronous** actions are performed by at least two sub-process in the combination.
- **Blocked Synchronous** actions cannot be performed, because at least one of the sub-processes can never preform them, because they were added to their alphabet using alphabet extension.
- **Asynchronous** actions are preformed independently by a single sub-process.

Group actions together if appropriate, e.g. if they include indexes in[0], in[1], …, in[5] as in[1..5]. Add rows as necessary.

| **Processes** | **Alphabet**<br>(Use LTSA's **compressed notation**, if alphabet is large.) |
|---|---|
| TICKET_MACHINE | {passenger1, passenger2, ticketTechnician, tonerTechnician}.{acquireMachine, acquireRefill, insufficientPapers, paperFilled, printTicket, refillPrinter, release} |
| passenger1:PASSENGER(3) | passenger1.{{acquireMachine, acquireRefill, insufficientPapers, printTicket}, printTicket[0], {refillPaper, refillPrinter, refillToner, release}} |
| passenger2:PASSENGER(2) | passenger2.{{acquireMachine, acquireRefill, insufficientPapers, printTicket}, printTicket[0], {refillPaper, refillPrinter, refillToner, release}} |
| tonerTechnician:TONER_TECHNICIAN | tonerTechnician.{acquireMachine, acquireRefill, insufficientPapers, printTicket, refillPaper, refillPrinter, refillToner, release, wait} |
| ticketTechnician:TICKET_TECHNICIAN | ticketTechnician.{acquireMachine, acquireRefill, insufficientPapers, printTicket, refillPaper, refillPrinter, refillToner, release, wait} |

| **Synchronous Actions** | **Synchronised by Sub-Processes (List)** |
|---|---|
| passenger2.acquireMachine, passenger2.printTicket | passenger2:PASSENGER(2) , TICKET_MACHINE |
| passenger1.acquireMachine, passenger1.printTicket | passenger1:PASSENGER(3) , TICKET_MACHINE |
| passenger2.release, ticketTechnician.release | ticketTechnician:TICKET_TECHNICIAN , passenger2:PASSENGER(2), TICKET_MACHINE |
| passenger2.release, tonerTechnician.release | tonerTechnician:TONER_TECHNICIAN, passenger2:PASSENGER(2), TICKET_MACHINE |

| | |
|---|---|
| passenger1.release, ticketTechnician.release | ticketTechnician:TICKET_TECHNICIAN , passenger1:PASSENGER(3), TICKET_MACHINE |
| passenger1.release, tonerTechnician.release | tonerTechnician:TONER_TECHNICIAN, passenger1:PASSENGER(3), TICKET_MACHINE |
| ticketTechnician.acquireRefill, ticketTechnician.refillPaper | ticketTechnician:TICKET_TECHNICIAN, TICKET_MACHINE |
| tonerTechnician.acquireRefill, tonerTechnician.refillToner | tonerTechnician:TONER_TECHNICIAN , TICKET_MACHINE |

| Blocked Synchronous Actions | Blocking Processes | Blocked Processes |
|---|---|---|
| passenger2. acquireRefill, passenger2.printTicket | passenger2:PASSENGER(2) , TICKET_MACHINE | passenger2:PASSENGER(2) |
| passenger3. acquireRefill, passenger3.printTicket | passenger1:PASSENGER(3) , TICKET_MACHINE | passenger1:PASSENGER(3) |
| tonerTechnician.acquireMachine, tonerTechnician.printTicket | ticketTechnician:TICKET_TECHNICIAN, TICKET_MACHINE | ticketTechnician:TICKET_TECH |
| ticketTechnician.acquireMachine, ticketTechnician.printTicket | tonerTechnician:TONER_TECHNICIAN , TICKET_MACHINE | tonerTechnician:TONER_TECH |

| Sub-Processes | Asynchronous Actions (List) |
|---|---|
| passenger2:PASSENGER(2) | N/A |
| passenger1:PASSENGER(3) | N/A |
| TICKET_MACHINE | N/A |
| ticketTechnician:TICKET_TECHNICIAN | {ticketTechnician, tonerTechnician}.wait |
| tonerTechnician:TONER_TECHNICIAN | { ticketTechnician, tonerTechnician}.wait |

## 5. Parallel Composition Structure Diagram

The structure diagram for the parallel composition.