Tracker Project

Juan Loja (lojag95@cse.yorku.ca)
Sadman Sakib Hasan (cse23152@cse.yorku.ca)

December 5, 2017

Statistics.

- Under which account submitted Prism: cse23152
- Under which account submitted Moodle: Hasan, Sadman Sakib
- Implementation Language: Eiffel
- Estimate hours on requirements document: 20 hours
- Estimate of hours on coding and acceptance testing: 32 hours
- Hours on debugging and testing: 20 hours out of 32 hours
- What was the hardest part of the project?: Developing the the TLA+ specefication

Revisions.

Date	Revision	Description
14 November 2017	1.0	Add initial Use Case Diagram and Textual Description
15 November 2017	1.1	Add Abstract Grammar, Acceptance Test and Safety Invariant
28 November 2017	2.0	Add E/R Descriptions
29 November 2017	2.1	Add additional Use Case Textual Descriptions
2 December 2017	3.0	Add complete TLA+ Specification

Requirements Document:
Tracker System

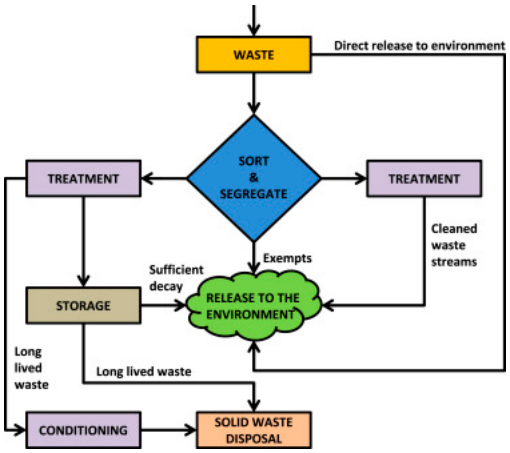
Contents

List of Figures

List of Tables

1. Informal description of nuclear waste tracker

A tracker system monitors the position of waste products in nuclear plants and ensures their safe handling. Our customer requires a software system that operators use to manage safe tracking of radioactive waste in their various nuclear plants. We have so far elicited the following information from our customer.



Containers of material pass through various stages of processing in the tracking part of the nuclear plant. The tracking plant consist of a number of phases usually corresponding to the physical processes that handle the radioactive materials. Not all plants have precisely the same phases.

As an example, containers (containing a a possibly radioactive material type) might arrive at an initial unpacking phase where they are stored for further processing depending on their material contents. All nuclear plants have only the following types of material: *glass*, *metal*, *plastic*, or *liquid*. No other materials are tracked.

A subsequent phase might be called the "assay" phase to measure the recoverable material content of each container before passing onto the next phase. A next stage

might be a compacting phase. A compacting phase might involve dissolving metal contents or crushing glass. Not all material types can necessarily be handled in a phase. For example, we should not move containers with liquid into a compacting phase. Finally the products of the process might be placed in storage. There may be other phases in a particular instance of the tracker.

Each container has a unique identifier and contains only one type of material. It is labelled with a preliminary radiation count (in mSv). When a container is registered in the system, it is also placed in a phase (not necessarily an initial phase).

The sievert (symbol: Sv) is a unit of ionizing radiation dose in the International System of Units (SI) and is a measure of the health effect of radiation on the human body. Quantities that are measured in sieverts are intended to represent the stochastic health risk, which for radiation dose assessment is defined as the probability of cancer induction and genetic damage. One sievert carries with it a 5.5% chance of eventually developing cancer.[1]

For a given plant, there is an initial setup of two important fixed global parameters for a given plant: there is a limit on the maximum radiation in any phase of the plant (in units of mSv), and there is also a limit on the maximum radiation that any container in the plant may have (in mSv). An error status message shall be signalled if there is an attempt to register a new container in the system with radiation that exceeds the container limit.

Another operation is to add a new phase (this is information provided by the Domain experts). Requirements elicitation so far yields that a new phase is specified by a phase ID, a name (e.g. "compacting"), a limit on the maximum number of containers in the phase, and a list of material types that may be treated in the phase. A phase may also be removed if there are no containers anywhere in the system. Also, it is possible for an operator to move a container from one phase to another.

Obviously when dealing with dangerous materials is very important to ensure that no material goes missing and that care is taken to avoid too much material getting into a phase, in case there is a buildup of dangerous substances in one area. The tracking manager is responsible for giving permission to movements of containers between processing phases in order to avoid dangerous situations.

[1] `https://en.wikipedia.org/wiki/Sievert`

2. High level goals

The high level goals for the System Under Description are:

G1: The operator should be able to safely manage tracking of radioactive waste in their nuclear plants.

G2: The cost of manufacturing the tracking system should be as low as possible.

3. Use Case Diagram and Textual Use Cases

The next two pages include the use case textual representation of a particular use case 1 and the use case diagram 1 for the System Under Description.

An additional use case (UC-2) is included in the appendix C. The normal flow of the use case in 1 *includes* calling UC-2.

Use Case ID: UC-1
Use Case Name: Move a container
Primary Actor: Operator
Secondary Actor: Tracking Manager
Description: The Operator specifies the option to move a container from a desired phase to a destination phase by entering the container id and the source and destination phase IDs. The system processes the input and asks the Tracking Manager to verify the movement.
Scenario: Sunny Day
Trigger: Operator indicates to move a container.
Priority: High
Precondition: PRE-1. Container *c* exists in the tracker. PRE-2. Container *c* is in phase *p1*. PRE-3. Container *c* is not in phase *p2*. PRE-4. The material type of *c* is in the list of acceptable materials of *p2*.
Postcondition: POST-1. Container *c* is in phase *p2*. POST-2. Container *c* is not in phase *p1*.
Normal Flow: 1.0 Move a container, *c*, from phase *p1* to a phase *p2* 1. Operator selects the option to move a container from the current phase to a new phase. 2. The System prompts the Operator to enter the ID of the container to be moved, the phase ID of the source and destination phase. 3. Operator enters the ID of the container to be moved and the source and destination phase IDs. 4. System processes the input, finds the destination phase and confirms the number of containers in the destination phase has not reached its limit. 5. System sends a request to the Tracking Manager for the verification of the container movement. 6. System triggers use case UC-2. (Inclusion Point) 7. System responses OK and provides the status of the container to the Operator and Tracking Manager.
Exception Flow: 1.0.E1 Destination phase ID is not in the tracker 1. System displays error **e9: phase identifier not in the system**. 2. System asks the Operator if they want to try another phase (3a) or to exit (4a). 3a. Operator requests to try another destination phase. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 1: UC-1 Textual Description

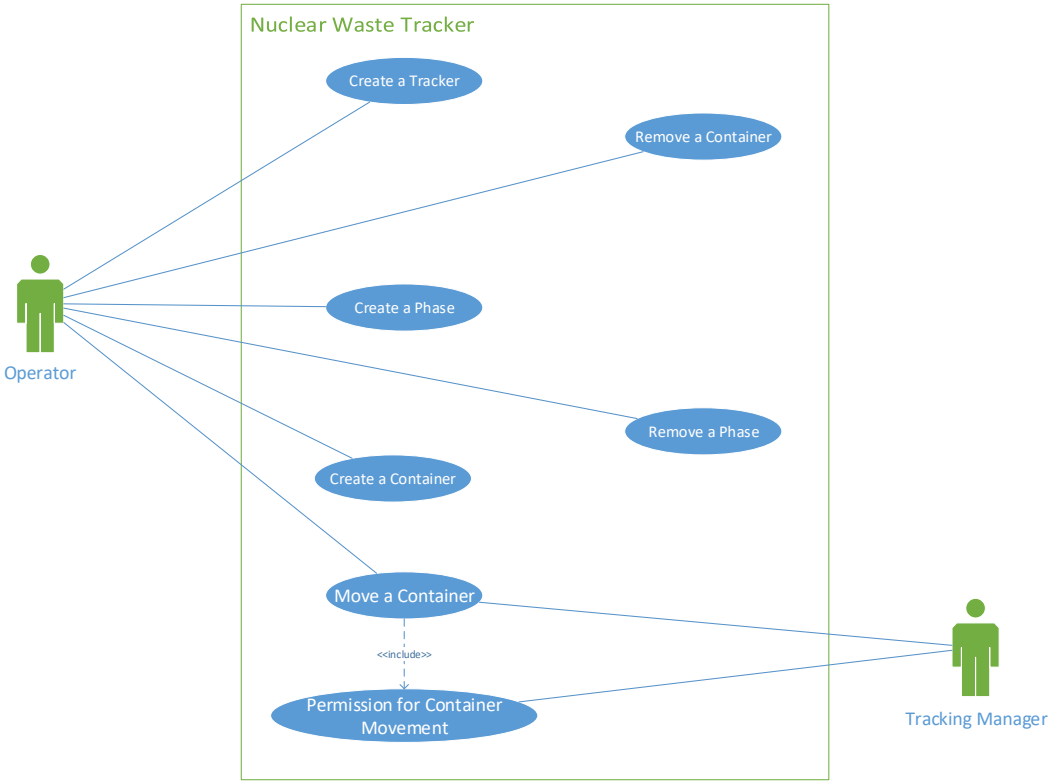


Figure 1: Use Case Diagram

4. E-Descriptions

Below are *three* E-Descriptions for the System Under Description.

ENV1	All nuclear plants must only allow the following types of material to be tracked: *glass*, *metal*, *plastic*, or *liquid*.	

Rationale: The tracking system will only track materials of type: *glass*, *metal*, *plastic*, or *liquid*. Any other materials are not to be tracked.

ENV2	The tracking manager must be responsible for permitting to move a container from one phase to another.	

Rationale: When dealing with dangerous materials it is important to ensure that no material goes missing and that care is taken to avoid too much material getting into a phase, in case there is a buildup of dangerous substances in one area.

ENV3	Materials of a container will be placed in storage once it has passed through various stages of processing.	

Rationale: The material inside a container might be placed in a storage before sending it out for disposal. This is to be completed once the container has passed through various stages of processing.

5. R-Descriptions

Below are *three* R-Descriptions for the System Under Description. Additional R-Descriptions are specified in the appendix B.

REQ1	A given plant shall have two important fixed global parameters: *maximum phase radiation* and *maximum container radiation*.	

Rationale: Each plant must have a fixed maximum phase and container radiation value fixed. The value should be in units of mSv. The sievert (symbol: Sv) is a unit of ionizing radiation dose in the International System of Units (SI) and is a measure of the health effect of radiation on the human body. Quantities that are measured in sieverts are intended to represent the stochastic health risk, which for radiation dose assessment is defined as the probability of cancer induction and genetic damage.

REQ2	Each phase in a plant shall have an unique identifier, a name, limit on the maximum number of containers and list of material types to be treated in that phase.	

Rationale: Each phase in a plant must be associated with an unique identifier. Along with an unique identifier, it should also have a name, maximum number of containers allowed in the phase and the list of material types that can be treated in the phase.

REQ3	A container shall be placed in a phase when registered in a plant.	

Rationale: When a container is first registered in a plant, it is placed in a phase of the operators wish. The phase does not necessarily have to be the initial phase.

6. Abstract UI Grammar for Monitored Events

After several rounds of requirements elicitation, the following user inputs are defined.

```
system tracker -- tracking system for a nuclear plant
type CID = STRING -- container identifiers
type PID = STRING -- phase identifiers
type MATERIAL = {glass, metal, plastic, liquid}
type CONTAINER = TUPLE[material: MATERIAL; radioactivity: VALUE]

new_tracker( -- create a new tracking system
    max_phase_radiation: VALUE
     -- max radiation allowed for all containers in a phase
  ; max_container_radiation: VALUE
     -- max radation allowed in a container
  )

new_phase( -- create a new phase
    pid: PID -- phase identifier
  ; phase_name: STRING
  ; capacity: INT -- number of containers phase handels
  ; expected_materials: ARRAY[MATERIAL] -- subset of materials
)

-- remove a phase with phase identifier pid
remove_phase(pid: PID)

new_container ( -- create a new container and move it to a phase
    cid: CID -- container identifier
  ; c: CONTAINER
  ; pid:PID
  )

remove_container (cid: CID) -- remove a container with container
   identifier cid

move_container ( -- move a container from source phase to target
   phase
    cid: CID -- container identifier to be moved
  ; pid1: PID -- source phase
  ; pid2: PID -- target phase
  )
```

7. Output and Abstract Variables

Below is the TLA+ analysis of the system. The informal table provides (a) the type of each variable and (b) a description explaining the role of each variable in the abstract state for the tracker, and its initial value in the TLA+ model

Abstract Variable	**Description**	**Initial value**
$e \in ERROR$	The error message to be displayed	"ok"
$pid \subseteq PID$	The set of phase identifiers in the tracker	{}
$cid \subseteq CID$	The set of container identifiers in the tracker	{}
$phases \in [pid- > PHASE]$	Function from pid to a record of phase	<>
$containers \in [cid- > CONTAINER]$	Function from cid to a record of container	<>
$mpr \in Int$	The maximum phase radiation in the tracker	0
$mcr \in Int$	The maximum container radiation in the tracker	0

Table 2: Description of TLA+ Output and Abstract Variables

8. ASCII encoded output of the abstract state

The delivered program shall execute from a CentOS7 console as follows:

```
>./tracker.exe -b tests/at0.txt
```

A sequence of user commands, e.g. at0.txt below, shall conform to the abstract UI grammar.

```
-- Acceptance test for Tracker: at0.txt
-- ASCII comments may occur at the beginning or in the middle of a line.
-- Very Basic Use Case to:
--      set tracker radiation parameters
--      create some phases
--      add some containers
--      move some containers

-- Phases are sorted by pid
-- Containers are sorted by cid
-- Sorting follows normal lexicographic string rules,
-- so that pid11 is before pid4; use pid04 if you want pid4 earlier.

-- create a new tracker
-- max. radiation in a phase is 50.0
-- max. radiation in a container is 10.0
new_tracker(50.0, 10.0)

-- add 2 phases each having a container handling capacity of 2:
new_phase("pid2", "compacting", 2, <<glass, metal, plastic>>)
new_phase("pid1", "unpacking", 2, <<glass, metal, plastic, liquid>>)

-- type CONTAINER = TUPLE[material: MATERIAL; radioactivity: VALUE]
-- add some containers
new_container("cid4", [metal,   3.0], "pid1")
new_container("cid1", [glass,   5.5], "pid1")
new_container("cid2", [liquid,  0.5], "pid1") -- e11

-- move some containers
move_container ("cid1", "pid1", "pid2")
move_container ("cid4", "pid1", "pid2")
```

The output shall display as follows:

```
  state 0 ok
  max_phase_radiation: 0.00, max_container_radiation: 0.00
  phases: pid->name:capacity,count,radiation
  containers: cid->pid->material,radioactivity
```

```
->new_tracker(50,10)
  state 1 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
  containers: cid->pid->material,radioactivity
->new_phase("pid2","compacting",2,<<glass, metal, plastic>>)
  state 2 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
    pid2->compacting:2,0,0.00,{glass,metal,plastic}
  containers: cid->pid->material,radioactivity
->new_phase("pid1","unpacking",2,<<glass, metal, plastic, liquid>>)
  state 3 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
    pid1->unpacking:2,0,0.00,{glass,metal,plastic,liquid}
    pid2->compacting:2,0,0.00,{glass,metal,plastic}
  containers: cid->pid->material,radioactivity
->new_container("cid4",[metal, 3],"pid1")
  state 4 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
    pid1->unpacking:2,1,3.00,{glass,metal,plastic,liquid}
    pid2->compacting:2,0,0.00,{glass,metal,plastic}
  containers: cid->pid->material,radioactivity
    cid4->pid1->metal,3.00
->new_container("cid1",[glass, 5.5],"pid1")
  state 5 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
    pid1->unpacking:2,2,8.50,{glass,metal,plastic,liquid}
    pid2->compacting:2,0,0.00,{glass,metal,plastic}
  containers: cid->pid->material,radioactivity
    cid1->pid1->glass,5.50
    cid4->pid1->metal,3.00
->new_container("cid2",[liquid, 0.5],"pid1")
  state 6 e11: this container will exceed phase capacity
->move_container("cid1","pid1","pid2")
  state 7 ok
  max_phase_radiation: 50.00, max_container_radiation: 10.00
  phases: pid->name:capacity,count,radiation
    pid1->unpacking:2,1,3.00,{glass,metal,plastic,liquid}
    pid2->compacting:2,1,5.50,{glass,metal,plastic}
  containers: cid->pid->material,radioactivity
    cid1->pid2->glass,5.50
    cid4->pid1->metal,3.00
->move_container("cid4","pid1","pid2")
```

```
state 8 ok
max_phase_radiation: 50.00, max_container_radiation: 10.00
phases: pid->name:capacity,count,radiation
  pid1->unpacking:2,0,0.00,{glass,metal,plastic,liquid}
  pid2->compacting:2,2,8.50,{glass,metal,plastic}
containers: cid->pid->material,radioactivity
  cid1->pid2->glass,5.50
  cid4->pid2->metal,3.00
```

9. TLA+ specfication, invariants and validation

9.1. Abstract State

The following are the abstract state variables used in the TLA+ Specification. (For the full TLA+ specification, see Appendix A):

First is defining *MATERIAL*, *ERROR*, *PHASE*, *CONTAINER*, and *VALUE*:

- **MATERIAL**: Set of all possible materials each container can have
- **ERROR**: Set of errors for the tracker
- **PHASE**: Set of records containing phase capacity, expected materials, current number of containers in phase, and current radioactivity of phase.
- **CONTAINER**: Set of records that has container radioactivity, container material, and the id of the phase that the container is in.
- **VALUE**: Allows to abstract real numbers to integers. Radioactivity values are represented as integers.

$MATERIAL \triangleq \{\text{“glass”}, \text{“metal”}, \text{“plastic”}, \text{“liquid”}\}$ the only materials each container is expected to have
$ERROR \triangleq \{\text{“ok”}, \text{“E1”}, \text{“E2”}, \text{“E3”}, \text{“E4”}, \text{“E5”}, \text{“E6”}\}$ one error for each action
set of all phase records
$PHASE \triangleq [capacity : Int, curr\_cont : Int, curr\_rad : Int, exp\_mats : \text{SUBSET } MATERIAL]$
$CONTAINER \triangleq [radioactivity : Int, curr\_phase : PID, mat : MATERIAL]$ set of all container records
$VALUE \triangleq 0 \,..\, N$ used for defining numbers for *radioactivity*. Abstracting away from decimals

Figure 2: Defining Tracker Types

The following are the variables that define the state of the tracker as well as their type:

- **cid**: Set of all container ids in the tracker
- **pid**: Set of all phase ids in the tracker
- **containers**: Function from cid to a record of container
- **phases**: Function from pid to a record of phase
- **mpr**: Maximum phase radiation in the tracker
- **mcr**: Maximum container radiation in the tracker
- **e**: Error message to be displayed

VARIABLES
cid
, $containers$
, pid
, $phases$
, mpr
, mcr
, e

$$
\begin{aligned}
&TypeOK \triangleq \\
&\quad \land \quad e \in ERROR \\
&\quad \land \quad pid \subseteq PID \\
&\quad \land \quad cid \subseteq CID \\
&\quad \land \quad phases \in [pid -> PHASE] \\
&\quad \land \quad containers \in [cid -> CONTAINER \\
&\quad \land \quad mpr \in Int \\
&\quad \land \quad mcr \in Int
\end{aligned}
$$

9.2. Abstractions Used

The main abstraction used in modeling the tracker in TLA+ was using integers instead of real numbers to set the radioactivity value for phases and containers. This was done to make it a bit simple when defining the container, phases, and the tracker itself. Another thing to note is that phase name was not included in the model. This was simply to reduce the amount of elements in the record representing the phase and make the model checker run faster. In addition to this, in the implementation of the tracker, phases and containers were stored in hash maps. In the TLA+ specification, there was a set representing all the current phase ids in the tracker as well as a set of functions with mapping $[pid \rightarrow CONTAINER]$. Containers were represented in a similar way.

9.3. Safety Invariants

The following are the safety invariants described in the TLA+ Specification:

- **SafeRadioactivity**: Checks if the radioactivity is within the limits of the tracker's maximum radiation level for all phases and containers
- **PhasesNotOverCapacity**: Checks if all phases are within their capacity in terms of the maximum containers they can hold
- **AllContainersInPhase**: Checks if all containers are in a phase that exists in a tracker

$$
\begin{array}{l}
SafeRadioactivity \triangleq \\
\quad \land \quad \forall\, id \in pid : \land \quad 0 <= phases[id].curr\_rad <= mpr \\
\quad \land \quad \forall\, id \in pid : \land \quad 0 <= containers[id].radioactivity <= mcr
\end{array}
$$

$$
\begin{array}{l}
PhasesNotOverCapacity \triangleq \\
\quad \forall\, id \in pid : 0 <= phases[id].curr\_cont <= phases[id].capacity
\end{array}
$$

$$
\begin{array}{l}
AllContainersInPhase \triangleq \\
\quad \forall\, id \in cid : containers[id].curr\_phasae \in pid
\end{array}
$$

9.4. Actions

The following are the actions for the tracker as specified in TLA:

This is the *new_tracker* action used to set the maixmum phase and container radioactivity. The gurad checks that the values are non-negative and also makes sure the maximum container radioactivity is not greater than the maximum phase radioactivity. In addition, it checks that there are no containers currently in the tracker. If these conditions are met, the values *mcr* and *mpr* are set. The rest of the variables remain unchanged.

$$
\begin{array}{l}
new\_tracker(p,\ c) \triangleq \\
\quad \land\ p \geq 0 \quad \text{max phase radiation is non-negative} \\
\quad \land\ c \geq 0 \quad \text{max container radiation is non-negative} \\
\quad \land\ p \geq c \quad \text{max container radiation cannot be greater than max phase radiation} \\
\quad \land\ cid = \{\} \quad \text{there should be no containers in tracker} \\
\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
\quad \land\ cid' = cid \\
\quad \land\ containers' = containers \\
\quad \land\ pid' = pid \\
\quad \land\ phases' = phases \\
\quad \land\ mpr' = p \quad \text{set the values for maximum phase and container radiation} \\
\quad \land\ mcr' = c \\
\quad \land\ e' = \text{“ok”} \\
\quad \land\ Print(\langle \text{“new\_tracker”},\ p,\ c \rangle,\ \text{TRUE})
\end{array}
$$

Figure 3: New Tracker Action in TLA

This is the *new_phase* action used to create a new phase in the tracker. The gurad checks that the phase does not exist in the tracker already. It checks that it has a capacity of at least 1. And it also checks that there are no containers currently in the tracker. If these conditions are met, then a new phase id is added to the set of tracker phase ids as well as a new PHASE record. The rest of the variables remain unchanged.

$$
\begin{array}{l}
new\_phase(p,\ cap,\ mats) \triangleq \\
\quad \land\ p \notin pid \quad \text{phase } id \text{ must not already exist} \\
\quad \land\ cap > 0 \quad \text{capacity must be positive} \\
\quad \land\ mats \subseteq MATERIAL \quad \text{check that expected materials is subset of all possible } MATERIALs \\
\quad \land\ cid = \{\} \quad \text{there should be no containers in tracker} \\
\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
\quad \land\ cid' = cid \\
\quad \land\ containers' = containers \\
\quad \land\ pid' = pid \cup \{p\} \quad \text{add new } id \text{ to set of phase ids and extend phase function with a new record} \\
\quad \land\ phases' = phases \ @@\ p :> [capacity \mapsto cap,\ curr\_cont \mapsto 0,\ curr\_rad \mapsto 0,\ exp\_mats \mapsto mats] \\
\quad \land\ mpr' = mpr \\
\quad \land\ mcr' = mcr \\
\quad \land\ e' = \text{“ok”} \\
\quad \land\ Print(\langle \text{“new\_phase”},\ p,\ cap \rangle,\ \text{TRUE})
\end{array}
$$

Figure 4: New Phase Action in TLA

This is the *new_container* action used to create a new container in the tracker. The gurad checks for several things. It checks that this container has not been added before by checking its id. It checks that the destination phase is already in the tracker. It checks that the container doesn't go over radioactivity limit. And finally, it checks if the container can go into the destination phase by checking the capacity, current phase radioactivity, and whether or not the phase accepts the container *MATERIAL*. If these conditions are met, then a new container is added by adding its id to the set of container ids in the tracker. A new *CONTAINER* record is added and finally, the destination phase is updated accordingly. The rest of the variables remain unchanged.

$$
\begin{aligned}
&new\_container(c,\ rad,\ p,\ m) \triangleq \\
&\quad \land\ c \notin cid \quad \text{container } id \text{ must not already exist} \\
&\quad \land\ p \in pid \quad \text{make sure the phase exists} \\
&\quad \land\ rad \geq 0 \land rad \leq mcr \quad \text{radiation must be within limits of tracker} \\
&\quad \land\ phases[p].curr\_rad + rad \leq mpr \quad \text{make sure the phase radiation won't exceed} \\
&\quad \land\ phases[p].curr\_cont + 1 \leq phases[p].capacity \quad \text{and capacity won't exceed} \\
&\quad \land\ m \in MATERIAL \land m \in phases[p].exp\_mats \quad \text{check phase accepts this material} \\
&\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
&\quad \land\ cid' = cid \cup \{c\} \quad \text{add new } id \text{ to set of ids and extend container function with new record} \\
&\quad \land\ containers' = containers\ @@\ c :> [radioactivity \mapsto rad,\ curr\_phase \mapsto p,\ mat \mapsto m] \\
&\quad \land\ pid' = pid \quad \text{update the phase by adding new container to it. That is update the radiation and count} \\
&\quad \land\ phases' = [phases\ \text{EXCEPT}\ ![p].curr\_rad = @ + rad,\ ![p].curr\_cont = @ + 1] \\
&\quad \land\ mpr' = mpr \\
&\quad \land\ mcr' = mcr \\
&\quad \land\ e' = \text{“ok”} \\
&\quad \land\ Print(\langle \text{“new\_container”},\ c,\ rad,\ p \rangle,\ \text{TRUE})
\end{aligned}
$$

Figure 5: New Container Action in TLA

This is the *remove_container* action used to remove an existing container in the tracker. The guard simply checks if the container exists in the tracker. If it does, it removes the container id from the set of container ids in the tracker. It also removes the CONTAINER record. In addition, it updates the phase where the container was currently by removing it from it. The rest of the variables remain unchanged.

$$
\begin{array}{l}
remove\_container(c) \triangleq \\
\quad \land\ c \in cid \ \text{this container must exist} \\
\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
\quad \land\ pid' = pid \ \text{update phase by removing container from it} \\
\quad \land\ phases' = [phases \ \text{EXCEPT} \\
\qquad\qquad ![containers[c].curr\_phase].curr\_rad = @ - containers[c].radioactivity, \\
\qquad\qquad ![containers[c].curr\_phase].curr\_cont = @ - 1] \\
\quad \land\ cid' = cid \setminus \{c\} \ \text{remove } id \text{ from set of container ids and also remove the } id \text{ from the} \\
\quad \land\ containers = RanSub(containers, \{c\}) \ \text{domain and the record from the range} \\
\quad \land\ containers' = DomSub(\{c\}, containers) \ \text{of containers function} \\
\quad \land\ mpr' = mpr \\
\quad \land\ mcr' = mcr \\
\quad \land\ e' = \text{“ok”} \land Print(\langle \text{“remove\_container”}, c \rangle, \text{TRUE})
\end{array}
$$

Figure 6: Remove Container Action in TLA

This is the *remove_phase* action used to remove an existing phase in the tracker. The guard simply checks if the phase exists in the tracker and if there are no containers currently in the tracker. If these conditions are met, the phase id is removed from the set of phase ids in the tracker and the PHASE record is removed as well. The rest of the variables remain unchanged.

$$
\begin{array}{l}
remove\_phase(p) \triangleq \\
\quad \land\ p \in pid \quad \text{this phase must exist} \\
\quad \land\ cid = \{\} \quad \text{cannot remove phase when there are containers currently in tracker} \\
\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
\quad \land\ cid' = cid \\
\quad \land\ containers' = containers \\
\quad \land\ pid' = pid \setminus \{p\} \quad \text{remove } id \text{ from set of phase ids} \\
\quad \land\ phases\ = RanSub(phases, \{p\}) \quad \text{remove } id \text{ from domain and record from range} \\
\quad \land\ phases' = DomSub(\{p\}, phases) \quad \text{of phases function} \\
\quad \land\ mpr' = mpr \\
\quad \land\ mcr' = mcr \\
\quad \land\ e' = \text{“ok”} \\
\quad \land\ Print(\langle \text{“remove\_phase”}, p \rangle, \text{TRUE})
\end{array}
$$

Figure 7: Remove Phase Action in TLA

This is the *move_container* action used to move a container from one phase to another. The guard checks for several things. Firstly it checks if the container, source and target phase exist. Then it checks if the target phase will be able to accept the container. If the conditions are met, the container is removed from the source phase and added to the destination phase. The rest of the variables remain unchanged.

$$
\begin{array}{l}
move\_container(c,\ p1,\ p2) \triangleq \\
\quad \land\ c \in cid \ \text{check container exists} \\
\quad \land\ p1 \in pid \land p2 \in pid \ \text{check phases exist} \\
\quad \land\ p1 \neq p2 \ \text{source and destination phases must be different} \\
\quad \land\ containers[c].curr\_phase = p1 \ \text{container should be in source phase} \\
\quad \land\ phases[p2].curr\_rad + containers[c].radioactivity \leq mpr \ \text{safe radiation} \\
\quad \land\ phases[p2].curr\_cont + 1 \leq phases[p2].capacity \ \text{and capacity on destination should not exceed} \\
\quad \land\ containers[c].mat \in phases[p2].exp\_mats \ \text{check destination phase accepts container material} \\
\quad \text{\_\_\_\_}guard \text{ ends } here\text{\_\_\_\_} \\
\quad \land\ cid' = cid \ \text{update record of container with the new phase it is in} \\
\quad \land\ containers' = [containers \ \text{EXCEPT} \ ![c].curr\_phase = p2] \\
\quad \land\ pid' = pid \ \text{update phase records by removing container from } p1 \text{ and adding it to } p2 \\
\quad \land\ phases' = [phases \ \text{EXCEPT} \\
\qquad\qquad ![p1].curr\_rad = @ - containers[c].radioactivity, \\
\qquad\qquad ![p1].curr\_cont = @ - 1, \\
\qquad\qquad ![p2].curr\_rad = @ + containers[c].radioactivity, \\
\qquad\qquad ![p2].curr\_cont = @ + 1] \\
\quad \land\ mpr' = mpr \\
\quad \land\ mcr' = mcr \\
\quad \land\ e' = \text{“ok”} \\
\quad \land\ Print(\langle \text{“move\_container”},\ c,\ p1,\ p2 \rangle, \ \text{TRUE})
\end{array}
$$

Figure 8: Move Container Action in TLA

9.5. TLC Model Checker

The initial predicate and next-state relation is below:

```
******** Defining INIT and NEXT ********

Init ≜ ∧ e = "ok"   initialize error ok
       ∧ mpr = 0    max phase radiation is 0
       ∧ mcr = 0    max container radiation is 0
       ∧ cid = {}   empty set of container ids
       ∧ containers = ⟨⟩   make empty set of function from cid → CONTAINER
       ∧ pid = {}   make emtpy set
       ∧ phases = ⟨⟩   make empty set of function from pid → PHASE

Next ≜ ∨ (∃ p, c ∈ VALUE : new_tracker(p, c))
       ∨ (∃ p ∈ PID, cap ∈ VALUE, mats ∈ SUBSET MATERIAL : new_phase(p, cap, mats))
       ∨ (∃ c ∈ CID, rad ∈ VALUE, p ∈ PID, mat ∈ MATERIAL : new_container(c, rad, p, mat))
       ∨ (∃ c ∈ CID : remove_container(c))
       ∨ (∃ p ∈ PID : remove_phase(p))
       ∨ (∃ c ∈ CID, p1, p2 ∈ PID : move_container(c, p1, p2))
```

Figure 9: Initial predicate and Next-State Relation

When running the model checker, it was set to check for Deadlock, and to check the invariants for the system. The following configuration was used:

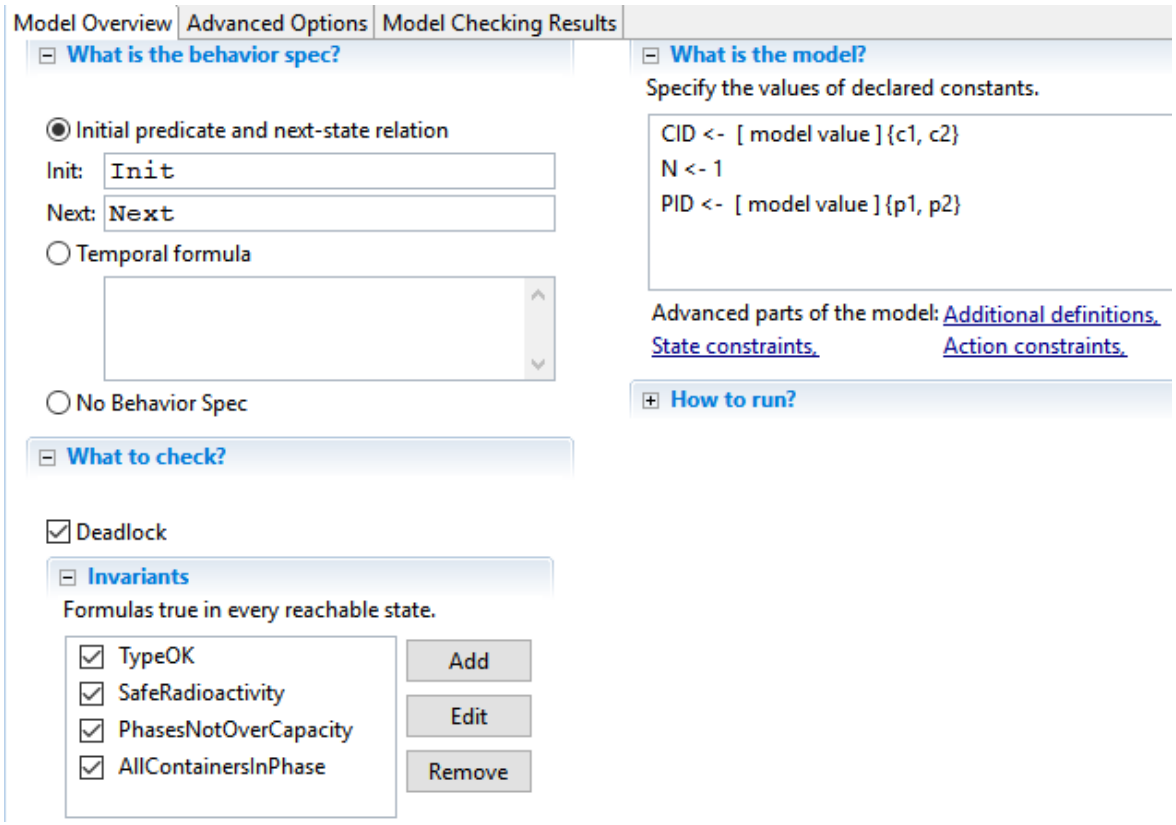


Figure 10: TLA Model Checker Configuration

Increasing N to more than 1 or adding more values to the mode for *CID* or *PID* (i.e. c3, c4, ..) makes the test run for more than one hour.

With the configuration above, the model checker takes about 1 minute to run on a system with specs: Intel Core i7-7820HQ 2.9 GHz with 16 GB RAM. The output is seen below.

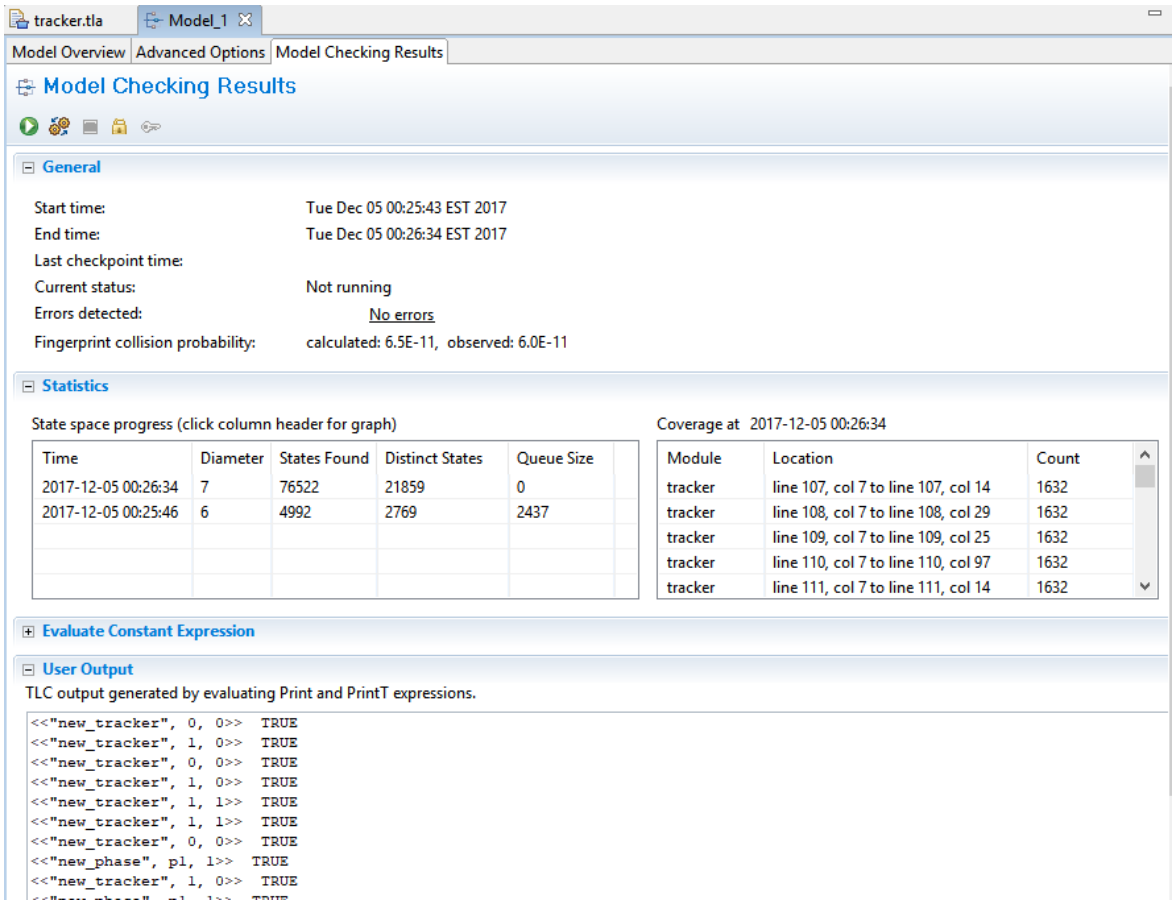


Figure 11: TLA Model Checker Output

10. Completeness, disjointness and well-definednes of the specification

			pid	cid	phases	containers	mpr	mcr	err
$i = 0$			{}	{}	<>	<>	0	0	"ok"
$i > 0$	$new\_tracker(p, c)(i)$	$G1$	NC				**p**	**c**	NC
		$\neg G1$	NC						E1
	$new\_phase(p, cap, mats)(i)$	$G2$	C1	NC	C2	NC			
		$\neg G2$	NC						E2
	$new\_container(c, rad, p, m)$	$G3$	NC	C3	C4	C5	NC		
		$\neg G3$	NC						E3
	$remove\_container(c)$	$G4$	NC	C6	C7	C8	NC		
		$\neg G4$	NC						E4
	$remove\_phase(p)$	$G5$	C9	NC	C10	NC			
		$\neg G5$	NC						E5
	$move\_container(c, p1, p2)$	$G6$	NC		C11	C12	NC		
		$\neg G6$	NC						E6

Table 3: Function Table for the actions in the TLA+ specification

Guards	**Value**
G1	$p >= 0 \land c >= 0 \land p >= c \land cid = \{\}$
G2	$p \notin pid \land cap > 0 \land mats \subseteq MATERIAL \land cid = \{\}$
G3	$c \notin cid \land p \notin pid \land rad >= 0 \land rad <= mcr \land$ $phases[p].curr\_rad + rad <= mpr \land phases[p].curr\_cont + 1 <=$ $phases[p].capacity \land m \in MATERIAL \land m \in phases[p].exp\_mats$
G4	$c \in cid$
G5	$p \in pid \land cid = \{\}$
G6	$c \in cid \land p1 \in pid \land p2 \in pid \land p1 \neq p2 \land$ $phases[p2].curr\_rad + containers[c].radioactivity <= mpr \land$ $phases[p2].curr\_cont + 1 <= phases[p2].capacity \land$ $containers[c].mat \in phases[p2].exp\_mats$

Table 4: Guards described in the TLA+ specification

Conditions	**Value**
C1	$pid \cup \{p\}$
C2	$phases@@p :> [capacity \mid -> cap, curr\_cont \mid -> 0, curr\_rad \mid -> 0, exp\_mats \mid -> mats]$
C3	$cid \cup \{c\}$
C4	$[phases \text{ EXCEPT } ![p].curr\_rad = @ + rad, ![p].curr\_rad = @ + 1]$
C5	$containers@@c :> [radioactivity \mid -> rad, curr\_phase \mid -> p, mat \mid -> m$
C6	$cid \setminus \{c\}$
C7	$[phases \text{ EXCEPT } ![containers[c].curr\_phase].curr\_rad = @ - containers[c].radioactivity, ![containers[c].curr\_phase].curr\_cont = @ - 1]$
C8	$DomSub(\{c\}, containers)$
C9	$pid \setminus \{p\}$
C10	$DomSub(\{p\}, phases)$
C11	$[phases \text{ EXCEPT } ![p1].curr\_rad = @ - containers[c].radioactivity, ![p1].curr\_cont = @ - 1, ![p2].curr\_rad = @ - containers[c].radioactivity, ![p2].curr\_cont = @ - 1]$
C12	$[containers \text{ EXCEPT } ![c].curr\_phase = p2]$

Table 5: Transformation functions for each variable described in the TLA+ specification

The error messages used in the TLA+ specification relates to the error messages specified by the customer during the elicitation phase refer to Table 7. Below is the table of relation between the error messages described in the TLA+ specification and the customer elicited error messages.

Error Messages	**Value**
E1	$E1 \in \{e1, e2, e3, e4\}$
E2	$E2 \in \{e1, e5, e6, e7, e8\}$
E3	$E3 \in \{e5, e9, e10, e11, e12, e13, e14, e18\}$
E4	$E4 \in \{e15\}$
E5	$E5 \in \{e1, e9\}$
E6	$E6 \in \{e9, e11, e12, e13, e15, e16, e17$

Table 6: Errors described in the TLA+ specification

11. Analysis of required status and error messages

Table below shows the value for each error message:

	Message
e1	current tracker is in use
e2	max phase radiation must be non-negative value
e3	max container radiation must be non-negative value
e4	max container must not be more than max phase radiation
e5	identifiers/names must start with A-Z, a-z or 0..9
e6	phase identifier already exists
e5	identifiers/names must start with A-Z, a-z or 0..9
e7	phase capacity must be a positive integer
e8	there must be at least one expected material for this
e9	phase identifier not in the system
e10	this container identifier already in tracker
e11	this container will exceed phase capacity
e12	this container will exceed phase safe radiation
e13	phase does not expect this container material
e14	container radiation capacity exceeded
e15	this container identifier not in tracker
e16	source and target phase identifier must be different
e17	this container identifier is not in the source phase
e18	this container radiation must not be negative
ELSE	ok

Table 7: Description table for error messages

12. Acceptance Tests

The table below describes the relationship between the use cases and concrete acceptance tests developed for the system:

Use Cases	Description	Acceptance Tests
UC-1	Moving a container	at1.txt
		at3.txt
UC-2	Permission for moving a container	at1.txt
		at3.txt
UC-3	Creating a new tracker	at1.txt
		at2.txt
		at3.txt
		at4.txt
		at5.txt
		at6.txt
UC-4	Creating a new phase	at4.txt
		at5.txt
		at6.txt
UC-5	Creating a new container	at4.txt
		at5.txt
		at6.txt
UC-6	Removing a container	at2.txt
		at4.txt
		at5.txt
UC-7	Removing a phase	at2.txt
		at4.txt
		at5.txt

Table 8: Acceptance Tests for each Use Case

13. Traceability Matrix

The table below traces which R-descriptions are tested by which acceptance tests:

REQ	a1.txt	a2.txt	a3.txt	a4.txt	a5.txt	a6.txt
REQ1	X	X	X	X	X	X
REQ2				X	X	X
REQ3				X	X	X
REQ4				X	X	X
REQ5		X	X	X		

Table 9: Traceability Matrix for R-descriptions

14. Requirements Elicitation

Our abstract UI grammar in Phase1 was similar to the abstract UI grammar provided by our customer in Phase2. However there were some major discrepancies between now and our initial elicitation of the customer needs:

- We assumed that when creating a new container, the radioactivity value of the container could be negative.
- We assumed that when moving a container, it was not necessary to know the source phase.
- Our initial abstract grammar used a tuple to represent Containers with all the values needed to create a container, whereas the customer's representation included only the material and radiation value.
- Our initial abstract grammar used a tuple to represent Phases, whereas the customer's representation did not have any tuple for Phases.

15. Appendices

A. TLA

The following pages include the TLA+ specification for the System Under Description:

MODULE *tracker*

Packages

EXTENDS *Integers*, *TLC*

Constants

CONSTANTS

N,
CID, set of all possible container identifiers
PID set of all possible phase identifiers

ASSUME

$N > 0$

$MATERIAL \triangleq \{\text{“glass”}, \text{“metal”}, \text{“plastic”}, \text{“liquid”}\}$ the only materials each container is expected to have
$ERROR \triangleq \{\text{“ok”}, \text{“E1”}, \text{“E2”}, \text{“E3”}, \text{“E4”}, \text{“E5”}, \text{“E6”}\}$ one error for each action

Note:
*E*1: maps to errors from *new_tracker* action (*e*1, *e*2, *e*3, *e*4)
*E*2: maps to errors from *new_phase* action (*e*1, *e*5, *e*6, *e*7, *e*8)
*E*3: maps to errors from *new_container* action (*e*5, *e*9, *e*10, *e*11, *e*12, *e*13, *e*14, *e*18)
*E*4: maps to errors from *remove_container* action (*e*15)
*E*5: maps to errors from *remove_phase* action (*e*1, *e*9)
*E*6: maps to errors from *move_contaier* action (*e*9, *e*11, *e*12, *e*13, *e*15, *e*16, *e*17)

set of all phase records
$PHASE \triangleq [capacity : Int, curr\_cont : Int, curr\_rad : Int, exp\_mats : \text{SUBSET } MATERIAL]$
$CONTAINER \triangleq [radioactivity : Int, curr\_phase : PID, mat : MATERIAL]$ set of all container records
$VALUE \triangleq 0 \,..\, N$ used for defining numbers for *radioactivity*. Abstracting away from decimals

Variables

VARIABLES

cid, set of container ids in the tracker
containers, set of functions mapping from *CID* to *CONTAINER*
pid, set of phase ids in the tracker
phases, set of functions mapping from *PID* to *PHASE*
mpr, the maximum radiation allowed in a phase
mcr, the maximum radiation allowed in a container
e error message holder

Helper Functions used for remvoving element from domain and range from function
Subtraction on domain of f for set S: This was obtained from Course Forum User *amin*9 $**$
$DomSub(S, f) \triangleq [x \in \text{DOMAIN } f \setminus S \mapsto f[x]]$
Subtraction on range of f for set S: This was obtained from Course Forum User *amin*9 $**$
$RanSub(f, S) \triangleq [x \in \{y \in \text{DOMAIN } f : f[y] \notin S\} \mapsto f[x]]$

**** INVARIANTS *****

type invariant

$$\begin{aligned} \mathit{TypeOK} \triangleq\ & \land e \in \mathit{ERROR} \\ & \land \mathit{pid} \subseteq \mathit{PID} \\ & \land \mathit{cid} \subseteq \mathit{CID} \\ & \land \mathit{phases} \in [\mathit{pid} \to \mathit{PHASE}] \\ & \land \mathit{containers} \in [\mathit{cid} \to \mathit{CONTAINER}] \\ & \land \mathit{mpr} \in \mathit{Int} \\ & \land \mathit{mcr} \in \mathit{Int} \end{aligned}$$

checking *radioactivity* is within limits of tracker

$$\begin{aligned} \mathit{SafeRadioactivity} \triangleq\ & \land \forall\, \mathit{id} \in \mathit{pid} : \land 0 \leq \mathit{phases}[\mathit{id}].\mathit{curr\_rad} \\ & \qquad\qquad\qquad\quad \land \mathit{phases}[\mathit{id}].\mathit{curr\_rad} \leq \mathit{mpr} \\ & \land \forall\, \mathit{id} \in \mathit{cid} : \land 0 \leq \mathit{containers}[\mathit{id}].\mathit{radioactivity} \\ & \qquad\qquad\qquad\quad \land \mathit{containers}[\mathit{id}].\mathit{radioactivity} \leq \mathit{mcr} \end{aligned}$$

checking that all phases are within capacity in terms of the number containers in them

$$\begin{aligned} \mathit{PhasesNotOverCapacity} \triangleq\ & \forall\, \mathit{id} \in \mathit{pid} : \land 0 \leq \mathit{phases}[\mathit{id}].\mathit{curr\_cont} \\ & \qquad\qquad\qquad\qquad\quad \land \mathit{phases}[\mathit{id}].\mathit{curr\_cont} \leq \mathit{phases}[\mathit{id}].\mathit{capacity} \end{aligned}$$

All containers are in a phase that exists in the tracker

$$\mathit{AllContainersInPhase} \triangleq \forall\, \mathit{id} \in \mathit{cid} : \mathit{containers}[\mathit{id}].\mathit{curr\_phase} \in \mathit{pid}$$

**** ACTIONS *****

$\mathit{new\_tracker}(p,\ c) \triangleq$

- $\land\ p \geq 0$ max phase radiation is non-negative
- $\land\ c \geq 0$ max container radiation is non-negative
- $\land\ p \geq c$ max container radiation cannot be greater than max phase radiation
- $\land\ \mathit{cid} = \{\}$ there should be no containers in tracker

____*guard* ends *here*____

- $\land\ \mathit{cid}' = \mathit{cid}$
- $\land\ \mathit{containers}' = \mathit{containers}$
- $\land\ \mathit{pid}' = \mathit{pid}$
- $\land\ \mathit{phases}' = \mathit{phases}$
- $\land\ \mathit{mpr}' = p$ set the values for maximum phase and container radiation
- $\land\ \mathit{mcr}' = c$
- $\land\ e' =$ "ok"
- $\land\ \mathit{Print}(\langle$"new_tracker", $p,\ c\rangle,$ TRUE$)$

$new\_phase(p,\ cap,\ mats) \triangleq$
$\land\ p \notin pid$ phase *id* must not already exist
$\land\ cap > 0$ capacity must be positive
$\land\ mats \subseteq MATERIAL$ check that expected materials is subset of all possible *MATERIALs*
$\land\ cid = \{\}$ there should be no containers in tracker
____guard ends *here____*
$\land\ cid' = cid$
$\land\ containers' = containers$
$\land\ pid' = pid \cup \{p\}$ add new *id* to set of phase ids and extend phase function with a new record
$\land\ phases' = phases\ @@\ p :> [capacity \mapsto cap,\ curr\_cont \mapsto 0,\ curr\_rad \mapsto 0,\ exp\_mats \mapsto mats]$
$\land\ mpr' = mpr$
$\land\ mcr' = mcr$
$\land\ e' =$ "ok"
$\land\ Print(\langle$"new_phase", $p,\ cap\rangle,$ TRUE$)$

$new\_container(c,\ rad,\ p,\ m) \triangleq$
$\land\ c \notin cid$ container *id* must not already exist
$\land\ p \in pid$ make sure the phase exists
$\land\ rad \geq 0 \land rad \leq mcr$ radiation must be within limits of tracker
$\land\ phases[p].curr\_rad + rad \leq mpr$ make sure the phase radiation won't exceed
$\land\ phases[p].curr\_cont + 1 \leq phases[p].capacity$ and capacity won't exceed
$\land\ m \in MATERIAL \land m \in phases[p].exp\_mats$ check phase accepts this material
____guard ends *here____*
$\land\ cid' = cid \cup \{c\}$ add new *id* to set of ids and extend container function with new record
$\land\ containers' = containers\ @@\ c :> [radioactivity \mapsto rad,\ curr\_phase \mapsto p,\ mat \mapsto m]$
$\land\ pid' = pid$ update the phase by adding new container to it. That is update the radiation and count
$\land\ phases' = [phases$ EXCEPT $![p].curr\_rad = @ + rad,\ ![p].curr\_cont = @ + 1]$
$\land\ mpr' = mpr$
$\land\ mcr' = mcr$
$\land\ e' =$ "ok"
$\land\ Print(\langle$"new_container", $c,\ rad,\ p\rangle,$ TRUE$)$

$remove\_container(c) \triangleq$
$\land\ c \in cid$ this container must exist
____guard ends *here____*
$\land\ pid' = pid$ update phase by removing container from it
$\land\ phases' = [phases$ EXCEPT
$![containers[c].curr\_phase].curr\_rad = @ - containers[c].radioactivity,$
$![containers[c].curr\_phase].curr\_cont = @ - 1]$
$\land\ cid' = cid \setminus \{c\}$ remove *id* from set of container ids and also remove the *id* from the
$\land\ containers\ = RanSub(containers,\ \{c\})$ domain and the record from the range
$\land\ containers' = DomSub(\{c\},\ containers)$ of containers function
$\land\ mpr' = mpr$
$\land\ mcr' = mcr$
$\land\ e' =$ "ok" $\land\ Print(\langle$"remove_container", $c\rangle,$ TRUE$)$

$remove\_phase(p) \triangleq$
$\land\ p \in pid$ this phase must exist
$\land\ cid = \{\}$ cannot remove phase when there are containers currently in tracker
____*guard* ends *here*____
$\land\ cid' = cid$
$\land\ containers' = containers$
$\land\ pid' = pid \setminus \{p\}$ remove *id* from set of phase ids
$\land\ phases\ = RanSub(phases, \{p\})$ remove *id* from domain and record from range
$\land\ phases' = DomSub(\{p\}, phases)$ of phases function
$\land\ mpr' = mpr$
$\land\ mcr' = mcr$
$\land\ e' =$ "ok"
$\land\ Print(\langle$"remove_phase", $p\rangle$, TRUE)

$move\_container(c, p1, p2) \triangleq$
$\land\ c \in cid$ check container exists
$\land\ p1 \in pid \land p2 \in pid$ check phases exist
$\land\ p1 \neq p2$ source and destination phases must be different
$\land\ containers[c].curr\_phase = p1$ container should be in source phase
$\land\ phases[p2].curr\_rad + containers[c].radioactivity \leq mpr$ safe radiation
$\land\ phases[p2].curr\_cont + 1 \leq phases[p2].capacity$ and capacity on destination should not exceed
$\land\ containers[c].mat \in phases[p2].exp\_mats$ check destination phase accepts container material
____*guard* ends *here*____
$\land\ cid' = cid$ update record of container with the new phase it is in
$\land\ containers' = [containers$ EXCEPT $![c].curr\_phase = p2]$
$\land\ pid' = pid$ update phase records by removing container from $p1$ and adding it to $p2$
$\land\ phases' = [phases$ EXCEPT
$![p1].curr\_rad = @ - containers[c].radioactivity,$
$![p1].curr\_cont = @ - 1,$
$![p2].curr\_rad = @ + containers[c].radioactivity,$
$![p2].curr\_cont = @ + 1]$
$\land\ mpr' = mpr$
$\land\ mcr' = mcr$
$\land\ e' =$ "ok"
$\land\ Print(\langle$"move_container", $c, p1, p2\rangle$, TRUE)

******** Defining *INIT* and NEXT ********

$$
\begin{aligned}
\mathit{Init} \triangleq\ & \land e = \text{“ok”} \quad \text{initialize error } ok \\
& \land mpr = 0 \quad \text{max phase radiation is 0} \\
& \land mcr = 0 \quad \text{max container radiation is 0} \\
& \land cid = \{\} \quad \text{empty set of container ids} \\
& \land containers = \langle\rangle \quad \text{make empty set of function from } cid \to CONTAINER \\
& \land pid = \{\} \quad \text{make emtpy set} \\
& \land phases = \langle\rangle \quad \text{make empty set of function from } pid \to PHASE
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Next} \triangleq\ & \lor (\exists\, p,\, c \in VALUE : new\_tracker(p,\, c)) \\
& \lor (\exists\, p \in PID,\, cap \in VALUE,\, mats \in \text{SUBSET } MATERIAL : new\_phase(p,\, cap,\, mats)) \\
& \lor (\exists\, c \in CID,\, rad \in VALUE,\, p \in PID,\, mat \in MATERIAL : new\_container(c,\, rad,\, p,\, mat)) \\
& \lor (\exists\, c \in CID : remove\_container(c)) \\
& \lor (\exists\, p \in PID : remove\_phase(p)) \\
& \lor (\exists\, c \in CID,\, p1,\, p2 \in PID : move\_container(c,\, p1,\, p2))
\end{aligned}
$$

B. Additional R-Descriptions

Below are additional R-Descriptions for the System Under Description:

REQ4	An error message shall be signalled if a container is registered with a radiation value greater than the plant's maximum radiation limit.	

Rationale: It is dangerous to allow having a container in the system with a radiation value that exceed the plant's maximum radiation limit. Whenever a new container with a radiation value greater than the plant's maximum radiation limit is attempted to be added to the plant, an error shall be signalled to the operator.

REQ5	A phase may also be removed if there are no containers in the plant.	

Rationale: Keeping a phase in a plant may not be necessary when there are no containers in the plant. Hence, the phase can be removed by the operator if there are no more containers left in the plant.

C. Additional Textual Use Cases

Use Case ID: UC-2
Use Case Name: Permission for Container Movement
Primary Actor: Tracking Manager
Description: The Tracking Manager is requested to verify if a containers movement between two phases is safe. The Tracking Manager verifies the destination phases accepted materials consist of the material from the container and enters it. The system processes it and signals success for the movement of the container.
Scenario: Sunny Day
Trigger: Normal Flow 1.0.6
Priority: High
Precondition: PRE-1. Number of containers in destination phase, *p2*, is less than the maximum containers allowed in *p2*.
Postcondition: POST-1. Movement has been verified.
Normal Flow: **2.0 Permission for container, *c*, to move from phase *p1* to a phase *p2*** 1. Tracking Manager is notified upon a verification request comes in from the system. 2. Tracking Manager verifies the destination phases accepted materials consist of the material in the container. 3. The Tracking Manager inputs verified. 4. System processes the input and signals success for the movement of the container.

Table 10: UC-2 Textual Description

Use Case ID: UC-3
Use Case Name: Create a Tracker
Primary Actor: Operator
Description: The Operator specifies the option to create a tracker. The system prompts the operator to input the maximum phase radiation level in each phase in the tracker and the maximum container radiation level in each container in a phase in the tracker. The Operator inputs the desired values and clicks enter. The system processes the input and succeeds on creating the tracker.
Scenario: Sunny Day
Trigger: Operator indicates to create a tracker.
Priority: High
Precondition: PRE-1. *mpr* must be greater than or equal to 0. PRE-2. *mcr* must be greater than or equal to 0. PRE-3. *mpr* must be greater than or equal to *mcr*. PRE-4. There should not be any active containers or phases in the tracker.
Postcondition: POST-1. Tracker, *t*, is created.
Normal Flow: **3.0 Create a tracker, *t* with max phase radiation,*mpr* and max container radiation,*mcr*** 1. Operator selects the option to create a tracker. 2. System prompts the Operator to enter the maximum phase radiation level allowed in each phase and the maximum container radiation allowed in a container. 3. Operator enters the max phase radiation and max container radiation and clicks enter. 4. System processes the input and responses OK.
Exception Flow: 3.0.E1 Max Phase Radiation is negative 1. System displays error **e2: max phase radiation must be non-negative value**. 2. System asks the Operator if they want to input a correct max phase radiation value (3a) or to exit (4a). 3a. Operator requests to enter a new max phase radiation value. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 11: UC-3 Textual Description

Use Case ID: UC-4
Use Case Name: Create a Phase
Primary Actor: Operator
Description: The Operator specifies the option to create a new phase. The system prompts the operator to input the phase ID, phase name, container capacity and a list of expected materials in the phase. The Operator inputs the desired values and clicks enter. The system processes the input and succeeds on creating a new phase.
Scenario: Sunny Day
Trigger: Operator indicates to create a phase.
Priority: High
Precondition: PRE-1. Phase with *pid* does not exist in the tracker. PRE-2. Name of the phase must start with A-Z, a-z or 0..9. PRE-3. Identifier of the phase must start with A-Z, a-z or 0..9. PRE-4. Phase capacity, *pcap*, must be greater than 0. PRE-5. The expected materials list must have at least one material.
Postcondition: POST-1. Phase, *p*, with *pid* is created.
Normal Flow: **3.0 Create a phase, *p*, with *pid*, *pname*, *pcap* and *pmat*** 1. Operator selects the option to create a phase. 2. System prompts the Operator to the phase ID, phase name, container capacity and a list of expected materials in the phase. 3. Operator enters the desired values and clicks enter. 4. System processes the input and responses OK.
Exception Flow: 3.0.E1 Expected materials list is empty 1. System displays error **e8: there must be at least one expected material for this phase**. 2. System asks the Operator if they want to input a new set of expected materials (3a) or to exit (4a). 3a. Operator requests to enter a new set of expected materials. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 12: UC-4 Textual Description

Use Case ID: UC-5
Use Case Name: Create a Container
Primary Actor: Operator
Description: The Operator specifies the option to create a new container. The system prompts the operator to input the container ID, a tuple containing the material in the container and the radioactivity value in the container and the phase ID to be added to. The Operator inputs the desired values and clicks enter. The system processes the input and succeeds on creating a new container.
Scenario: Sunny Day
Trigger: Operator indicates to create a container.
Priority: High
Precondition: PRE-1. Container with *cid* does not exist in the tracker. PRE-2. Phase with *pid* exists in the tracker. PRE-3. Identifier of the container must start with A-Z, a-z or 0..9. PRE-4. The containers radiation must be greater than 0. PRE-5. The container must have a material type.
Postcondition: POST-1. Container, *c*, with *cid* is created and added to phase with *pid*.
Normal Flow: **5.0 Create a container, *c*, with *cid*, *container* and *pid*** 1. Operator selects the option to create a container. 2. System prompts the Operator to the container ID, containers radioactivity value and material type and phase ID it needs to be added. 3. Operator enters the desired values and clicks enter. 4. System processes the input and responses OK.
Exception Flow: 5.0.E1 **Phase to be added to does not expect this container material** 1. System displays error **e13: phase does not expect this container material**. 2. System asks the Operator if they want to a add another container with the correct expected material (3a) or to exit (4a). 3a. Operator requests to create a new container with the correct expected material. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 13: UC-5 Textual Description

Use Case ID: UC-6
Use Case Name: Remove a Container
Primary Actor: Operator
Description: The Operator specifies the option to remove a container. The system prompts the operator to input the container ID. The Operator inputs the ID and clicks enter. The system processes the input and succeeds on removing the container.
Scenario: Sunny Day
Trigger: Operator indicates to remove a container.
Priority: High
Precondition: PRE-1. Container with *cid* exist in the tracker.
Postcondition: POST-1. Container with *cid* has been removed from the tracker.
Normal Flow: 6.0 Remove a container with *cid* 1. Operator selects the option to remove a container. 2. System prompts the Operator to the enter container ID. 3. Operator enters the value and clicks enter. 4. System processes the input and responses OK.
Exception Flow: 6.0.E1 Container to be removed is not in the tracker 1. System displays error **e15: this container identifier not in tracker**. 2. System asks the Operator if they want to a remove another container (3a) or to exit (4a). 3a. Operator requests to remove another container. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 14: UC-6 Textual Description

Use Case ID: UC-7
Use Case Name: Remove a Phase
Primary Actor: Operator
Description: The Operator specifies the option to remove a container. The system prompts the operator to input the phase ID. The Operator inputs the ID and clicks enter. The system processes the input and succeeds on removing the phase.
Scenario: Sunny Day
Trigger: Operator indicates to remove a phase.
Priority: High
Precondition: PRE-1. Phase with *pid* exist in the tracker.
Postcondition: POST-1. Phase with *pid* has been removed from the tracker.
Normal Flow: 7.0 Remove a phase with *pid* 1. Operator selects the option to remove a phase. 2. System prompts the Operator to the enter phase ID. 3. Operator enters the value and clicks enter. 4. System processes the input and responses OK.
Exception Flow: 7.0.E1 Phase ID is not in the tracker 1. System displays error **e9: phase identifier not in the system**. 2. System asks the Operator if they want to a remove another phase (3a) or to exit (4a). 3a. Operator requests to remove another phase. 3b. System starts normal flow over. 4a. Operator asks to exit. 4b. System terminates the use case.

Table 15: UC-7 Textual Description