EECS4312 Project
Nuclear Waste Tracker Project

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

November 15, 2017

Prism account used for submission: cse23152@cse.yorku.ca

Requirements Document: Nuclear Waste Tracker

Revisions

Date	Revision	Description
14 November 2017	1.0	Add Use Case Diagram and Textual Description
15 November 2017	1.1	Add Abstract Grammar, Acceptance Test and Safety Invariant

Contents

List of Figures

List of Tables

1 System Overview

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

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

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.

2 Abstract Grammar

The following is an abstract grammar modeling the tracker and its functionality based on information received from client in Phase 1:

```
% types of materials
type MATERIAL = { liquid, glass, metal, plastic }
% ID for container
type ID_C = STRING
% ID for phase
type ID_P = STRING
% custom type for the radiation sievert unit of ionizing dose
type SIEVERT = VALUE

% modeling the container for tracker
type CONTAINER =  TUPLE
        [
        id: ID_C,
        material: MATERIAL,
        radiation : SIEVERT
        phase_id: ID_P,
        ]

% modeling a phase
type PHASE = TUPLE
        [
        id: ID_P,
        name: STRING,
        max_containers : INT,
        number_of_containers : INT,
        radiation : SIEVERT, % total radiation in phase
        acceptable_materials: LIST[MATERIAL]
        ]

% modeling a plant
type PLANT = TUPLE
        [
        % maximum phase radiation
        mp_radiation : SIEVERT,
        % maximum container radiation
        mc_container_radiation : SIEVERT
```

```
        ]


% status message of the waste tracker
type ERROR = STRING
type STATUS = {ok, ERROR}

% -------------------------  USER ACTIONS -------------------------

% plants actions
create_plant(
                max_phase_radiation: SIEVERT,
                max_container_radiation : SIEVERT )

% phase actions
add_phase(
                id: ID_P,
                name: STRING,
                max_containers : INT,
                acceptable_materials: LIST[MATERIAL])

remove_phase(id: ID_P)

% container actions
add_container(
                id: ID_C,
                material:  MATERIAL,
                radiation : SIEVERT,
                phase_id: ID_P)

move_container(
                container_id: ID_C,
                phase_id: ID_P)

remove_container(id: ID_C)
```

3 Use Cases

3.1 Use Case Textual Description

Below is the textual use case description of a following scenario:

Use Case: Move a container, c, to a new phase, p **Scenario:** Sunny Day **Precondition:** a) Container c exists in the plant. b) Container c is not in phase p. c) The material type of c is in the list of acceptable materials of p. **Postcondition:** a) Container c has been moved to phase p	**Actors:** Customer, Tracking Manager
Actor	**System Responses**
1. Customer chooses the option to move a container.	1. The System prompts the Customer to enter the ID of the container to be moved and the phase ID to be moved to.
2. Customer enters the ID of the container to be moved and the ID of the phase to be moved to.	2. System sends the input to the Tracking Manager for verification.
3. Tracking Manager finds the following movement of container between phases is safe and approves the movement.	3. System responses OK and provides the status of the container to the Customer and Tracking Manager.

Table 1: Use Case Textual Representation

3.2 Use Case Diagram

Below is the use case diagram of the scenario described in the last page:

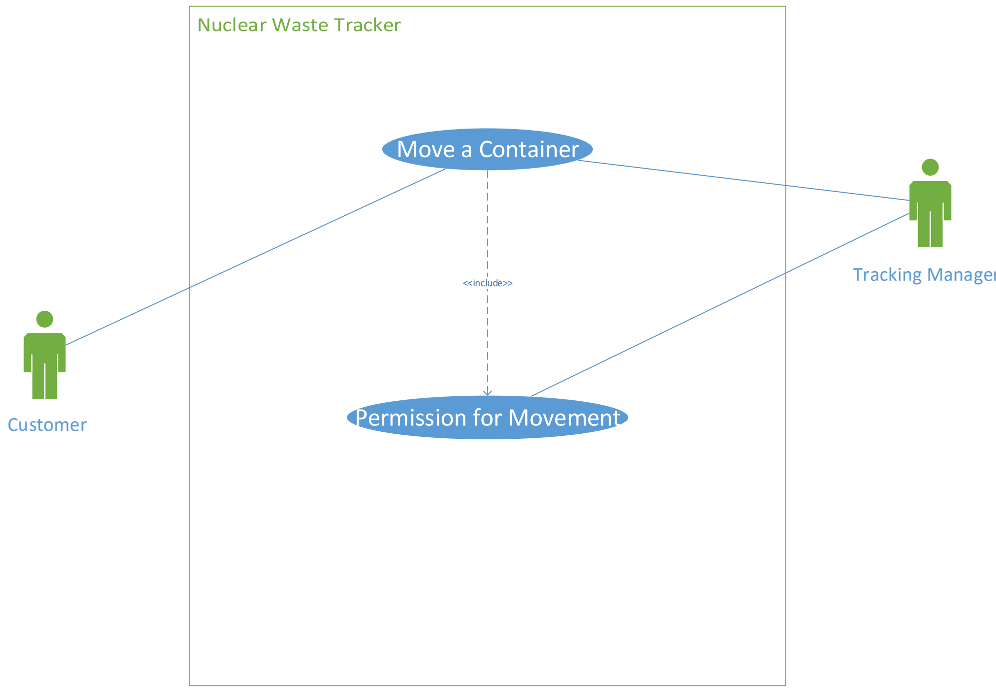

Figure 1: Use Case Diagram

4 Acceptance Test

The following shows some sample user input and the output state of the tracker in an abstract manner:

```
->create_plant( 5.0, 3.0)
status: ok
phases: {}
containers: {}
->add_phase( "ph_unpack", "unpack", 2, [metal, plastic, glass])
status: ok
phases: {ph_unpack}
containers: {}
->add_phase( "ph_store", "compact", 3, [metal, glass, liquid])
status: ok
phases: {ph_unpack, ph_store}
containers: {}
->add_container("cn_1", metal, 2.0, "ph_unpack")
status: ok
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack]}
->add_container("cn_2", liquid, 2.0, "ph_unpack")
status: ERROR: material cannot be treated in this phase
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack]}
->add_container("cn_2", liquid, 2.0, "ph_store")
status: ok
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store]}
->add_container("cn_3", glass, 2.0, "ph_store")
status: ok
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_store]}
->move_container("cn_1", "ph_store")
status: ERROR:  phase radiation exceeds limit
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                     [cn_3 in ph_store]}
->move_container("cn_3", "ph_unpack")
status: ok
```

```
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_unpack]}
->add_phase( "ph_clean", "compact", 3, [metal, glass, plastic])
status: ok
phases: {ph_unpack, ph_store, ph_clean}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_unpack]}
->remove_phase( "ph_clean")
status: ok
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_unpack]}
->add_container("cn_4", metal, 1.0, "ph_unpack")
status: ERROR: no more containers allowed in phase
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_unpack]}
->add_container("cn_4", metal, 1.0, "ph_store")
status: ok
phases: {ph_unpack, ph_store}
containers: {[cn_1 in ph_unpack], [cn_2 in ph_store],
                  [cn_3 in ph_unpack], [cn_4 in ph_store]}
```

5 Safety Invariant

One important invariant is to make sure that all phases and containers have a radiation that is less than the maximum amount of radiation allowed in a container and phase as specified in the plant (*mc_radiation* and *mp_radiation*). This is a way to ensure that care is taken to prevent too much build up of dangerous materials in a phase.

$$\boxed{\begin{array}{l} SafeRadiationAmount : \forall c \in CONTAINERS, p \in PHASES : \\ c.radiation <= mc\_radiation \wedge p.radiation <= mp\_radiation \end{array}}$$