# “RM-ODP part 2: Foundations” in Alloy

Andrey Naumenko, Alain Wegmann

Institute for computer Communication and Applications,
Swiss Federal Institute of Technology – Lausanne.
EPFL-DSC-ICA, CH-1015 Lausanne, Switzerland
`{Andrey.Naumenko, Alain.Wegmann}@epfl.ch`

This paper presents the Alloy ([3], [4], [5]) code for the formalization of “RM-ODP part 2: Foundations” [1]. The formalization is defined and explained in the separate work [6], [7].

```
    /* ------------------------------------------------ beginning of code ------------------------------------------------*/

model RM-ODP {
domain {ODP_Concepts}
state {

/* declaration of ODP concept categories (RM-ODP 2.5) */

  partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts : static ODP_Concepts

/* declaration of "Basic interpretation concepts" (RM-ODP 2.6) */

            partition UniverseOfDiscourse, InterpretationPossibilities  : static  BasicInterpretationConcepts
            partition Entity, Proposition: UniverseOfDiscourse
            partition FirstOrderProposition, HigherOrderProposition: Proposition
            holds : Proposition -> UniverseOfDiscourse+
            System : Entity
            Sybsystem : System

/* introduction of relations between RM-ODP concept categories */

            modeledByBMC : FirstOrderProposition -> BasicModellingConcepts
            modeledBySC : HigherOrderProposition -> SpecificationConcepts
            mappedToBMC : SpecificationConcepts -> BasicModellingConcepts
            mappedToSC : BasicModellingConcepts -> SpecificationConcepts

/* declaration of "Basic modelling concepts" (RM-ODP 2.8) */

  partition Constitution,  SpaceTime, Information : static BasicModellingConcepts
  partition Object, Environment : static Constitution
  environment (~object) : Object! -> Environment!
  partition StructuralInfo, BehavioralInfo : static Information
  Behavior : BehavioralInfo
  State_ : StructuralInfo
  partition Action, BehavioralConstraint: static Behavior
  corresponding_constraint (~constrained_action) : Action -> BehavioralConstraint
  partition InternalAction, Interaction : static Action
  partition InteractionPoint, Space, Time : static SpaceTime
  Interface: Behavior
  Activity: Behavior
  LocationInSpace : Space
  space_within_interval : LocationInSpace -> Space+
  state_location(~corresponding_state) : State_! -> Space! /* introduced to link Information and Space for the defini-
tion of LocationInSpace */
  LocationInTime : Time
  time_within_interval : LocationInTime -> Time+
  interface_at_interaction_point: InteractionPoint -> Interface
  space_location: InteractionPoint -> LocationInSpace!
  time_location: InteractionPoint -> LocationInTime!
  constitution_state: Constitution! -> State_
  object_state: Object! -> State_
  environment_state: Environment! -> State_
  potential_activity: State_ -> Activity+
  object_behavior: Object! -> Behavior!
  environment_behavior: Environment! -> Behavior!
  instant: Time -> Time!
  state_existence: Time! -> State_!
  constraining : Environment! -> BehavioralConstraint
  participant : Action -> Constitution
  participating_object : Action -> Object!
  instant_begin : Action -> Time!
  instant_end : Action -> Time!
  X : BasicModellingConcepts

        /* declaration of "Specification concepts" (RM-ODP 2.9) */

  partition Type, Class, Instance, Composition, Decomposition: SpecificationConcepts
  TemplateType: Type
  Template, InstantiationRules : TemplateType
  Instantiation: Instance
  partition Creation, Introduction : Instantiation
  associated_type: Class! -> Type!
  member_of_class(~set_of): Instance+ -> Class!
  satisfies_type(~valid_for): Instance+ -> Type!
  TemplateClass: Class
  associated_template_type: TemplateClass! -> TemplateType!
  member_of_template_class(~set_of_instantiations): Instantiation+ -> TemplateClass!
  subtype(~supertype): Type -> Type
  subclass(~superclass): Class -> Class
  specification (~instantiation): Instantiation -> Template!
  derived_class(~base_class): TemplateClass -> TemplateClass
  incremental_modification: Template -> Template
  refinement: SpecificationConcepts -> SpecificationConcepts

        /* declaration of "Specific Specification concepts" (RM-ODP 2.9) */

  Role, Invariant, Precondition, Postcondition : Type
  composite_object: Decomposition -> Object!
  interface_signature: Template -> Interface!

}

        /* invariant for "Basic interpretation concepts" (RM-ODP 2.6) */

inv AssertOrDeny {
        all a: UniverseOfDiscourse, b: Proposition | (a in b.holds) || (a not in b.holds)
        }

        /* invariant for "Basic modelling concepts" (RM-ODP 2.8) */

inv TimeDependance{
        all o: Object, t: Time  |  one t.instant ->one o.object_state
}

        /* definitions for "Basic interpretation concepts" (RM-ODP 2.6) */

def FirstOrderProposition {
        all p: FirstOrderProposition | (p.holds: Entity)
}
def HigherOrderProposition {
        all p: HigherOrderProposition | (p.holds: Proposition)
}

        /* definitions for relations between RM-ODP concept categories  */
```

```
def mappedToBMC {
            all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop: FirstOrderProposition, hop: HigherOrder-
Proposition | bmc in sc.mappedToBMC <-> (fop.holds=hop.holds.holds) && (fop.modeledByBMC = bmc) &&
(hop.modeledBySC = sc)
            }
def mappedToSC {
            all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop: FirstOrderProposition, hop: HigherOrder-
Proposition | sc in bmc.mappedToSC <-> (fop.holds=hop.holds.holds) && (fop.modeledByBMC = bmc) &&
(hop.modeledBySC = sc)
            }

            /* definitions for "Basic modelling concepts" (RM-ODP 2.8) */

def participant {
            all a: Action, b: Constitution | b in a.participant <-> (a.instant_begin.state_existence in b.constitution_state)
&& (a.instant_end.state_existence in b.constitution_state) /*pre & post state are in the allowed states of the content;
introduced to characterize those entity models, which participate in an Action - to define Action associated with an
object, Internal Action and Interaction */
}
def Action{
            all a: Action |  (a.instant_begin != a.instant_end) && (a.instant_begin.state_existence !=
a.instant_end.state_existence) && (a.participating_object in a.participant) /* the last condition is here to show that
there exist at least one object associated with an action*/
}
def InternalAction {
            all a: InternalAction | a.participating_object in a.participant -> a.participating_object.environment not in
a.participant
}
def Interaction {
            all a: Interaction | a.participating_object in a.participant -> a.participating_object.environment in a.participant
}
def Behavior {
            all b: Behavior |  ((b in Action) && ( some b.corresponding_constraint) && ( b.corresponding_constraint in
Behavior)) || ((b in BehavioralConstraint) && ( some b.constrained_action) && ( b.constrained_action in Behavior))
}
def Interface {
            all i: Interface | ((i in Interaction) && ( some i.corresponding_constraint) && ( i.corresponding_constraint in
Interface)) || ((i in BehavioralConstraint) && ( some i.constrained_action) && ( i.constrained_action in Interface))
}
def LocationInSpace {
            all ls: LocationInSpace | some a | (a.instant_begin.state_existence.state_location in
ls.space_within_interval) && (a.instant_end.state_existence.state_location in ls.space_within_interval)
}
def LocationInTime {
            all lt: LocationInTime | some a | (a.instant_begin in lt.time_within_interval) && (a.instant_end in
lt.time_within_interval)
}
def interface_at_interaction_point {
            all ip: InteractionPoint, i: Interface | (i in ip.interface_at_interaction_point) <->
((i.instant_begin.state_existence.state_location in ip.space_location.space_within_interval) &&
(i.instant_end.state_existence.state_location in ip.space_location.space_within_interval) && (i.instant_begin in
ip.time_location.time_within_interval) && (i.instant_end in ip.time_location.time_within_interval))
}
def InteractionPoint {
            all ip: InteractionPoint | one ls: LocationInSpace | one lt: LocationInTime | ip.space_location = ls &&
ip.time_location = lt && some ip.interface_at_interaction_point
}
def X {
            all a: X | a.mappedToSC.mappedToBMC = a
}

            /* definitions for "Specification concepts" (RM-ODP 2.9) */

def Class {
```

```
            all c: Class | some i | one t | i.satisfies_type = t && i in c.set_of && i.member_of_class = c &&
c.associated_type = t
            }
def Instance {
            all a: Instance | some t | a.satisfies_type = t
            }
def subtype {
            all t1: Type, t2: Type | t1 in t2.subtype <-> ( t1.valid_for.satisfies_type=t2)
}
def supertype{
            all t1: Type, t2: Type | t2 in t1.supertype <-> ( t1.valid_for.satisfies_type=t2)
}
def subclass {
            all c1: Class, c2: Class | c1 in c2.subclass <-> ( c1.associated_type in c2.associated_type.subtype)
}
def superclass {
            all c1: Class, c2: Class | c2 in c1.superclass <-> ( c1.associated_type in c2.associated_type.subtype)
}
def Template {
            all tpl: Template |  tpl.instantiation.specification = tpl
            //recurrence is introduced in RM-ODP: 2-9.11 references 2-9.13 and vice versa
            }
def associated_template_type {
            all c: Class, t: Type, tc: TemplateClass, tt: TemplateType | ((c.associated_type = t) &&
(tc.associated_template_type = tt)) <-> ((tt = t) && (tc = c))
}
def member_of_template_class {
            all ii: Instantiation, tc: TemplateClass, i: Instance, c: Class | ((i.member_of_class = c) &&
(ii.member_of_template_class = tc)) <-> ((i = ii) && (tc = c))
}
def derived_class {
            all tc1: TemplateClass, tc2: TemplateClass | tc1.derived_class = tc2 <->
tc1.set_of_instantiations.specification.incremental_modification.instantiation.member_of_template_class = tc2
}
def Composition {
            all c: Composition | one a1:X | some a2:X | one a3:X  | a1+a2=a3 <-> (c = a1.mappedToSC) && (c =
a2.mappedToSC)
}
def Decomposition {
            all d: Decomposition | one a1:X | some a2:X | one a3:X  | a1+a2=a3 <-> d = a3.mappedToSC
}
def refinement {
            all spec1: SpecificationConcepts, spec2: SpecificationConcepts | some d: SpecificationConcepts |
(spec1.refinement = spec2) -> (spec1+d=spec2)
}
def Type {
            all t: Type | some x:X | x.mappedToSC = t
}

            /* definitions for "Specific Specification concepts" (RM-ODP 2.9) */

def composite_object {
            all d: Decomposition | one o: Object | (d.composite_object = o) <-> ((d.mappedToBMC = o) &&
(o.mappedToSC = d))
}
def interface_signature{
            all t: Template, i: Interface, a: Interaction | (t.interface_signature = i) <-> a in i && t in a.mappedToSC && a
in t.mappedToBMC
}
def Role{
            all id: Role | one b:Behavior | b.mappedToSC = id
}
def Invariant{
            all i: Invariant | some o: Object | o.mappedToSC = i
}
```

```
def Precondition{
        all prec: Precondition | some a : Action | one s : State_ | a.mappedToSC = prec &&
a.instant_begin.state_existence = s
}
def Postcondition{
        all postc: Postcondition |some a : Action | one s : State_ | a.mappedToSC = postc &&
a.instant_end.state_existence = s
}

        /* Instantiate and Delition operations (RM-ODP 2.9) */

op Instantiate (i: Instantiation, bmc: BasicModellingConcepts, new_bmc: BasicModellingConcepts', new_i: Instantia-
tion', x:X'!, ix: Instantiation'!) {
        x not in bmc
        ix not in i
        new_bmc = bmc + x
        new_i = i + ix
        x.mappedToSC' = ix
        ix.mappedToBMC' = x
}

op Deletion (i: Instantiation, bmc: BasicModellingConcepts, new_bmc: BasicModellingConcepts', new_i: Instantia-
tion', x:X!, ix: Instantiation!) {
        x in bmc
        ix in i
        x.mappedToSC = ix
        ix.mappedToBMC = x
        new_bmc = bmc - x
        new_i = i - ix
        x not in new_bmc
        ix not in new_i
}

}
  /* ---------------------------------------------- end of code ----------------------------------------------*/
```

## References

[1] ISO, ITU. ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation X.901, X.902, X.903, X.904. “Open Distributed Processing - Reference Model”. 1995-96.

[3] D. Jackson, “A Comparison of Object Modelling Notations: Alloy, UML and Z”. MIT Lab for Computer Science. August 1999. http://sdg.lcs.mit.edu/~dnj/pubs/alloy-comparison.pdf

[4] D. Jackson, “Alloy: A Lightweight Object Modelling Notation”. Technical Report 797, MIT Laboratory for Computer Science, Cambridge, MA, February 2000. http://sdg.lcs.mit.edu/~dnj/pubs/alloy-journal.pdf

[5] D. Jackson, Software Design Group. “The Alloy Constraint Analyzer. Online documentation: Short Guide to Alloy”. http://sdg.lcs.mit.edu/alloy/docs/alloy-guide.html ; MIT Laboratory for Computer Science, Cambridge, MA.

[6] A. Naumenko, A. Wegmann, G. Genilloud, W. F. Frank. "Proposal for a formal foundation of RM-ODP concepts". *Proceedings of ICEIS 2001, Workshop On Open Distributed Processing - WOODPECKER`2001, J. Cordeiro, H. Kilov (Eds.),* Setúbal, Portugal, July 2001.

[7] A. Naumenko, A. Wegmann. “A Viewpoint on Formal Foundation of RM-ODP Conceptual Framework”. *EPFL-DSC Technical report No. DSC/2001/040*, Swiss Federal Institute of Technology, Lausanne, Switzerland, July 2001.