Specification Case Study: Patient Record System

A first step towards developing a generalized (and persistent) "Patient Record System" is to model the known "Patient Population", specified as class $PP[\mathbb{P}, \mathbb{I}]$:

A Patient Population includes an initially-empty set of *patients* (as uniquely identified by elements of a given type $\mathbb{P}$). Each patient has some associated *information* (of type $\mathbb{I}$), and is either *alive* or *dead*.

PP

$$Patient : \wp\, \mathbb{P}$$
$$Info : Patient \to \mathbb{I}$$
$$\{Alive, Dead\} : \mathsf{part}\ Patient$$

$$Patient' = \emptyset$$

1. Two *queries*, leaving its state *unchanged*, are to be provided for the class PP:

Current information i for patient p can be *shown* at any time.

$PP?info(p \to i)$

p: Patient, i : Info p

Whether patient p is or is not alive can be *shown* as a boolean test t.

$PP?isAlive(p \to t)$

P : Patient, t: Boolean

Give a *simple* formal specification for each query, guided by its documentation. (5)

2. All possible *changes-of-state* at this level are specified by the following *events*:

New patient p is created with information i, with p having an initial value of Alive.

$PP!NewPatient(i \to p)$

$$i : \mathbb{I}\ ;\ p : \mathbb{P} \setminus Patient$$

$$Info'\ p = i\ ;\ p \in Alive'$$

Information i on Patient p can only be updated if the value for is different to the current Info for p.

$PP!UpdateInfo(p, i)$

$$p : Patient\ ;\ i : \mathbb{I}$$
$$i \neq Info\ p$$

$$Info'\ p = i$$

Death of patient is registered, in order for the patient to be recorded as dead, must have been previously alive.

$PP!Death(p)$

$$p : Alive$$

$$p \in Dead'$$

Document each event informally, using *simple* but *precise* natural language. (9)

3. At a later development stage, the overall "Health-Care System" might then be introduced by *extending* PP. This is specified as class $HCS[\mathbb{P}, \mathbb{I}, \mathbb{S}, \mathbb{H}]$:

The Health-Care System supports its Patient Population by means of some *hierachical* set of *services* and nested *sub-services* (with unique identifiers from type $\mathbb{S}$). It also maintains a central register of *health-professionals* (with unique identifiers from type $\mathbb{H}$), and their current *associations* with any service or sub-service; those who have at least one such association are said to be *active*. All of the services for which each patient is *enrolled* are maintained at this level as well.

HCS

$$PP[\mathbb{P}, \mathbb{I}]$$

$$Service : \wp\, \mathbb{S}$$
$$SubService := \mathsf{dom}\ NestIn$$
$$NestIn : Service \pfun Service$$
$$NestIn^+ \cap \mathsf{id}\ Service = \emptyset$$

$$HProf : \wp\, \mathbb{H}$$
$$Assoc : HProf \rightleftarrows Service$$
$$Active := \mathsf{dom}\ Assoc$$

$$Roll : Patient \rightleftarrows Service$$

$$Service' = \emptyset\ ;\ HProf' = \emptyset$$

Several new queries will now be required for this class, e.g. the following:

The query subService, applied to any object of the class Health-Care System, subset of S with elements nested in a partial function from S.

$HCS?subServices(s \to S)$

$$s : Service$$
$$S = \{x : \mathbb{S} \bullet NestIn\ x = s\}$$

The query associations applied to any object of the class Health-Care System, subset of S with elements h turple x with elements from Assoc

$HCS?associations(h \to S)$

$$h : HProf$$
$$S = \{x : \mathbb{S} \bullet h \mapsto x \in Assoc\}$$

Informally document each of these queries, once again using natural language. (6)

4. A number of events will obviously be required for HCS, e.g. the following:

A new sub-service s nested in n can be *defined*, provided its name is unique.

$HCS!NewSubService(s, n)$

$$n: S;\ s := dom\ n;\ s'\ n$$

A new health professional can be *registered*, and given unique identifier h.

$HCS!RegisterHProf(\to h)$

$$h: H$$

Some patient p can be newly *enrolled* for (sub-)service s, provided they are not already enrolled there.

$HCS!EnrolPatient(p, s)$

$$p: P\ ;\ s : S;\ Rolls\ s := p$$

Formally specify each event, guided by its documentation. (10)

5. An initial model for the required "Patient Record System" may be introduced by further *refining* class HCS. This is specified as class $PRS[\,\mathbb{P}, \mathbb{I}, \mathbb{S}, \mathbb{H}, \mathbb{R}, \mathbb{F}\,]$:

A Patient Record System *extends* the Health-Care System, so as to provide a shared *database* of *patient records* (uniquely identified by elements from type $\mathbb{R}$). Each such record *refers to* one particular patient, and identifies both the *originator* and *destination* (as professional-service pairs). Automatically imposed are its DATE and TIME of entry. All of them include a *message* (of type TEXT), while some may also have one or more *attached* files (of type $\mathbb{F}$). Every record having the *same* originator and destination is said to be a *note*; otherwise, it corresponds to a medical *communication*.

PRS

$HCS[\,\mathbb{P}, \mathbb{I}, \mathbb{S}, \mathbb{H}\,]$

$PRec : \wp\, \mathbb{R}$

$Ref \;\; : PRec \to Patient$
$Orig : PRec \to HProf \times Service$
$Dest : PRec \to HProf \times Service$
$DaT : PRec \to \mathsf{DATE} \times \mathsf{TIME}$
$Msg \;\; : PRec \to \mathsf{TEXT}$
$Att \;\; : PRec \leftrightarrow \mathbb{F}$

$\{Note, Comm\}$: part $PRec$
$Note =$
$\{r : PRec \bullet Orig\, r = Dest\, r\}$

$PRec' = \emptyset$

Assume TEXT, DATE and TIME are *pre-defined* types, and that the usual order-relations $(<, \leq, \geq, >)$ apply to DATE or TIME values. Assume also that a function sorted_by is provided, where R sorted_by DaT constructs the *sequence* of record-identifiers from selected set R which is *ordered* (using $\leq$) by their DaT values.

The capabilities at this level are very *general*; the challenge is to exploit them in a way that supports *clinical practice* – by providing appropriate operations.

Consider for example a simple query, informally documented as follows:

The sequence S of record-identifiers (sorted by their entry date and time) can be *shown*, provided they refer to patient p, were entered on dates $\geq d$ and have a professional h of service s as the originator and/or destination.

$PRS?showFrom(p, d, h, s \to S)$

...

Formally specify this query. (10)

6. Consider also a simple event, with the following informal documentation:

A new note which refers to patient p can be *entered* with message m (but no attachments) by professional h of service s, provided p is enrolled for s and h is associated with s; doing so returns its unique record-identifier r.

$PRS!EnterNote(p, m, h, s \to r)$

h: $\mathbb{H}$; P : IP, m : TEXT;

S: $\mathbb{S}$; *r : IR*

Ref' r = P; msg'r = m;
Orig'r = h X S;

Formally specify this event. (10)