Towards Formal Specification of PaperLess Office Systems

Bambang Nurcahyo Prastowo
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
prastowo@ugm.ac.id

Ahmad Ashari
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
ashari@ugm.ac.id

Jazi Eko Istiyanto
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
jazi@ugm.ac.id

Khabib Mustofa
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
khabib@ugm.ac.id

Abstract—**The Center for Information and Communication Technology Services of Gadjah Mada University runs a kind of formal office communication system called PLO (stands for PaperLess Office). Developed to serve ad hoc needs of the center, the system evolves for several years without proper documentation. Several attempt store write the system failed due to the lack of the documentations. Judging from the fact that it has been serving the institution well for several years, the system can be treated as a user requirement. A formal specification is written based on it.**

Keywords—formal specification; paperless office; information system

I. INTRODUCTION

Since 2004, the Center for Information and Communication Technology Services of Gadjah Mada University runs an information system called PLO (stands for PaperLess Office). In terms of software engineering, the system serves as a rapid prototype of an office communication system.

Historically PLO replaced the mailing-list based office communication management which did not work well due to it's exponential growth as described in [1]. PLO project worked so well in the office such that it not only reduced annual paper use down to 10% of the previous one but also eliminated the need of conducting office meetings. Its communication threads are so well organized such that they also serve as minutes of meetings.

Being operational while developed incrementally by a single person, PLO's modules have never been inspected formally for their security, functionality and efficiency. Lacking documentations, numerous attempts to rewrite the codes failed.

In 2011, the government of Indonesia issue an official regulation for system conducting government electronic communications [2]. The formal specification of the paperless office is also needed to examine it's compliance to the regulation.

To build industrial grade PLO system, we need formal specification of the system. This document is the first attempt to document the PLO formally based on the operation PLO system as the prototype. Z specification language is utilized for its simplicity yet rich enough to capture the dynamic nature of office data [3].

II. STATE

In this section, we specify states of PLO elements and conditions among them that must be hold all the time. The conditions address issues of formal condition of data integrity as well as security requirements.

A. Basic Types

There are 4 basic types in PLO. They are :

$$[PERSON, MEMO, MEDIA, GROUP]$$

PERSON is the set of potential users, MEMO is the set of potentially written messages, MEDIA is a set of all kind of files which may be attached to memos, and GROUP is a set of identified sets of users. The 5th type LOCATION is added later to complete the specification of additional features.

We define *DateTime* as an alias to the natural number N. It denotes a type of time stamp, mandatory attributes of messages.

$$DateTime == N$$

B. The User

Users are PERSON who have access to the system.

$$
\begin{array}{l}
\hline
\textit{User} \\
\hline
user : \mathbb{P}\,PERSON \\
\hline
\end{array}
$$

To simplify the specification, usual attributes of person such as names, unique identifiers, sexes, and dates of birth are

omitted. Initially, there is one user, called admin which has the authority of setting up the operation of the system.

```
┌─ InitUser ──────────────────
│ ΔUser
│ admin : PERSON
├─────────────────────────────
│ user ' = {admin}
└─────────────────────────────
```

C. Message

Messages are MEMO written by the users at specific time.

```
┌─ Message ───────────────────
│ User
│ memo : ℙMEMO
│ authorisedBy : MEMO ⇸ PERSON
│ postedTime : MEMO ⇸ DateTime
├─────────────────────────────
│ dom authorisedBy = memo
│ ran authorisedBy ⊆ user
│ dom postedTime = memo
└─────────────────────────────
```

Initially, the *memo* set is empty.

```
┌─ InitMessage ───────────────
│ ΔMessage
├─────────────────────────────
│ memo' = ∅
└─────────────────────────────
```

A MEMO may be associated with a set of attached MEDIA files. A MEDIA can be attached to asset of MEMOs.

```
┌─ Attachment ────────────────
│ Message
│ file : ℙMEDIA
│ attachedTo : MEDIA ↔ MEMO
├─────────────────────────────
│ dom attachedTo = file
│ ran attachedTo ⊆ memo
└─────────────────────────────
```

D. Grouping

Grouping is used to define groups of users. The purpose of specifying groups is to make it shorter list of memo's target readers. The grouping is also used to control access.

```
┌─ Group ─────────────────────
│ User
│ group : ℙGROUP
│ member : GROUP ↔ PERSON
├─────────────────────────────
│ dom member = group
│ ran member = user
└─────────────────────────────
```

Initially, there is one *group* of users having the authority of setting up the operation of the user.

```
┌─ InitGroup ─────────────────
│ ΔGroup
│ ads : GROUP
├─────────────────────────────
│ group' = {ads}
└─────────────────────────────
```

E. Target

Each memo has at least one target of user which can be specified directly by mentioning the name of the user or indirectly via group membership by mentioning a group.

```
┌─ Target ────────────────────
│ Message
│ Group
│ userTarget : MEMO ⇸ ℙPERSON
│ groupTarget : MEMO ⇸ ℙGROUP
│ allUserTarget : MEMO ⇸ ℙPERSON
├─────────────────────────────
│ dom userTarget = memo
│ ran userTarget = ℙ user
│ dom groupTarget = memo
│ ran groupTarget = ℙ group
│ dom allUserTarget = memo
│ ∀m : memo • #(allUserTarget(m)) > 0
│ ∀m : memo •
│      allUserTarget(m) = userTarget(m)∪
│   {u : user | ∃ g : groupTarget(m) •
│        u = member (g)}
└─────────────────────────────
```

Users in a memo's target may access media attached to it.

```
┌─ MediaAccess ───────────────
│ Target
│ Attachment
│ mediaAccess : PERSON ↔ MEDIA
├─────────────────────────────
│ dom mediaAccess = user
│ ran mediaAccess = file
│ ∀ u : user ; f : file • (u, f ) ∈ mediaAccess ⟺
│      (∃m : memo •
│         u ∈ allUserTarget(m) ∧
│         (f ,m) ∈ attachedTo)
└─────────────────────────────
```

F. Comments

Comments are memos which follow with other memos. Comments are specified by defining global *isFollowing* function.

```
┌─ Comment ───────────────────
│ Target
│ isFollowing : MEMO ⇸MEMO
├─────────────────────────────
│ dom isFollowing ⊆ memo
│
```

ran *isFollowing* ⊂ *memo*
∀*m1,m2* : *memo* • *m1* = *isFollowing*(*m2*) ⇒
    postedTime(*m1*) > *postedTime*(*m2*) ∧
    allUserTarget(*m1*) = *allUserTarget*(*m2*)

G. Read

We specify read to mark time when users access memos.

─── *Read* ───────────────────────
  Target
  isRead : *PERSON* × *MEMO* ↛ *DateTime*
─────────
  dom *isRead* ⊆ *user* × *memo*
  ∀ *p* : *user* ; *m* : *memo* •
    p ∈ *allUserTarget*(*m*) ∧
      (*p*,*m*) ∈ dom *isRead* ⇒
        postedTime(*m*) < *isRead*(*p*,*m*)

III. OPERATION

PLO operations are procedures of querying the states which can modify them.

A. Listing Memos

The global function *listMemo* returns all main memos users have access to. Main memos are memos which are not comments. They do not follow any other memo. Listing memos does not change PLO's states.

─── *MemoList* ───────────────────
  Comment
  listMemo : *PERSON* ↛ ℙ*MEMO*
─────────
  dom *listMemo* = *user*
  ∀ *u* : *user* •
    listMemo(*u*) =
      {*m* : *memo* | *u* ∈ *allUserTarget*(*m*) ∧
      ¬ (∃ *n* : *memo* • (*m*, *n*) ∈ *isFollowing*)}

The return values of *listMemo* can be ordered by *postedTime* either descending or ascending.

B. Viewing a Memo

The operation of viewing a memo returns all comments following the memo.

─── *MemoView* ───────────────────
  Target
  Comment
  Δ*Read*
  reader? : *PERSON*
  toread? : *MEMO*
  now? : *DateTime*
  comment ! : ℙ*MEMO*
─────────
  reader? ∈ *user*
  toread? ∈ *memo*

reader? ∈ *allUserTarget*(*toread*?)
comment ! =
    {*m* : *memo* | (*m*, *toread*?) ∈ *isFollowing*}
∀ *p* : *PERSON*; *m* : *MEMO* •
    (*p*,*m*) ∈ dom *isRead* ⇒
      isRead(*p*,*m*) < *now*?
isRead' = *isRead*
    ∪{(*reader*?, *toread*?) ↦ *now*?}
    ∪{*c* : *comment* ! • (*reader*?, *c*) ↦ *now*?}

C. Notification

Notifications are sets of memos which have not been accessed by the users.

─── *Notify* ─────────────────────
  Read
  getNot : *PERSON* ↛ ℙ*MEMO*
─────────
  dom *getNot* = *user*
  ∀ *u* : *user* • *getNot*(*u*) =
    {*n* : *memo* | *u* ∈ *allUserTarget*(*n*) ∧
      (*u*, *n*) ∉ dom *isRead*}

D. Adding a Memo

The *AddMemo* schema specifies the behavior of the system when the state of Message is modified by adding a new one. The new memo brings authorship and optional attachment and target reader properties with it.

─── *AddMemo* ────────────────────
  Δ*Target*
  Δ*Attachment*
  author? : *PERSON*
  newMemo? : *MEMO*
  attachment? : ℙ*MEDIA*
  time? : *DateTime*
  newUserTarget? : ℙ*PERSON*
  newGroupTarget? : ℙ*GROUP*
─────────
  newMemo? ∉ *memo*
  author? ∈ *user*
  ∀*m* : *memo* • *time*? > *postedTime*(*m*)
  memo' = *memo* ∪ {*newMemo*?}
  newUserTarget? ⊆ *user*
  newGroupTarget? ⊆ *group*
  userTarget ' = *userTarget*∪
    {*newMemo*? ↦ *newUserTarget*?}
  groupTarget ' = *groupTarget*∪
    {*newMemo*? ↦ *newGroupTarget*?}
  attachedTo' = *attachedTo*∪
    {*a* : *attachment*? • *a* ↦ *newMemo*?}
  postedTime' = *postedTime*∪
    {*newMemo*? ↦ *time*?}

E. Adding a Comment

A comment id a memo to follow another memo.

```
┌─ AddComment ──────────────────────
│ AddMemo
│ ΔComment
│ parent? : MEMO
├───────────────────────────────────
│ parent? ∈ memo
│ isFollowing' =
│ isFollowing ∪ {newMemo? ↦ parent?}
└───────────────────────────────────
```

IV. ADDITIONAL FEATURES

In addition to the basic features, there are several additional ones that are useful to the users. There are news, blogs, event calendar and expressions. News and events have additional attributes are formally specified while blogs and expression are not explicitly specified since they are merely subsets of news.

A. News

News are memos targeted to all users.

```
┌─ News ────────────────────────────
│ Target
│ news : ℙMEMO
├───────────────────────────────────
│ news ⊆ memo
│ ∀ n : news • userTarget(n) = user
└───────────────────────────────────
```

B. Event Calendar

Calendar events are memos with additional information of locations, times and durations. The set of LOCATIONs are introduced to abstract locations where the events occur. There are two disjoint sets of LOCATIONs, *location* and *xlocation*; *location* is the set of LOCATION where events can occur at the same time and *xlocation* is the set of LOCATION where only one event can occur at the same time.

[LOCATION]

```
┌─ EventCalendar ───────────────────
│ Message
│ event : ℙMEMO
│ occur : MEMO ⇸ DateTime
│ duration : MEMO ⇸ N
│ location : MEMO ⇸ LOCATION
│ xlocation : MEMO ⇸ LOCATION
├───────────────────────────────────
│ event ⊆ memo
│ dom occur = event
│ dom duration = event
│ dom location ∪ dom xlocation = event
│ dom location ∩ dom xlocation = ∅
│ ∀ a, b : dom xlocation •
│     occur (a) + duration(a) ≤ occur (b) ∨
│     occur (b) + duration(b) ≤ occur (a)
└───────────────────────────────────
```

V. SECURITY ISSUES

PLO security relies on the traditional user id password login system. Logged user are in the log set which initially is empty. Password is a set of natural numbers assigned to the users.

A. Login

Login is a set of users authorized to access the system.

```
┌─ Log ─────────────────────────────
│ User
│ login : ℙPERSON
│ password : PERSON ⇸ N
├───────────────────────────────────
│ login ⊆ user
│ dom password = user
└───────────────────────────────────
```

Initially login is empty.

```
┌─ InitLogin ───────────────────────
│ ΔLog
├───────────────────────────────────
│ login' = ∅
└───────────────────────────────────
```

Users log in by submitting their pair of *userids* and *passwords*.

```
┌─ Login ───────────────────────────
│ ΔLog
│ userid? : PERSON
│ password? : ℕ
├───────────────────────────────────
│ password? = password(userid?)
│ password' = password ∪ {userid? ↦ password?}
└───────────────────────────────────
```

B. Logged Operation

All operations should be logged. For example, listing memo should be logged. The global variable *actor?* is introduced as the one who operate the system.

```
┌─ LoggedMemoList ──────────────────
│ MemoList
│ Log
│ actor? : PERSON
│ listing! : ℙMEMO
├───────────────────────────────────
│ actor? ∈ login
│ listing! = listMemo(actor?)
└───────────────────────────────────
```

Person who add a comment have to be in the set of it's parent memo

```
┌─ LoggedAddComment ────────────────
│ AddComment
│ Log
│ actor? : PERSON
```

$$actor? \in login$$
$$actor? = author?$$
$$actor? \in allUserTarget(parent?)$$

VI. Concluding Remarks

A formal specification of the PaperLess Office system has been written. Some organizational and security issues have been addressed in the specification. The formal specification has been fully implemented. The success of the system relies on the design which does not place the system as a resemblance the traditional office business as usual but places the system as the core engine to run the office electronically.

References

[1] Z.S. Demirdjian, Toward taming the monster in electronic mail, 7(1): I,II, September 2005.

[2] E.E. Mangindaan, "Pedoman Umumdi Lingkungan Instansi Pemerintah", Nomor 6, Kementrian Negara Pendayagunaan aparatur Negara dan Reformasi Birokrasi, 2011.

[3] Mike McMorran and steve Powel, "Z Guide for Beginners", Alfred Waller Ltd, March 1993.