۵-۱- مدلسازی ساختاری

در این بخش قصد داریم مدلسازی پروژه را در زبان مدلسازی UML انجام دهیم و همچنین آن را با استفاده از زبان OCL دقیق‌تر نمائیم تا اینکه بتوانیم آن را با استفاده از چهارچوب ترجمه ارائه شده در فصل ٤ راحت‌تر به زبان AMN تبدیل کنیم. همانطور که مشاهده خواهیم کرد در این مدل‌سازی فقط از تعداد اندکی از ابزارهای UML استفاده کرده‌ایم. استفاده از تعداد زیادی دیاگرام به دقیق‌تر بودن مدل کمک می‌کند، ولی در این پروژه نیازی به این کار احساس نشد.

۵-۱-۱- دیاگرام کلاس فاز تحلیل

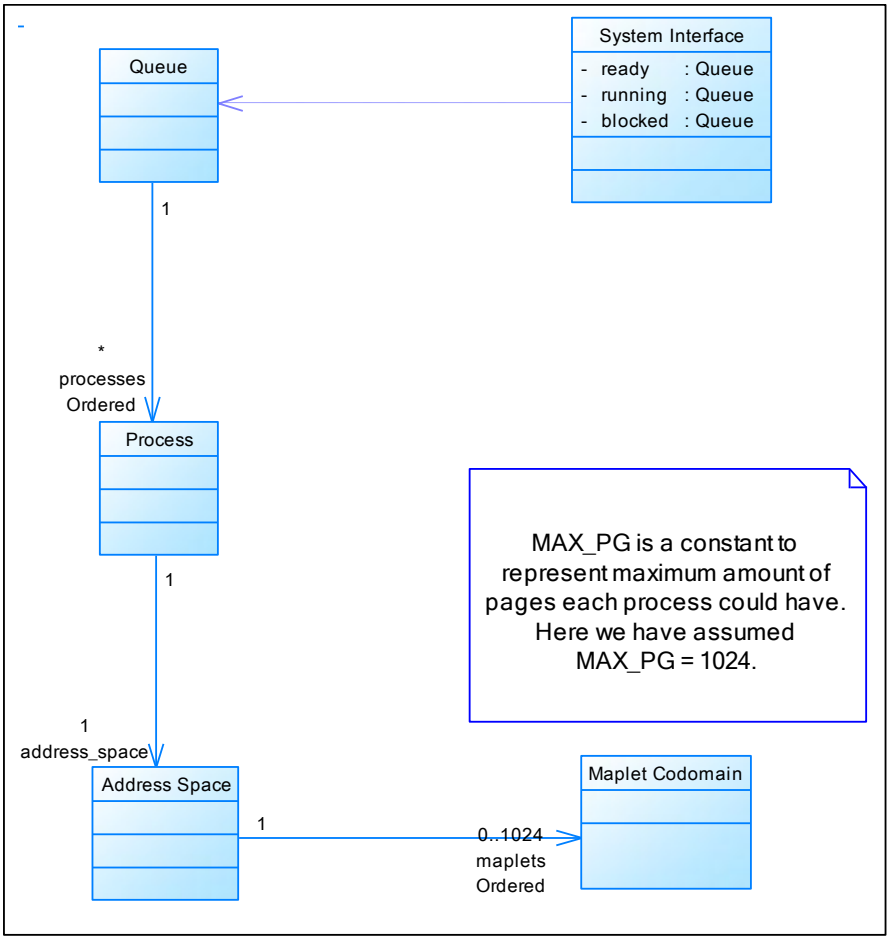
با بررسی نیازمندی‌های سیستم و نیز مرور شرح موارد استفاده به این نتیجه می‌رسیم که در سیستمی که در حال طراحی آن هستیم اشیائی از این نوع‌ها موجود است:

- فرآیندها؛
- فضاهای آدرس؛
- پیام‌ها؛
- صف‌ها.

البته روابط بین این کلاس از اشیاء را نیز می‌توانیم باز با بررسی شرح موارد استفاده و نیز قواعد حاکم بر سیستم استخراج کنیم. در شکل ۱ (صفحه بعد) می‌توانید دیاگرام کلاس‌های سیستم را مشاهده کنید که در فاز تحلیل آن را آماده کرده‌ایم.

همانطور که مشاهده می‌کنید در این دیاگرام کلاس صرفاً نام کلاس‌ها و روابط آن‌ها آمده است، چرا که این دیاگرام یک دیاگرام فاز تحلیل می‌باشد و در این مرحله نیاز به وارد شدن به جزئیات را نداریم. دقت کنید که علاوه بر کلاس‌های ذکر شده دو کلاس دیگر را نیز در دیاگرام شکل ۱ می‌توان یافت، کلاس System Interface، و کلاس Maplet Codomain. کلاس System Interface یک کلاس استاتیک بوده و در حقیقت فقط یک نمونه از آن ساخته می‌شود و عملگرهای آن نیز مجموعه توابع API سیستم را تشکیل می‌دهند. از آنجایی که وظیفه اشیاء از نوع فضای آدرس نگاشت کردن آدرس‌های مجازی به آدرس‌های حقیقی است، این اشیاء باید شامل آرایه‌ایی از اشیاء باشند که وظیفه

این اشیاء این باشد که یا بصورت مستقیم و یا بصورت غیر مستقیم یک آدرس مجازی را به یک آدرس حقیقی نگاشت کنند.



شکل ۱- دیاگرام کلاس‌های اشیاء سیستم در فاز تحلیل.

در این دیاگرام سایر محدودیت‌های سیستم را نیز می‌توان مشاهده کرد. برای مثال این که در سیستم فقط سه صف وجود خواهد داشت، آماده، در حال اجرا، و بلاک شده؛ همچنین یک صف می‌تواند چند تا فرآیند در خود نگه دارد ولی هر فرآیند تنها به یک صف تعلق دارد. در این دیاگرام به همراه روابط Association می‌توان Stereotypeهایی را هم مشاهده کرد که طراحی را محدودتر و دقیق‌تر می‌کنند؛ برای مثال اسم نقش processes برای کلاس Process به همراه یک Stereotype با عنوان Ordered در رابطه همنشینی که با کلاس Queue دارد به این معنی است که هر شئ Queue دارای دنباله‌ایی از اشیاء فرآیند می‌باشد. چند یادداشت نیز می‌تواند برای ارائه توضیحات بیشتر مؤثر باشد. در همین جا لازم است به این نکته هم اشاره کنیم که در این سیستم دو ثابت کلی تعریف می‌شود که این سیستم با این ثوابت نمونه‌سازی می‌شوند، یکی ثابت MAX_PG و دیگری ثابت MAX_PR

می‌باشد. این دو ثابت سقف تعداد صفحات موجود در یک فضای آدری و سقف تعداد فرآیندهای موجود در سیستم را تعیین می‌کنند.

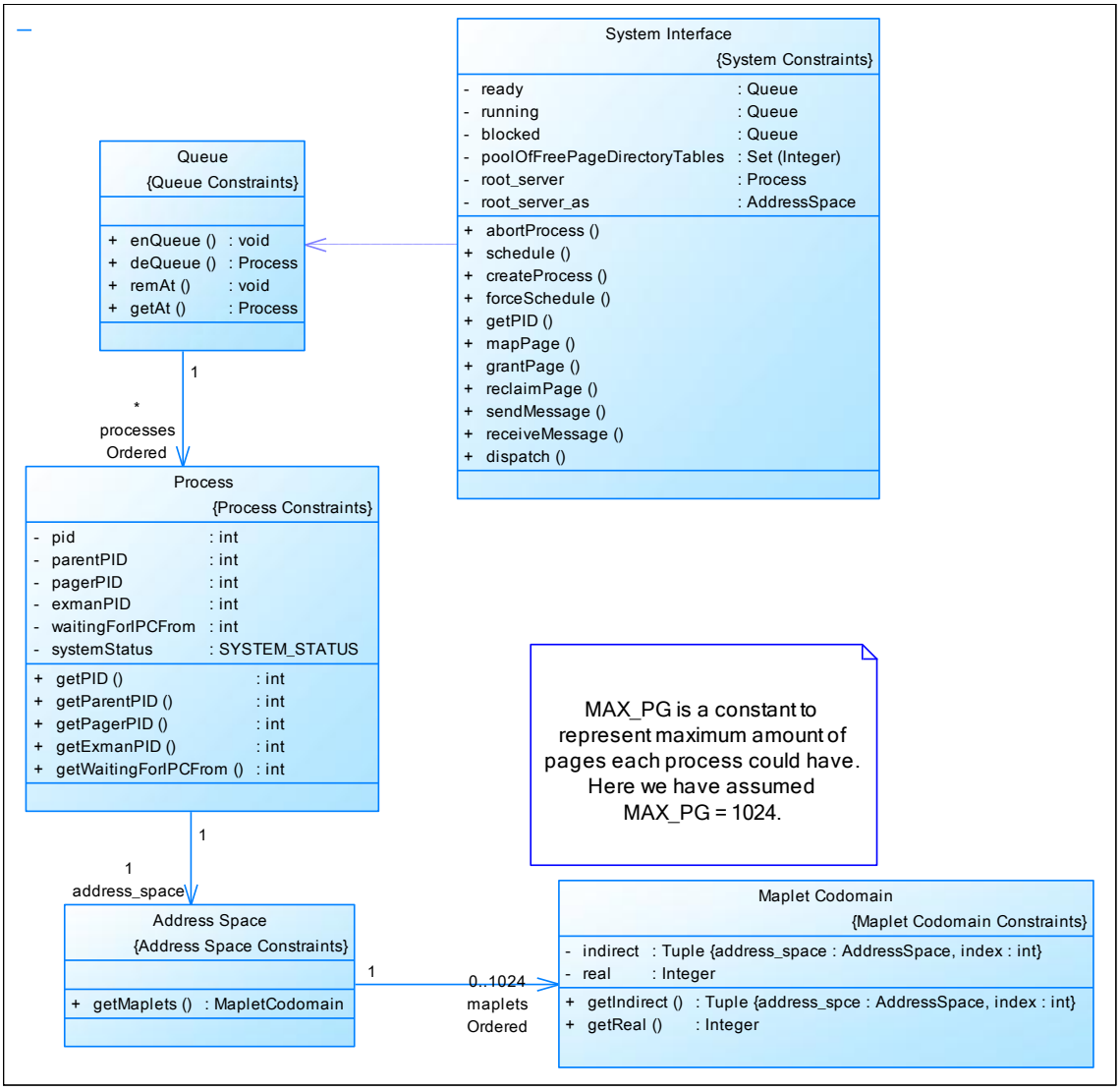
در فاز تحلیل به جز این دیاگرام کلاس و شرح موارد استفاده چیز دیگری لازم به نظر نمی‌رسد. دیاگرام‌های فعالیت برای بیان دقیق‌تر رفتار سیستم در حین رجوع به هر یک از موارد استفاده بسیار مؤثر است ولی از آنجایی که شرح موارد استفاده را به اندازه کافی بدون ابهام و دقیق نوشته‌ایم نیازی نیست که در فاز تحلیل اقدام به طراحی دیاگرام فعالیت نمائیم.

۵-۱-۲- دیاگرام‌های فاز طراحی

در این مرحله از توسعه سیستم نیاز داریم که سیستم را قدری بیشتر با جزئیات بیشتر توصیف و مدلسازی کنیم. دیاگرام کلاس شکل ۱ را با قدری جزئیات بیشتر به گونه‌ای بیان خواهیم کرد که ضمن حفظ عدم پیچیدگی دارای جزئیات لازم برای تکمیل شدن توسط OCL است.

در شکل ۲ می‌توانید دیاگرام کلاس مربوط بـه فاز طراحی را مشاهده کنید. همانطور که مشـــاهده می‌کنید در این دیاگرام کلاس محدودیت‌های بیشتری اضافه شده است، برای مثال برخی از کلاس‌ها دارای محدودیت زبان OCL است که برای مثال در مورد کلاس System Interface محدودیت {System Constraints} به این معنی است که کلاس System Interface با استفاده از عبارت‌های زبان OCL محدودتر و دقیق‌تر شده است. استفاده از زبان OCL در این مرحله برای دقیق‌تر کردن توصیف و مناسب کردن آن برای ترجمه به زبان AMN مؤثر می‌باشد.

در این دیاگرام خصیصه‌ها و عملگرهای بیشتری را از کلاس‌ها مشاهده می‌کنیم. توصیف دقیق‌تر رفتار این عملگرها و خواص این خصیصه‌ها را در ادامه در بخش مربوط به بیان رفتار سیستم توسط OCL بیان خواهیم کرد.

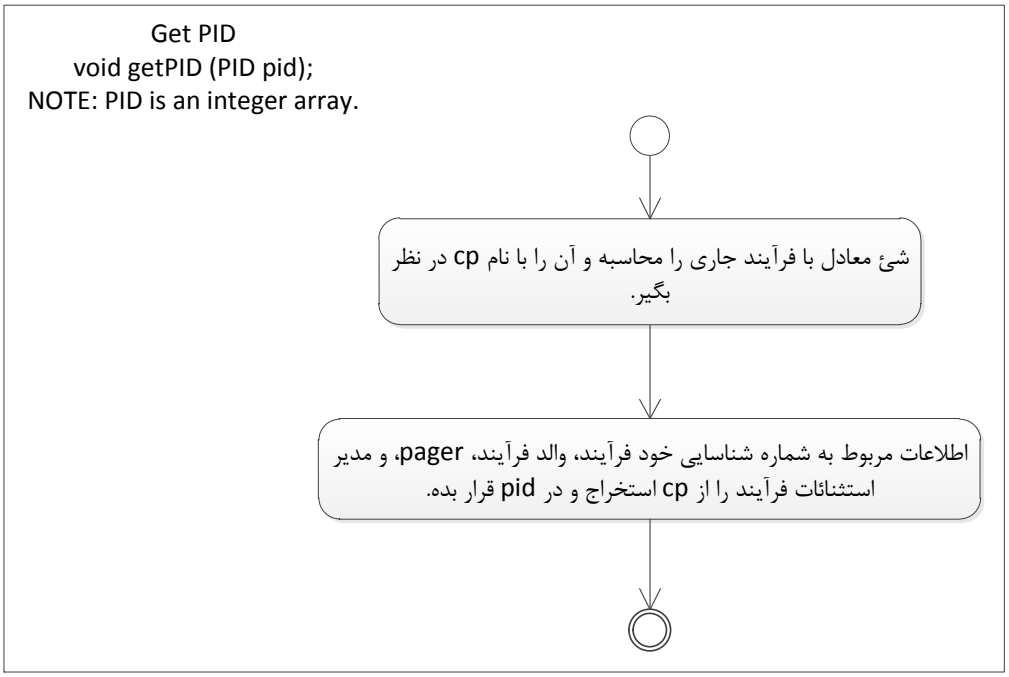


شکل ۲- دیاگرام کلاس‌های اشیاء سیستم در فاز طراحی.

۵-۲- مدلسازی رفتاری

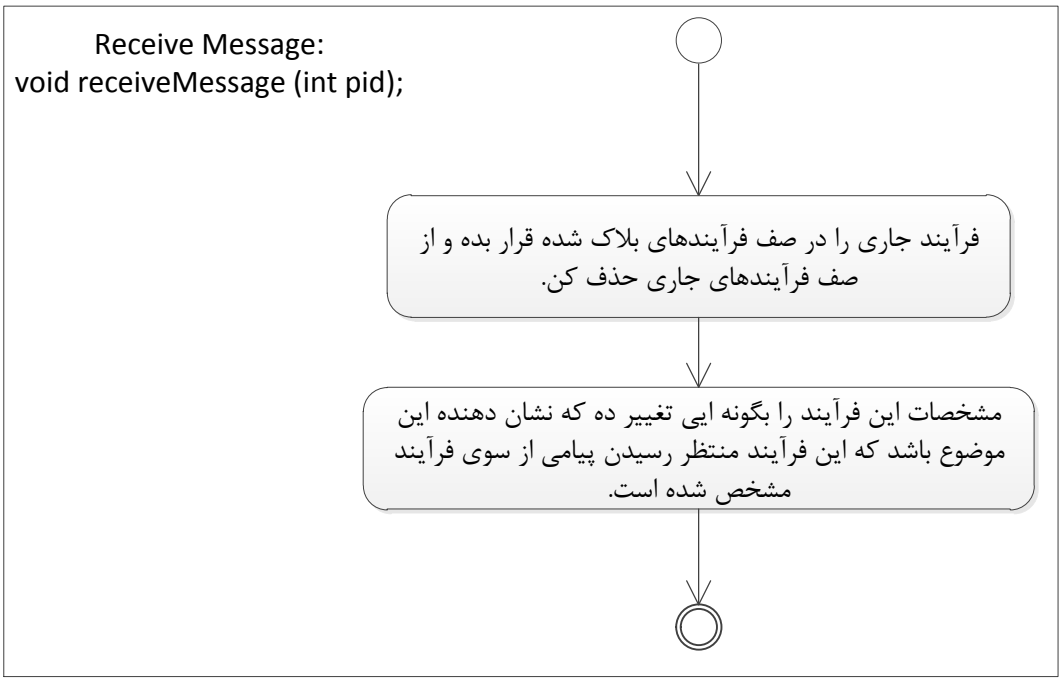
قصد داریم در این بخش مدل رفتاری سیستم را ارائه دهیم. دیاگرام‌های فعالیت از جمله ابزارهای UML برای مدلسازی رفتاری سیستم‌ها می‌باشد. از آنجایی که رفتارهای سیستمی که در حال توسعه آن هستیم بسیار پیچیده است، وارد جزئیات نشده و دیاگرام‌های فعالیت فاز تحلیل را مستقیماً به توصیف B تبدیل خواهیم کرد. در ادامه این بخش دیاگرام‌های فعالیت را برای تک تک موارد استفاده مشاهده خواهیم کرد؛ Actionها در این دیاگرام‌ها به زبان فارسی نوشته شده‌اند و در اصل پالایش شده و دقیق‌تر شده‌ی شرح موارد استفاده می‌باشد. این دیاگرام‌ها در اصل شروعی برای دقیق‌تر کردن مدلسازی رفتارهای پیچیده می‌باشد. دقت کنید که محدودیتی در این دیاگرام‌ها وجود دارد و آن هم این است که *اجرای همزمان Actionها باید اثری معادل با اجرای ترتیبی آن‌ها داشته باشد*. این محدودیت موجب می‌شود که بتوانیم هر Action را بصورت یک عبارت AMN بنویسیم، بدین ترتیب ترجمه مدل UML به مدل B خیلی ساده‌تر می‌شود.

در ارائه هر یک از این دیاگرام‌ها ترتیبی وجود ندارد و آن‌ها را با ترتیب دلخواه آورده‌ایم؛ در ضمن از آنجایی که در این سیستم هر مورد استفاده معادل با یک فراخوان سیستم است، به ازای هر دیاگرام فعالیت صورت فراخوان سیستم آن مورد استفاده را هم ذکر کرده‌ایم.

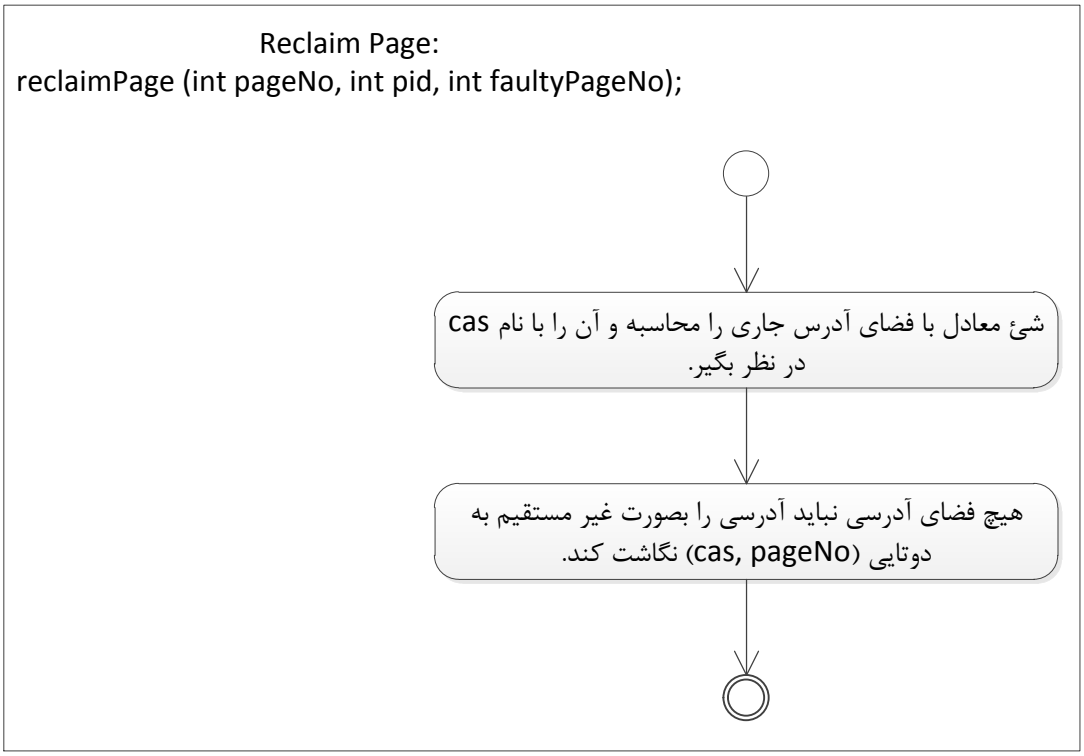


شکل ۳- دیاگرام فعالیت (در فاز تحلیل) مورد استفاده Get PID.

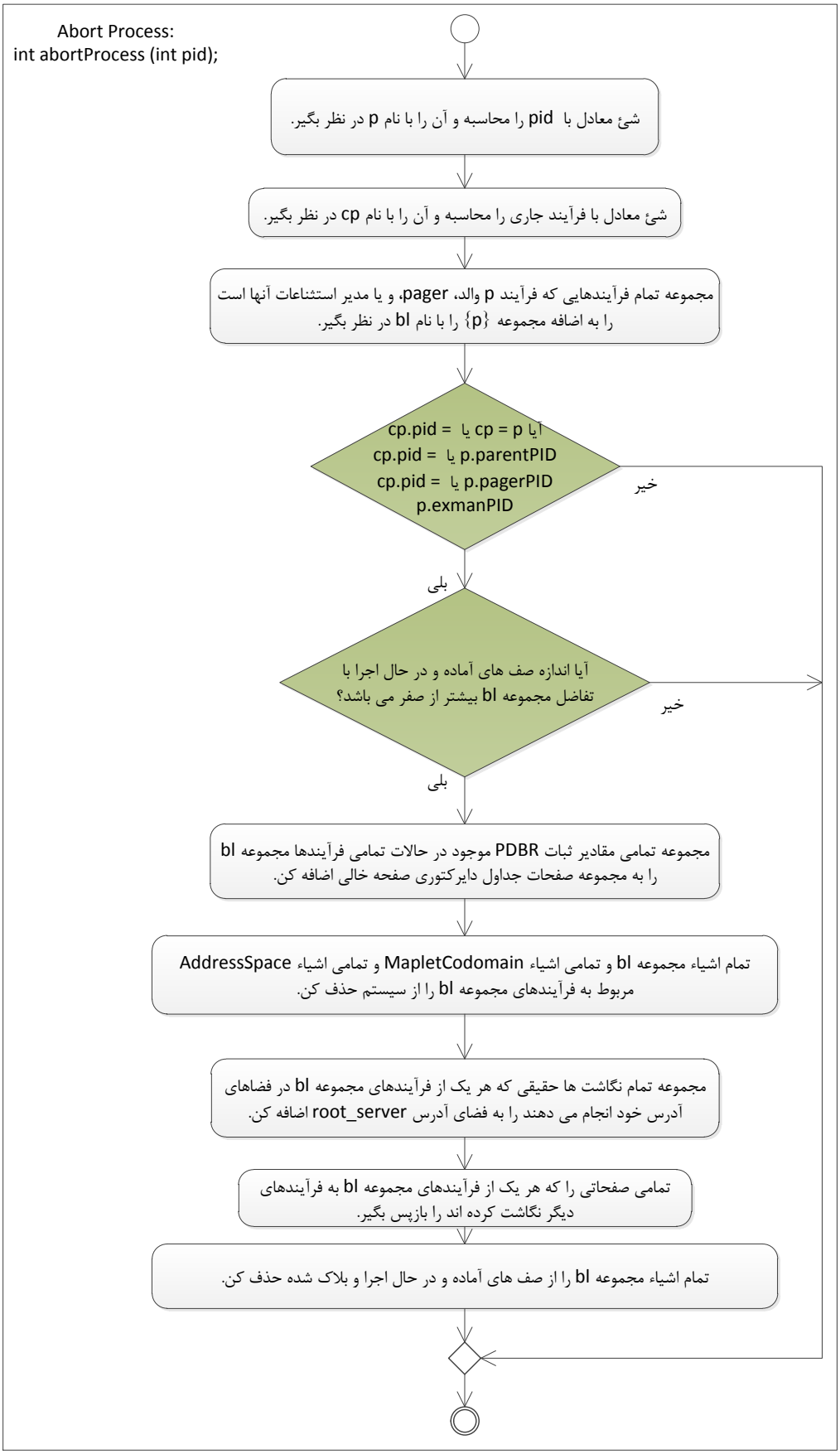
شکل ٤- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Create Process.



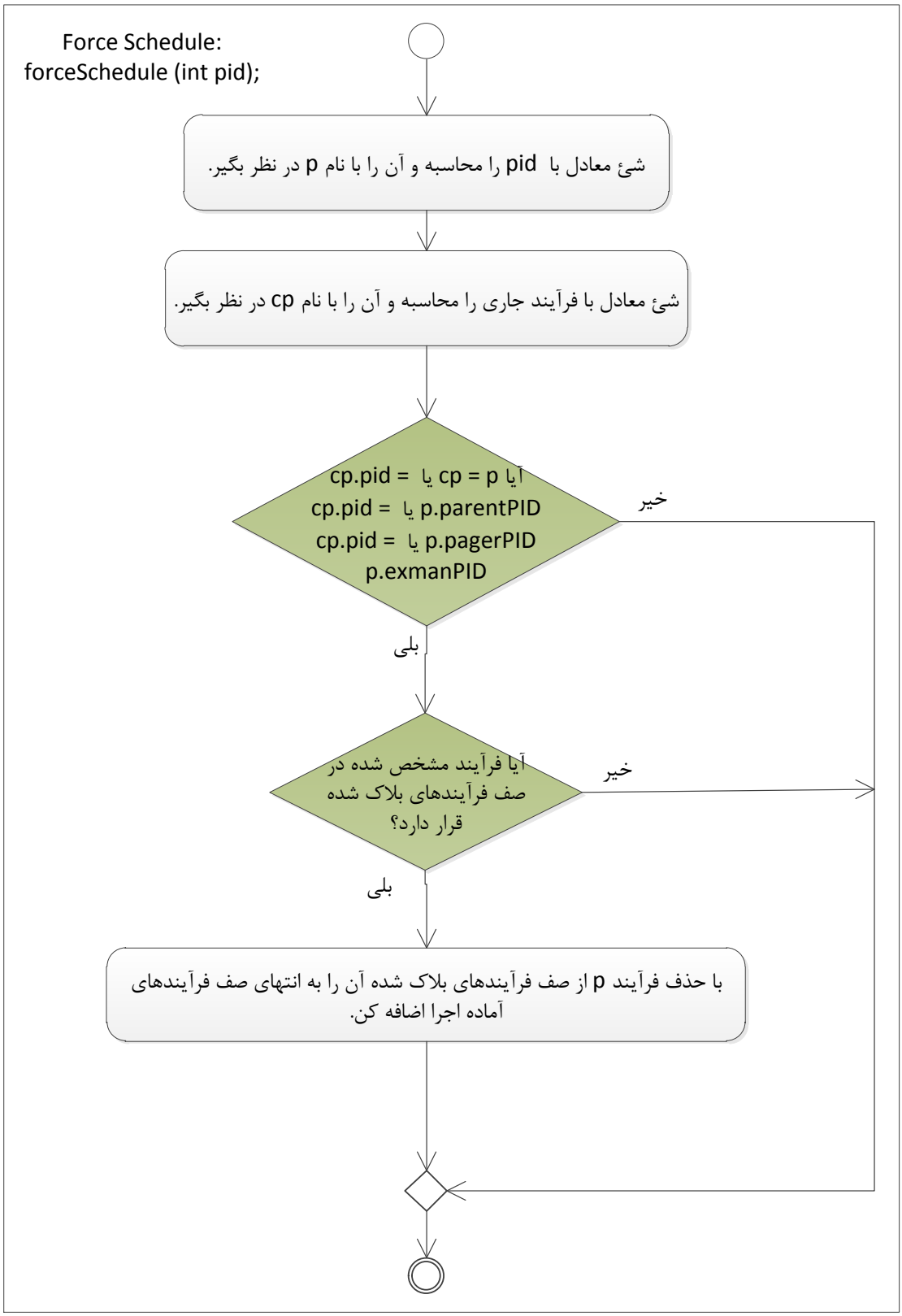
شکل ٥- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Receive Message.



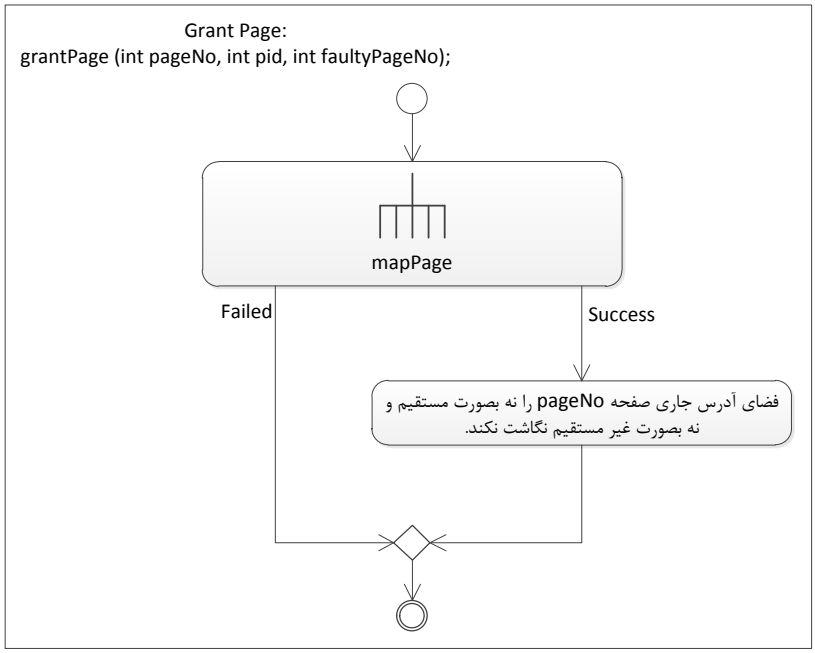
شکل ٦- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Reclaim Page.



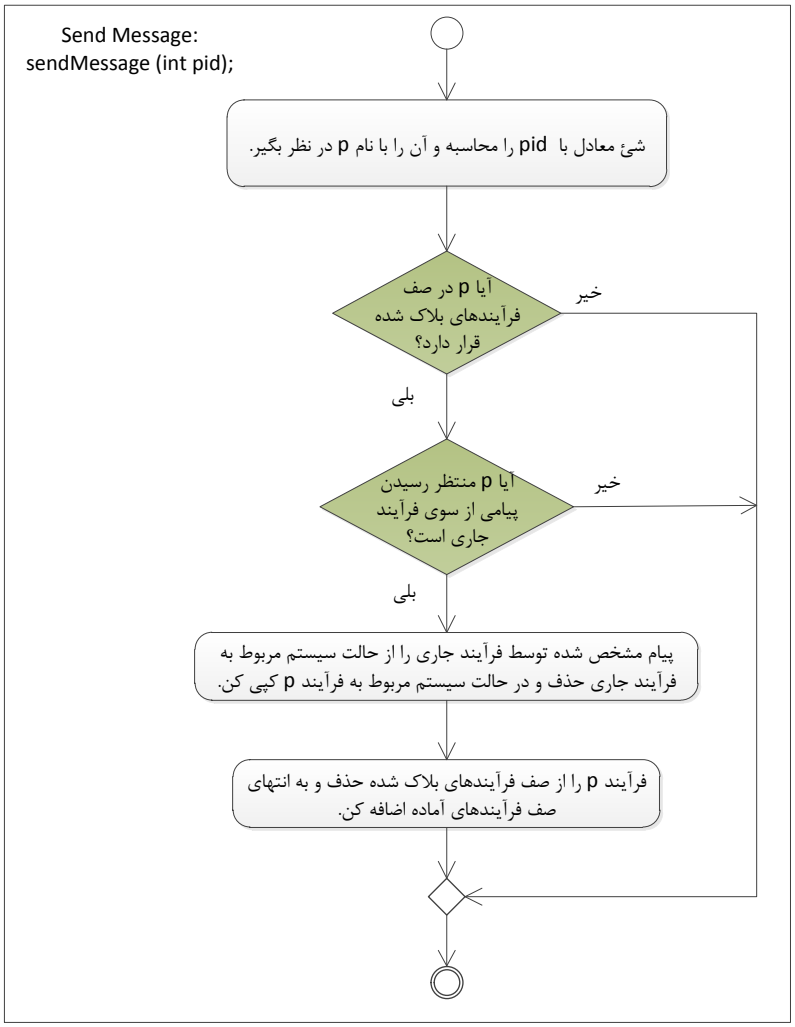
شکل ۷- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Abort Process.



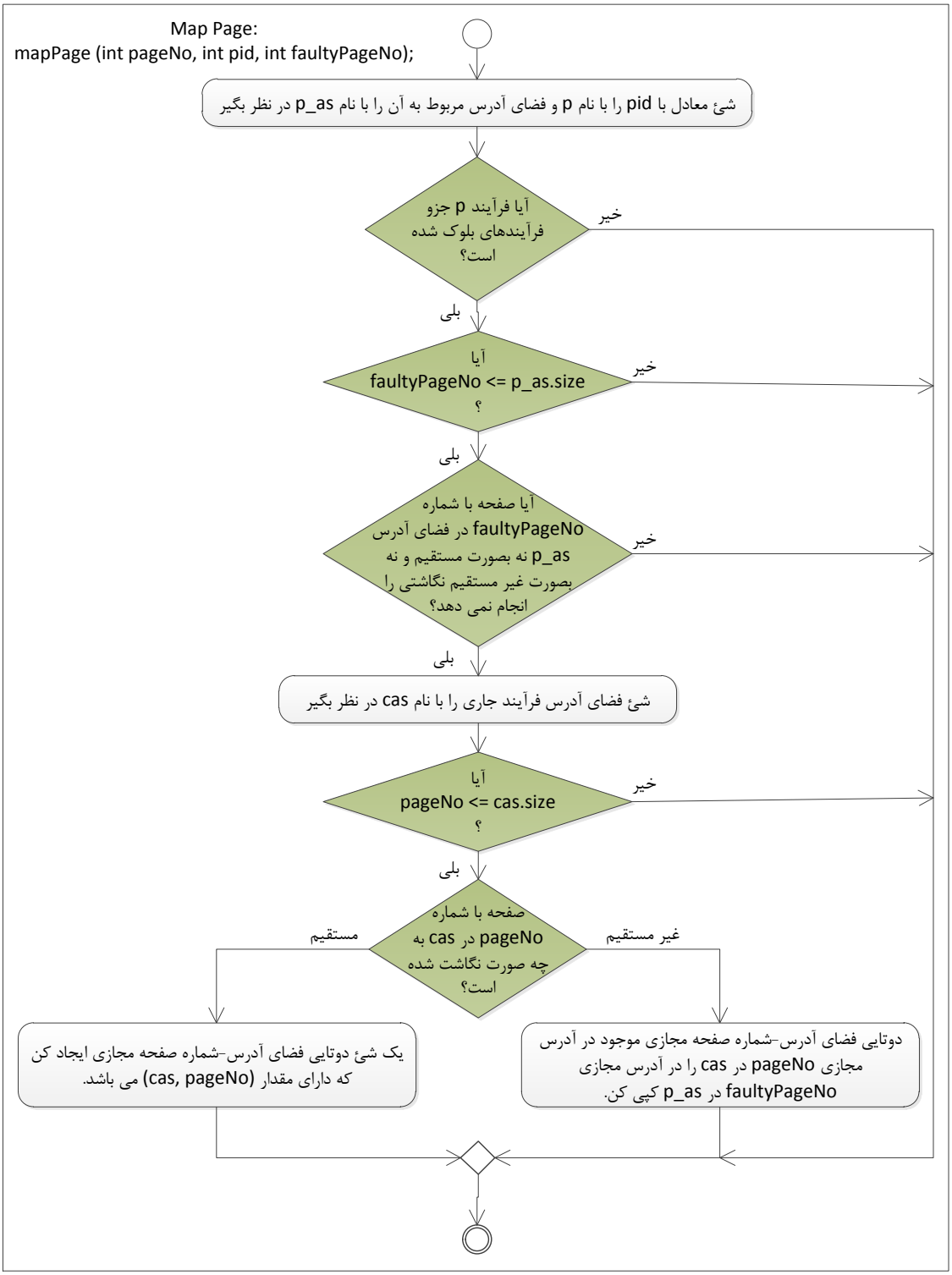
شکل ۸- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Force Schedule.



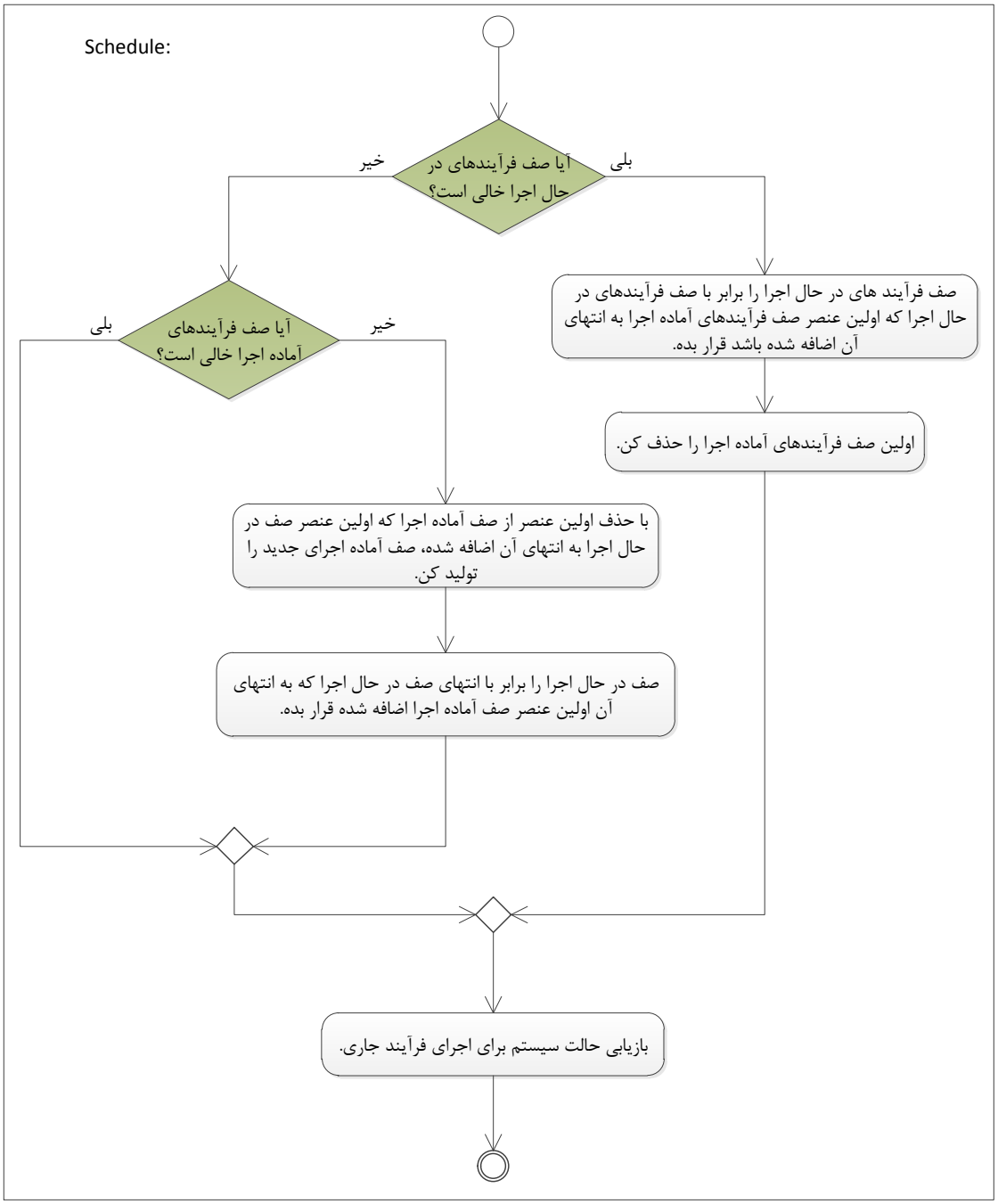
شکل ۹- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Grant Page.



شکل ۱۰- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Send Message.



شکل ۱۱- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Map Page.



شکل ۱۲- دیاگرام فعالیت (فاز تحلیل) مورد استفاده Schedule.

۵-۳- مدلسازی رفتاری با استفاده از OCL

قصد داریم در این بخش با استفاده از زبان OCL، مدل UML خود را که تا کنون توسعه داده‌ایم دقیق‌تر کنیم. این کار به نظام‌مند بودن روند ترجمه کمک بزرگی می‌کند. با این کار می‌توانیم قوانین همیشه حاکم بر روی اشیاء و نیز کل سیستم را بصورت دقیق در دسترس داشته باشیم، بدین ترتیب قادر خواهیم بود این قوانین که بصورت عبارات OCL بیان شده‌اند را به زبان AMN ترجمه کنیم؛ حتی درستی ترجمه خود را نیز اثبات کنیم، اما از آنجایی که این کار مستلزم صرف وقت زیادی می‌باشد در این پروژه نتوانستیم درستی عمل ترجمه را اثبات کنیم.

قواعد حاکم بر سیستم در کلاس System Interface.

```
context SystemInterface inv:
   running.processes->size () <= 1 and
   ready.processes->forAll (pr | ready.processes->count () = 1) and
   blocked.processes->forAll (pr | blocked.processes->count () = 1) and
   (
   let
      e1 = running.processes->asSet ()
   in
      let
         e2 = ready.processes->asSet ()
      in
         let
            e3 = blocked.processes->asSet ()
         in
            e1->union (e2->union (e3)) = Process.allInstances () and
            e1->intersection (e2->intersection (e3)) = {}
   ) and
   Process.allInstances ()->includes (root_server) and
   running.processes->size () + ready.processes->size () > 0 and
    (poolOfFreePageDirectoryTables->size () + Process.allInstances ()->size ()) >= MAX_PR
```

پیش‌شرط‌ها و پس‌شرط‌های عملگرهای کلاس System Interface:

```
context SystemInterface::abortProcess (pid : Integer)
pre: running.processes->size () > 0 and Process.allInstances ()->exists (pr | pr.getPID () = pid) and
   Process.allInstances ()->size () > 0 and pid <> root_server.getPID ()
post: let
         cp_obj = running@pre.processes.at (1)
      in
         let
            param_obj = (Process.allInstances ()->iterate (pr; acc = {} |
               if pr.getPID () = pid then acc->including (pr) else acc endif)->asSequence ()->at (1)
          in
             if (param_obj.getPID () = cp_obj.getPID () or
                param_obj.getParentPID () = cp_obj.getPID () or
                param_obj.getPagerPID () = cp_obj.getPID () or
                param_obj.getExmanPID () = cp_obj.getPID ()) then
                   if ((running@pre.processes->union (ready@pre.processes))->exists (pr | pr = param_obj) or
                      (running@pre.processes->size () + ready@pre.processes->size ()) = 1) then false
                   else
                      let
                         black_list = Process.allInstances ()->iterate (pr; acc = {} | if (pr.getPID () = pid or
                            pr.getParentPID () = pid or pr.getPagerPID () = pid or pr.getExmanPID () = pid) then acc->including (pr) else acc)
                      in
                         let
                            res = black_list->interate (pr; ml : Set (MapletCodomain); ml = {} |
                               ml.including (pr.getAddressSpace ().getMaplets ()->iterate (mm; acc = {} |
                                  if mm.getType () = REAL then acc.including (mm) else acc endif)))
                         in
                            root_server.getAddressSpace ().maplets = root_server.getAddressSpace ().maplets@pre->union (res)
             else false
             endif

context SystemInterface::mapPage (pageNo : Integer, pid : Integer, faultyPage : Integer)
pre: Process.allInstances ()->exists (pr | pr.getPID () = pid) and running.processes->size () > 0
post: let
         param_obj = (Process.allInstances ()->iterate (pr; acc = {} |
               if pr.getPID () = pid then acc->including (pr) else acc endif)->asSequence ()->at (1)
      in
         if blocked@pre.processes.exists (pr | pr = param_obj) then
            if (param_obj.getAddressSpace ().getMaplets ()->size () > faultyPage) then
               let
                  cp_obj = running.processes.at (1)
               in
                  if (cp_obj.getAddressSpace ().getMaplets ()->size () > pageNo) then
```

```
                    param_obj.getAddressSpace ().getMaplets ()->at (faultyPage).getIndirect () = Tuple {address_space = cp_obj.getAddressSpace,
                                                                                                        index = pageNo}
                 else false
           else false
           endif
        else false
        endif

context SystemInterface::grantPage (pageNo : Integer, pid : Integer, faultyPage : Integer)
pre: Process.allInstances ()->exists (pr | pr.getPID () = pid) and running.processes->size () > 0 and
   running.processes->asSet ()->forAll (pr | pr.getPID () <> pid)
post: let
         param_obj = (Process.allInstances ()->iterate (pr; acc = {} |
               if pr.getPID () = pid then acc->including (pr) else acc endif)->asSequence ()->at (1)
      in
         if blocked@pre.processes.exists (pr | pr = param_obj) then
            if (param_obj.getAddressSpace ().getMaplets ()->size () > faultyPage) then
               let
                  cp_obj = running.processes.at (1)
               in
                  if (cp_obj.getAddressSpace ().getMaplets ()->size () > pageNo) then
                     param_obj.getAddressSpace ().getMaplets ()->at (faultyPage).getIndirect () = Tuple {address_space = cp_obj.getAddressSpace,
                                                                                                         index = pageNo} and
                     let mm = cp_obj.getAddressSpace ().getMaplets ()->at (pageNo)
                     in mm.getReal () = Unknown and mm.getIndirect () = Unknown
                  else false
            else false
            endif
         else false
         endif

context SystemInterface::reclaimPage (pageNo : Integer)
pre: running.processes->size () > 0
post: let
         cas = running.processes->at (1).getAddressSpace ()
      in
         AddressSpace.allInstances ()->forAll (as | as.getMaplets ()->forAll (ml | ml.isIndirect () implies not (ml.getAddressSpace () = cas and
                                                                                  ml.getIndex () = pageNo)))


context SystemInterface::forceSchedule (pid : Integer)
pre: Process.allInstances ()->exists (pr | pr.getPID () = pid) and running.processes->size () > 0
post: let
         cp_obj = running.processes->at (1)
      in
         if blocked@pre.processes->exists (pr | pr = cp_obj) then
```

```
            blocked.processes = blocked@pre.processes->excluding (pr) and
            ready.processes = ready@pre.processes->append (pr)
         else false
         endif


context SystemInteface::createProcess (pagerPID : Integer, exmanPID : Integer, addressSpaceSz : Integer)
pre: running.processes->size () > 0 and Process.allInstances ()->size () < max_pr and
   poolOfFreePageDirectoryTables->size () > 0 and addressSpaceSz <= MAX_PG and
   Process.allInstances ()->exists (pr | pr.getPID () = pagerPID) and
   Process.allInstances ()->exists (pr | pr.getPID () = exmanPID)
post: let
         cp_obj = running.processes.at (1)
      in
         ready.processes->exists (pr | pr.getParentPID () = cp_obj.getPID and pr.getPagerPID () = pagerPID and pr.getExmanPID () = exmanPID and
            pr.getAddressSpace ().getMaplets ()->size () = addressSpaceSz)


context SystemInteface::getPID (parentPID : Integer out, pagerPID : Interger out, exmanPID : Integer out) : Integer
pre: running.processes->size () > 0
post: let
         cp_obj = running.processes.at (1)
      in
         result = cp_obj.getPID () and
         parentPID = cp_obj.getParentPID () and
         pagerPID = cp_obj.getPagerPID () and
         exmanPID = cp_obj.getExmanPID ()

context SystemInterface::schedule ()
pre: true
post: if running@pre.processes->size () < 1 then
         let
            xx = ready@pre.processes->first ()
         in
            ready.processes = ready@pre.processes->excluding (xx) and
            running.processes = running@pre.processes->append (xx)
      else
         if ready@pre.processes->size () < 1 then true
         else
            let
               xx = ready@pre.processes->first ()
            in
               running.processes = <xx>
               ready.processes = ready@pre.processes->excluding (xx)
         endif
      endif
```

```
context SystemInterface::receiveMessage (pid : Integer)
pre: running.processes->size () > 0 and ready.processes->size () > 0
post: let
         cp = running.processes@pre->at (1)
      in
         cp.getWaitingForIPCFrom () = pid and
         not running.processes->exists (pr | pr = cp) and
         blocked.processes->exists (pr | pr = cp)


context SystemInterface::sendMessage (pid : Integer)
pre: running.processes->size () > 0 and Process.allInstances ()->exists (pr | pr.getPID () = pid)
post: let
         param_obj = (Process.allInstances ()->iterate (pr; acc = {} |
               if pr.getPID () = pid then acc->including (pr) else acc endif)->asSequence ()->at (1)
      in
         ready.processes->exists (pr | pr = param_obj)
```

قواعد حاکم بر سیستم در کلاس Process:

```
context Process inv:
   Process->allInstances ()->isUnique (pr | pr.pid) and
   Process->allInstances ()->size <= MAX_PR
```

پیش‌شرط‌ها و پس‌شرط‌های عملگرهای کلاس Process:

```
context Process::getPID () : Integer
pre: true
post: result = pid


context Process::getParentPID () : Integer
pre: true
post: result = parentPID


context Process::getPagerPID () : Integer
pre: true
post: result = pagerPID


context Process::geExmanPID () : Integer
pre: true
post: result = exmanPID
```

```
context Process::getWaitingForIPCFrom () : Integer
pre: true
post: result = waitingForIPCFrom
```

قواعد حاکم بر سیستم در کلاس *Address Space*:

```
context AddressSpace inv:
   let
      e1 = maplets->iterate (ml; acc = <> |
         if ml.isIndirect () then acc->append (ml) else acc endif)->asSet ()
   in
      e1->forAll (ml | ml.getIndex () < MAX_PG)
```

پیش‌شرط‌ها و پس‌شرط‌های عملگرهای کلاس *Address Space*:

```
context AddressSpace::getMaplets () : Squence (MapletCodomain)
pre: true
post: result = maplets
```

قواعد حاکم بر سیستم در کلاس *Maplet Codomain*:

```
context MapletCodomain inv:
   real <> Unknown implies (indirect = Unknown) and
   indirect <> Unknown implies (real = Unknown and indirect.index <= indirect.address_space.GetMaplets ()->size () and
                                (let
                                   mm = indirect.address_space.getMaplets ()->at (index)
                                in
                                   mm.getReal () <> Unknown or mm.getIndirect () <> Unknown))
```

پیش‌شرط‌ها و پس‌شرط‌های عملگرهای کلاس *Maplet Codomain*:

```
context MapletCodomain::isIndirect () : Boolean
pre: true
post: result = indirect

context MapletCodomain::getIndex () : Integer
pre: true
post: result = index

context MapletCodomain::getAddressSpace () : AddressSpace
pre: true
post: result = address_space
```

۵-۴- نتیجه‌گیری

در این فصل سیستمی که در حال توسعه آن هستیم را با استفاده از زبان مدلسازی UML، در دو فاز تحلیل و طراحی مدلسازی کردیم. در مورد مدلسازی ساختاری، از دیاگرام کلاس استفاده کردیم، و در فاز طراحی دیاگرام‌های فاز تحلیل را با جزئیات بیشتری بیان کردیم. در مورد مدلسازی رفتاری نیز، از دیاگرام‌های فعالیت استفاده کردیم، اما از آنجایی که افزودن جزئیات بیشتر به دیاگرام‌های پیچیده فاز تحلیل مستلزم وقت زیادی بود، در مدت انجام پروژه قادر نشدیم این کار را انجام دهیم؛ هر چند که افزودن این قدر دقت و بررسی صحت ترجمه خارج از اهداف این پروژه می‌باشد. در فصل بعد مدلسازی پروژه را در B خواهیم دید.