Analiza și proiectarea sistemelor software

curs 14

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

SISTEME EMBEDDED și RT/E

Utilizare sisteme de calcul pentru a controla o gamă largă de sisteme :

Ex. dispozitive casnice, Alte exemple ?

Sistem *embedded* – sistem hw/sw, programabil sau cu funcție fixă, proiectat pentru a îndeplini o anumită funcționalitate în cadrul unui sistem în care este inclus.

Software-ul acestor sisteme trebuie să *reacționeze la evenimentele* generate de hardware și, în general, să trimită *semnale de control* ca răspuns la aceste evenimente.

Software-ul acestor sisteme este inclus (*embedded*) în componentele hardware, de obicei în *memorii read-only*, și, în general, răspunde în timp real la evenimente din contextul sistemului.

Sisteme RT/E – Real Time / Embedded

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

REACTIVITATE

Reactivitatea în timp real este *diferența critică* dintre sistemele RT/E și alte sisteme software (sisteme de informații, sisteme bazate web, sisteme software personale).

Corectitudinea sistemelor software este definită prin modul în care *intrările* sistemului se proiectează pe *ieșirile corespunzătoare* ce trebuie produse de către sistem.

Corectitudinea sistemelor de timp real depinde în plus și de *timpul* de generare a răspunsului.

Sistemele de timp real trebuie să reacționeze la evenimentele din context *la viteze specifice contextului respectiv.*

Răspunsul trebuie să fie dat LA TIMP și să fie PREDICTIBIL.

În general există un *set de termene* ce trebuie respectate.

DEFINIȚII

Sistem de timp real = sistem software a cărui *funcționare corectă* depinde de *rezultatele* produse de sistem și de *timpul* în care acestea sunt produse.

Sistem de **timp real soft** = sistem de timp real a cărui *funcționare este degradată* dacă rezultatele nu sunt produse conform cerințelor de timp specificate.

Sistem de **timp real hard** = sistem de timp real a cărui *funcționare este incorectă* dacă rezultatele nu sunt produse conform cerințelor de timp specificate.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CARACTERISTICILE SISTEMELOR RT/E

- *Execuție continuă* (în general), pe toată durata funcționării dispozitivului fizic.

=> Cerințe : fiabilitate ridicată, configurare dinamică

- *Interacțiuni impredictibile* cu contextul

=> Cerințe : implementare cu procese paralele concurente

- Proiectarea sistemului poate fi afectată de *limitări fizic*e

=> Cerințe : conservare resursă de electricitate, preluare funcții hardware în software pentru a limita dimensiunea fizică (numărul de chip-uri)

- Poate fi necesară *interacțiunea directă cu hardware*

- Proiectarea sistemului poate fi dominată de cerințe de *siguranță și fiabilitate.*

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

PLAN CURS

Proiectarea sistemelor RT/E

Șabloane arhitecturale pentru aplicații RT/E

Analiza de timp

Sisteme de operare de timp real

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

PROIECTAREA SISTEMELOR RT/E

Procesul de proiectare a sistemelor embedded este un proces de inginerie a sistemelor ce trebuie să ia în considerare, în detaliu, *construcția și performanța sistemului hardware.*

În procesul de proiectare se va *decide* care capabilități ale sistemului vor fi implementate în *software* și care în *hardware.*

În *fazele inițiale* ale procesului de proiectare se vor lua deciziile referitoare la *hardware,* la *software suport* și la *restricții de timp*.

În software pot fi incluse *funcții suplimentare specifice* sistemelor embedded (ex. Managementul bateriei și al sursei de alimentare).

Modelul stimul-răspuns = abordarea tipică de proiectare

Stimul = eveniment ce apare în contextul sistemului software și care declanșează o reacție de la software.

Răspuns = semnal sau mesaj trimis de software către contextul său.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

STIMULI ȘI RĂSPUNSURI
Exemplu: Sistem de alarmă anti-furt

STIMUL	RĂSPUNS
Ștergere alarmă	Închiderea tuturor alarmelor active. Închiderea tuturor semnalelor luminoase activate de alarmă.
Activare buton de panică de pe console	Inițiere alarmă. Aprinderea tuturor luminilor de pe consolă. Apel la poliție.
Căderea tensiunii de alimentare	Apel la tehnicianul de service.
Defectarea unui senzor	Apel la tehnicianul de service.
Activarea unui senzor	Inițiere alarmă. Aprinderea luminilor din zona senzorului.
Activarea a doi sau mai multor senzori	Inițiere alarmă. Aprinderea luminilor din zonele senzorilor. Apel la poliție indicând localizarea spargerii suspectată.
Cădere tensiune între 10% și 20%	Comutare pe baterie; testare sursă de alimentare
Cădere tensiune peste 20%	Comutare pe baterie; inițiere alarmă; apel la poliție; testare sursă de alimentare

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

TIPURI DE STIMULI

Stimulii provin de la *senzorii* din contextul sistemului și de la *elementele de acționare (actuator)* controlate de sistem.

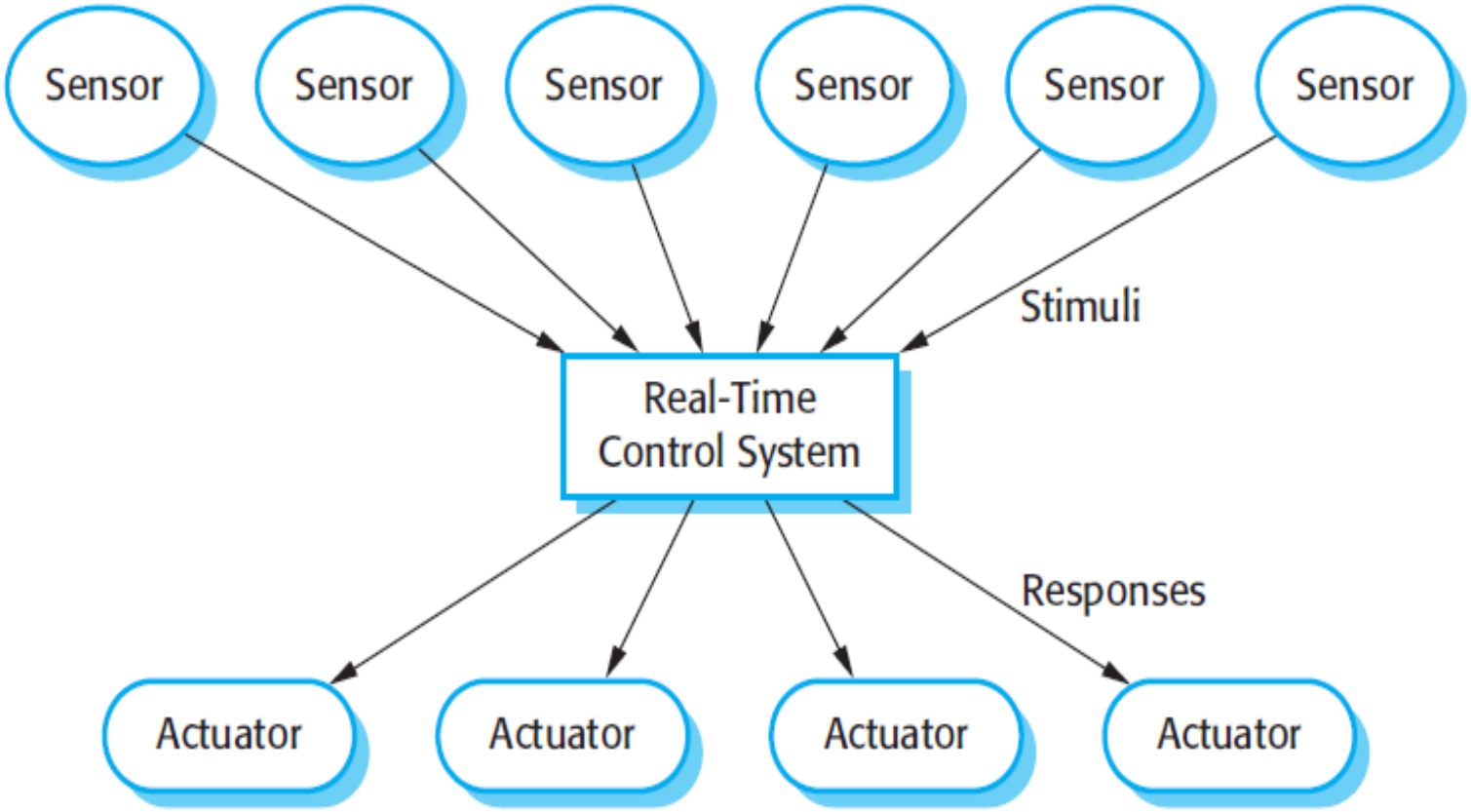
- Stimul ***periodic*** – apare la intervale de timp predictibile

Ex. examinare senzor la 50 msec și acțiune funcție de valoarea preluată.

- Stimul ***aperiodic*** – apare la intervale neregulate sau nepredictibile și este semnalat, în general, prin sistemul de întreruperi al procesorului

Ex. Întrerupere ce indică terminarea unui transfer de I/E și disponibilitatea datelor într-un buffer.

MODEL GENERAL AL UNUI SISTEM RT/E



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CONSIDERAȚII ARHITECTURALE

Cerință : răspuns conform cerințelor de timp specifice diferiților senzori.

Soluție : arhitectura sistemului trebuie să permită comutarea rapidă între rutinele de tratare (*handler*) a stimulilor.

Senzori multipli; cerințele de timp pentru diferiți senzori sunt diferite => o buclă secvențială simplă nu este în general utilă.

⇓

Sistemele de timp real sunt proiectate ca *procese concurente și cooperante* controlate de un *executiv de timp real* (real-time executive).

TIPURI FUNDAMENTALE DE PROCESE PENTRU SISTEM RT/E

Proces de *control senzor*

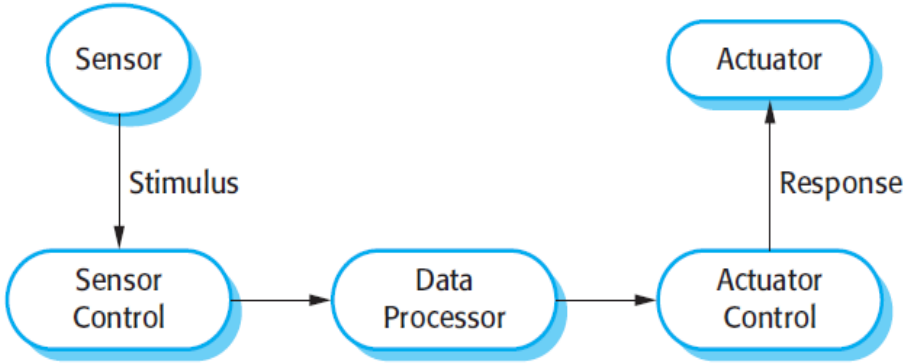
- Colectează informațiile de la senzori. Pot buffer-a informația colectată.

Proces *procesare date*

- Procesează informația colectată și calculează răspunsul sistemului.

Proces de *control element de acționare*

- Generează semnalele de control pentru elementele de acționare.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ACTIVITĂȚILE PROCESULUI DE PROIECTARE
PENTRU SISTEME RT/E

- ***Selectarea platformei*** – hardware și sistem de operare de timp real (RTOS).
- ***Identificarea perechilor stimul / răspuns***
- ***Analiza cerințelor de timp*** – restricțiile de timp pentru procesarea fiecărui stimul și răspunsului asociat ⇒ termene (*deadline*) pentru procesele din sistem.
- ***Proiectarea proceselor*** – definire set de procese concurente pentru procesare stimuli și răspunsuri.
- ***Proiectarea algoritmilor*** – prelucrare optimizată a stimulilor și răspunsurilor corespunzătoare.
- ***Proiectarea datelor*** – definirea informațiilor comunicate/partajate între procese, a evenimentelor ce coordonează schimbul de informații și proiectarea structurilor de date coresunzătoare.
- ***Planificarea proceselor*** *(scheduling)* – pornirea proceselor a.î. să răspundă la timp.
- ***Verificarea modelului*** – analiză statică bazată pe cunoștințe despre comportarea în timp a componentelor sau pe simulări.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ACTIVITĂȚILE PROCESULUI DE PROIECTARE PENTRU SISTEME RT/E

- Selectarea platformei
- Identificarea perechilor stimul / răspuns
- Analiza cerințelor de timp
- Proiectarea proceselor
- Proiectarea algoritmilor
- Proiectarea datelor
- Planificarea proceselor (scheduling)

Ordinea activităților în procesul de proiectare depinde de

- tipul sistemului
- cerințele pentru platformă

Variante :

1. Proiectare stimuli și răspunsuri - ulterior selectare platformă
2. Alegere platformă – proiectare stimuli și răspunsuri în cadrul unor restricții impuse de platformă.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

EXCLUDEREA MUTUALĂ

Procese concurente și cooperante ⇒ partajare date și coordonare.

Mecanismul de coordonare = excludere mutuală a accesului la resurse partajate.

Mecanisme tipice de asigurare a excluderii mutuale : **?**

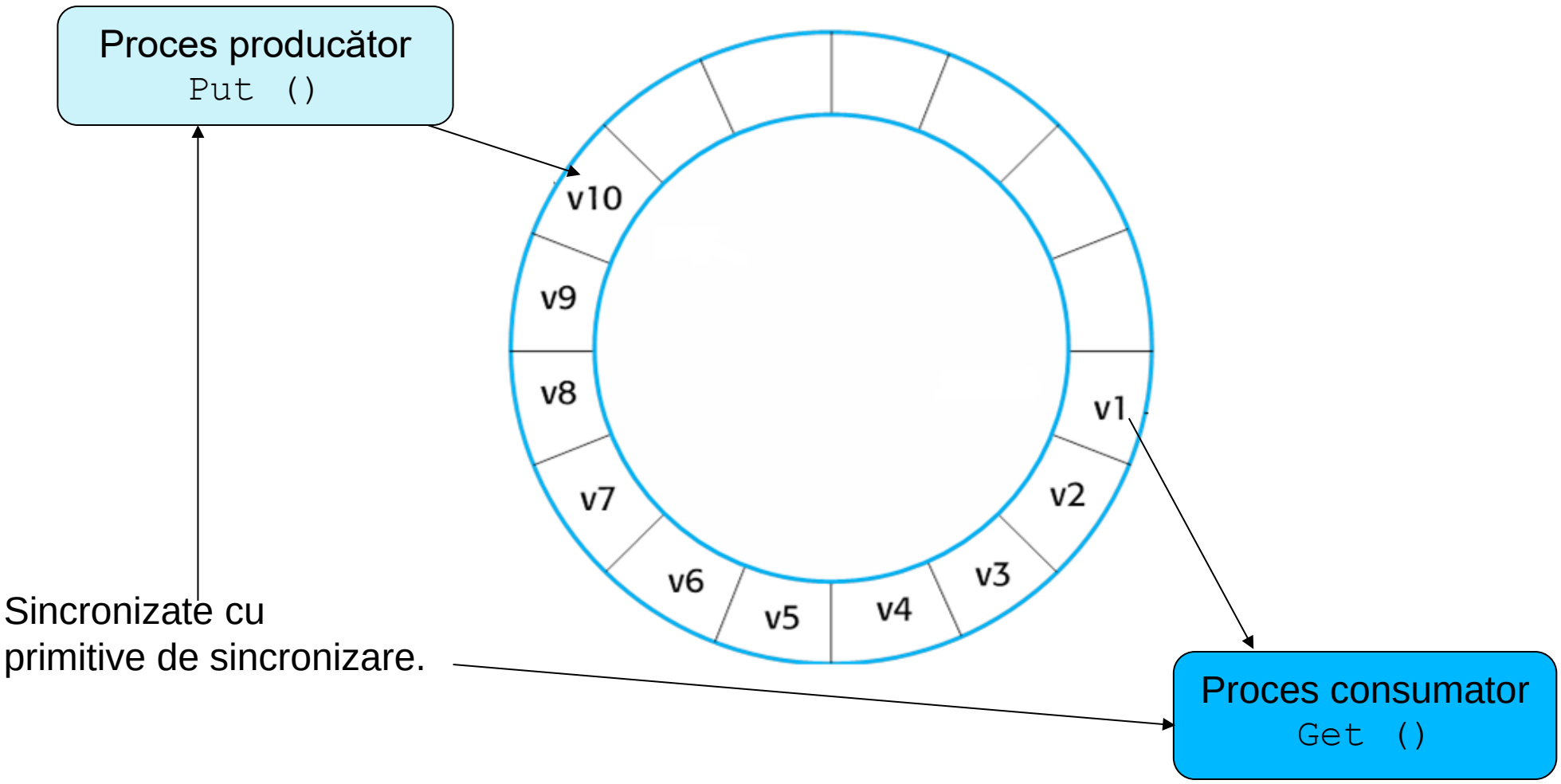
semafor, monitor, per secțiune critică.

Modelul clasic :

procese producător – consumator, viteze de execuție diferite, date în buffer

- Procesele producător colectează date și le adaugă într-un buffer. Procesele consumator preiau date din buffer și le prelucrează.
- Procesele producător și consumator trebuie să fie mutual exclusive referitor la accesarea acelorași date.
- Buffer-ul trebuie să oprească procesele producător dacă este plin și procesele consumator dacă este gol.

PROCESE PRODUCĂTOR/CONSUMATOR PARTAJÂND UN BUFFER CIRCULAR



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

DECIZII ALOCARE RESPONSABILITĂȚI
ÎN SISTEME RT/E

Context : restricții de timp

Soluție :

- Componente hardware :
  - Performanță mai bună decât echivalentele software
  - FPGA (field-programmable gate array) – circuite flexibile, adaptabile la diferite funcții
- Implementare anumite funcții sistem în hardware (ex. procesare semnale)
- Identificare gâtuiri de procesare și preluare funcție în hardware

Obs. De remarcat prioritatea restricțiilor de timp față de limitările fizice.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR RT/E
DIAGRAMA DE STĂRI ȘI TRANZIȚII

Sistemele de timp real sunt caracterizate de *stări multiple*.

Efectul unui *stimul* poate declanșa o *tranziție de la o stare la alta*.

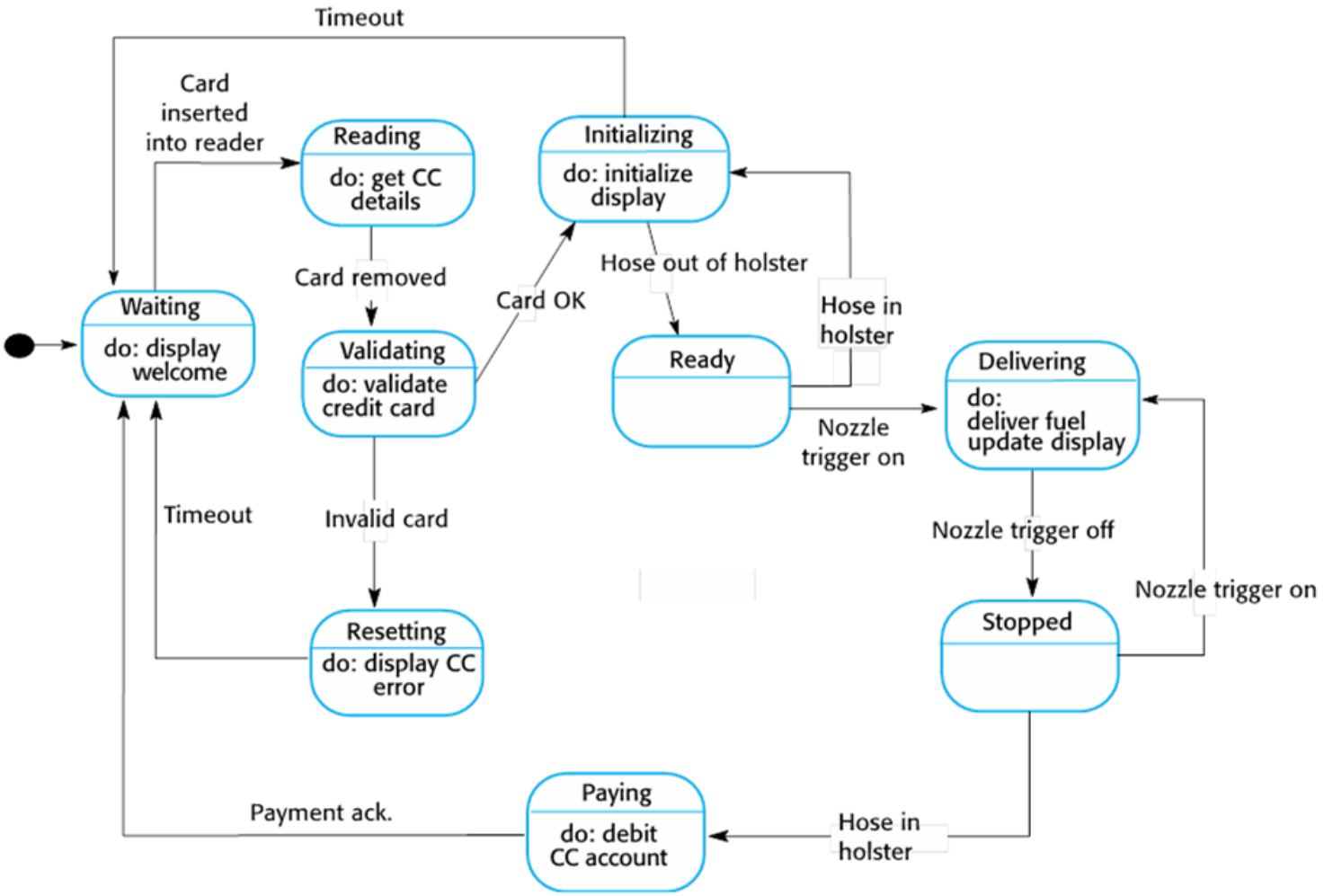
$\Downarrow$

Se utilizează modele de stări și tranziții pentru sistemele RT/E.

UML – diagrame de stări și tranziții (state machine diagram).

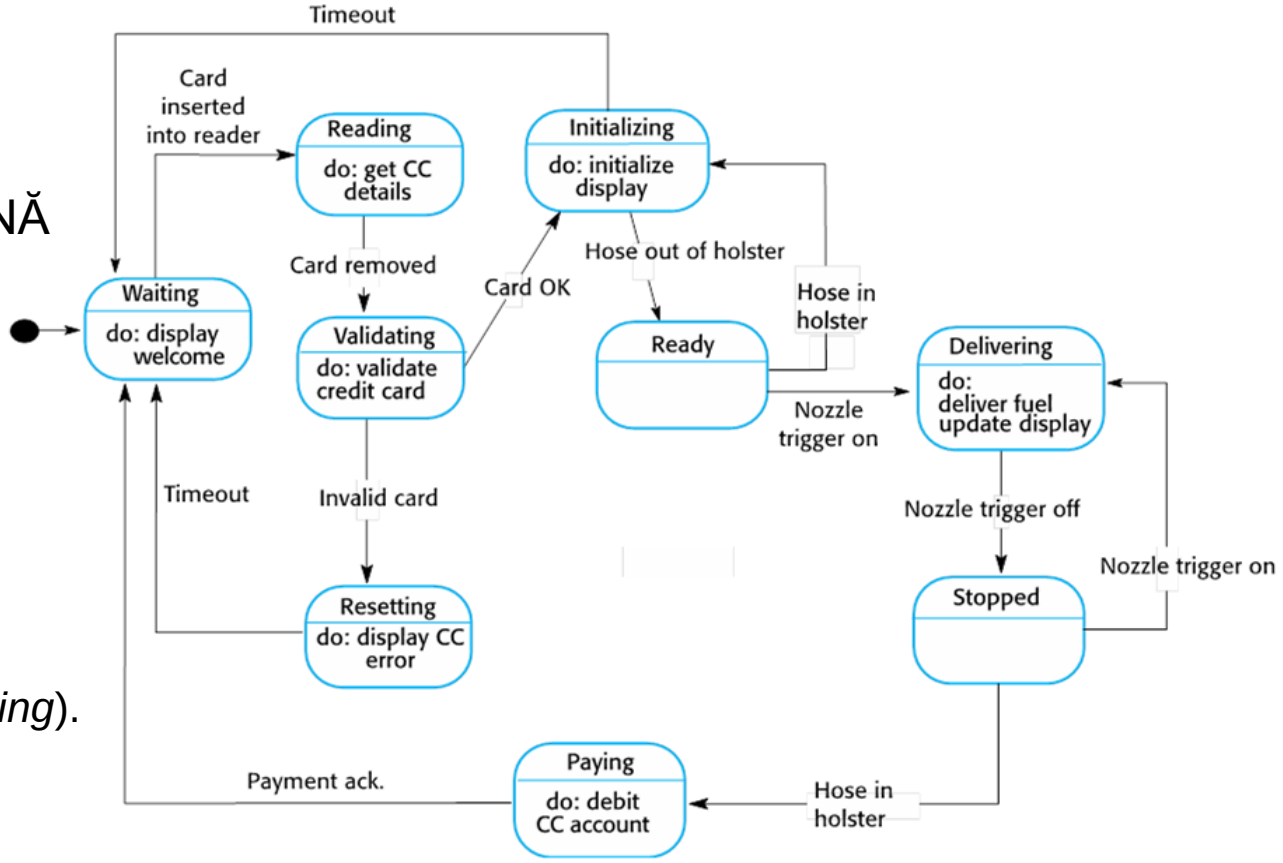
Obs. Pot fi transformate automat în cod executabil $\Rightarrow$ utilizate în MDD.

Exemplu : DIAGRAMA DE STĂRI ȘI TRANZIȚII PENTRU O POMPĂ DE BENZINĂ



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

Exemplu :
SECVENȚA OPERAȚIILOR
ÎN SISTEMUL DE CONTROL ÎN
TIMP REAL AL POMPEI DE BENZINĂ



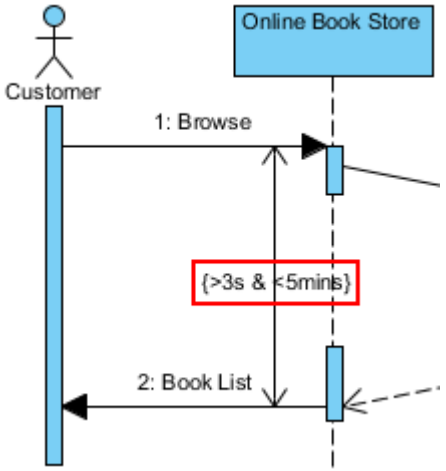
1. Sistemul este în starea inițială de așteptare a unei comenzi (starea *Waiting*).
2. *Cumpărătorul inserează un card de* credit în cititorul de la pompă.
3. Extragerea cardului declanșează tranziția în starea *Validating*, stare în care se validează cardul.
4. Dacă e valid, sistemul inițializează pompa (starea *Initializing*).
5. Preluarea furtunului din toc urmată de activarea acestuia declanșează tranziția în starea *Delivering.*
6. La încheierea alimentării furtunul este pus înapoi în toc, ceea ce declanșează tranziția în starea *Paying.*
7. După realizarea plății sistemul revine în starea *Waiting.*

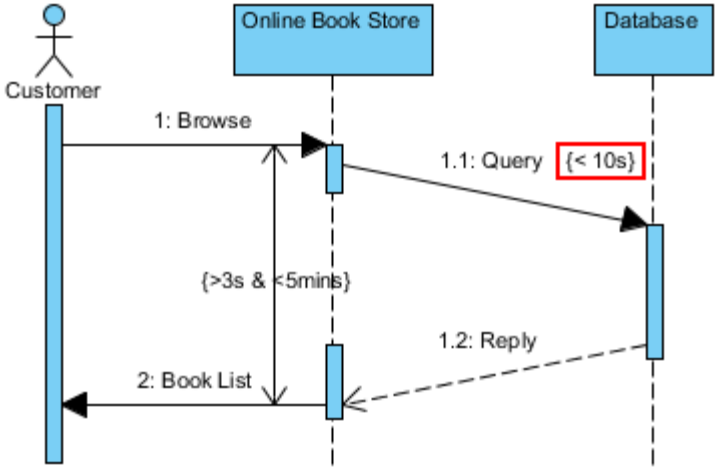
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E
DIAGRAMA DE SECVENȚE CU RESTRICȚII DE TIMP

Modelare interacțiuni între obiecte, cu restricții de timp

- între mesaje
- pentru un mesaj





https://www.visual-paradigm.com/tutorials/how-to-use-duration-constraints-in-sequence-diagram.jsp

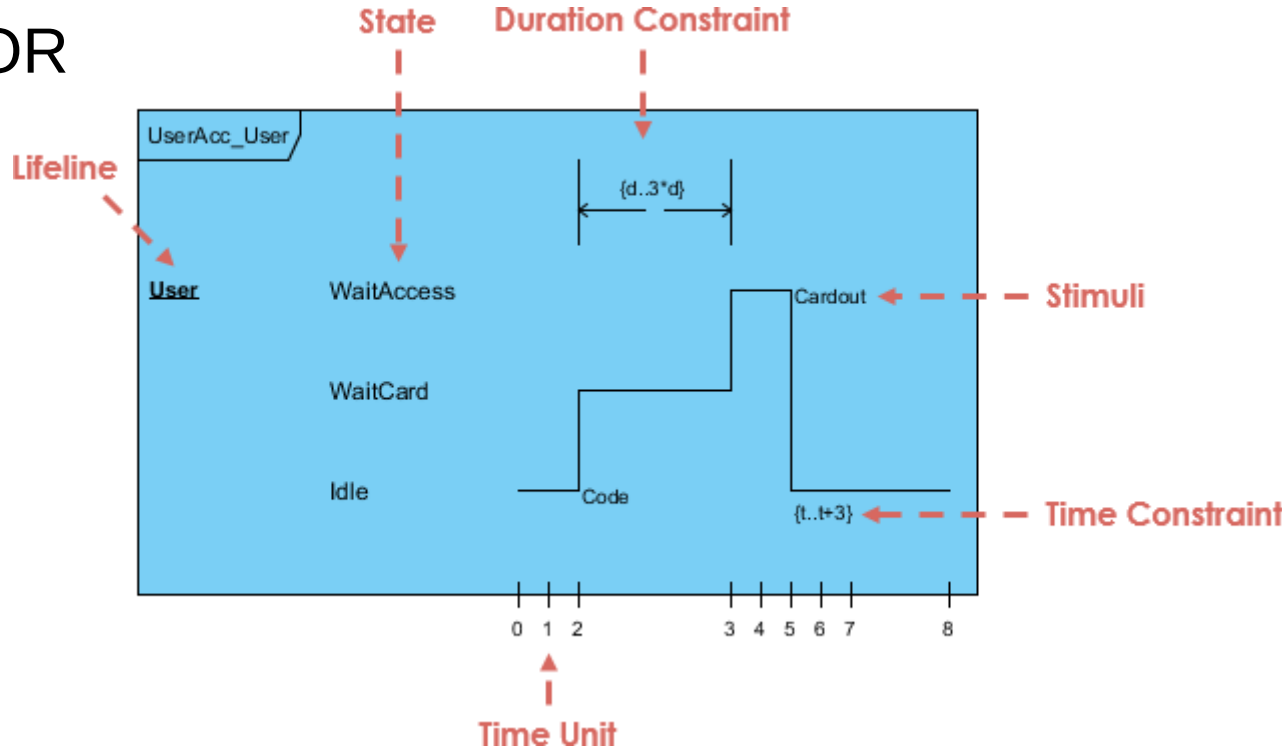
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E

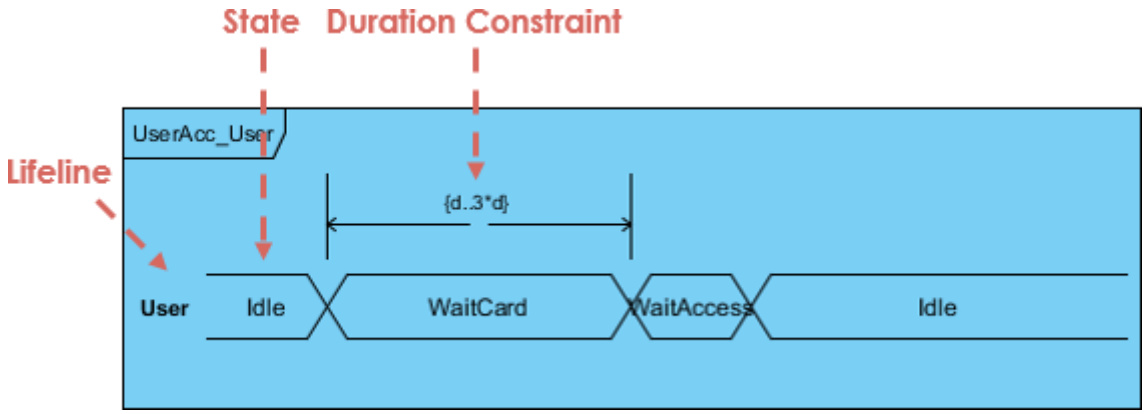
DIAGRAMA DE TIMP

Variante de reprezentare :

stări = nivele pe lifeline



stări = valori pe lifeline



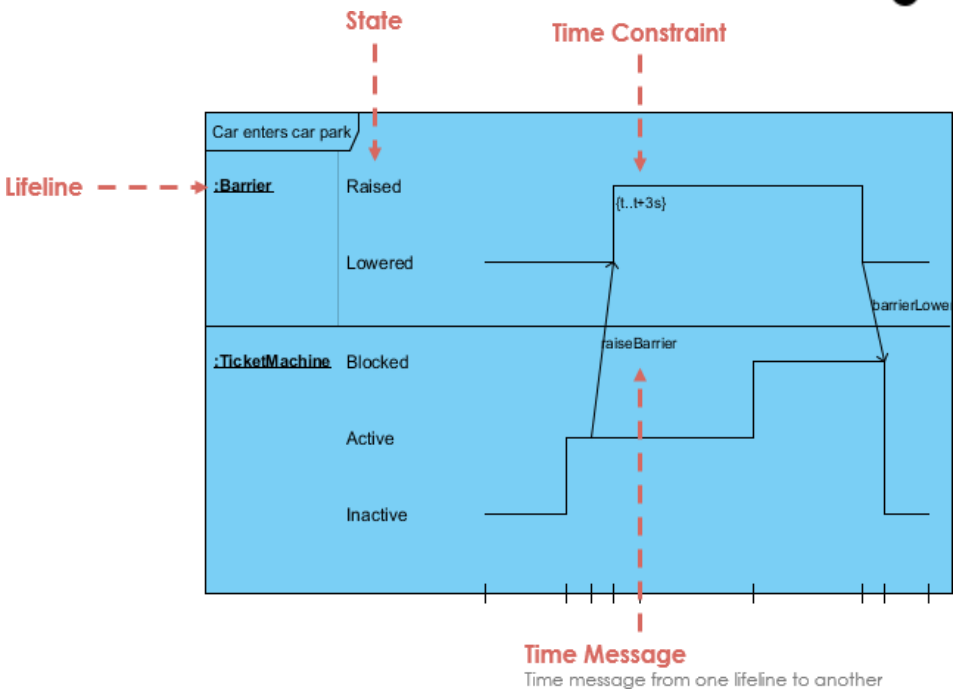
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/

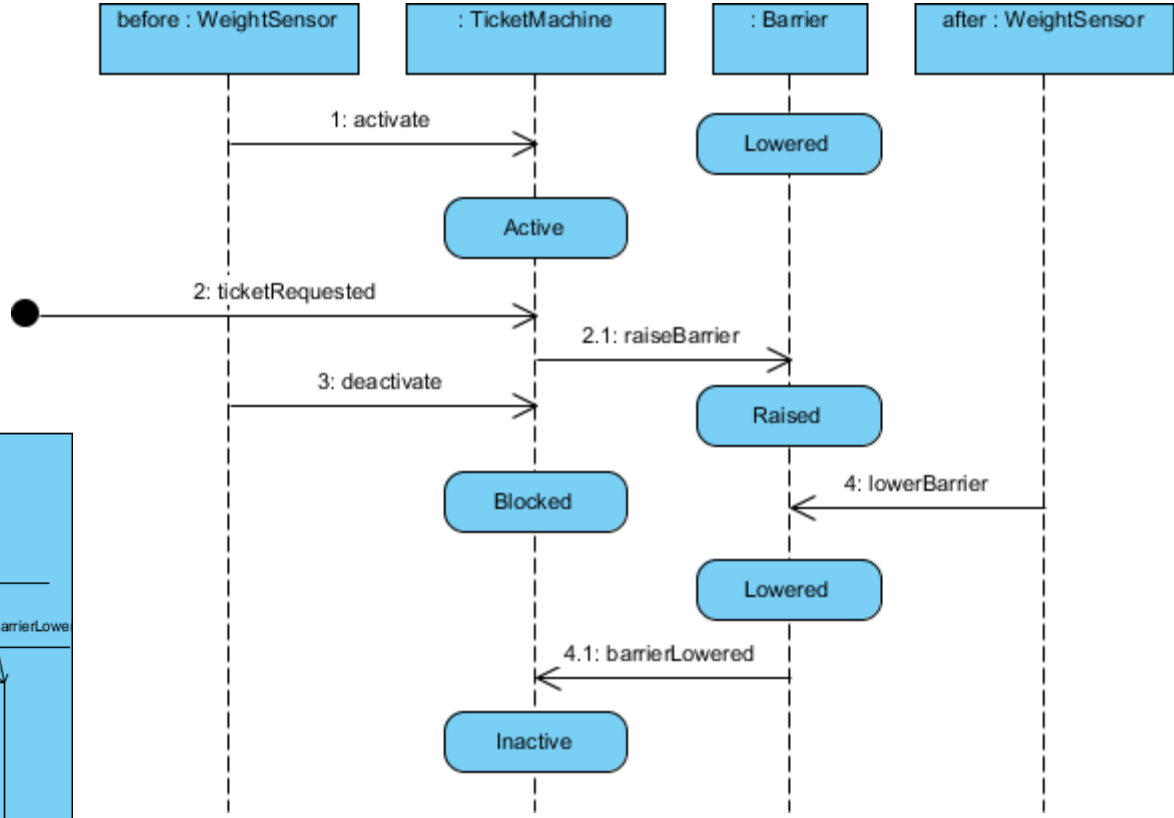
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E
COMBINARE DIAGRAME

Specificare stări ale obiectelor
în diagrama de secvențe.

Diagrama de timp corespunzătoare.





https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E
PROFIL UML

UML for MARTE Profile (**M**odeling and **A**nalysis of **R**eal-**T**ime **E**mbedded Systems)

set de extensii pentru :

1. Exprimare directă a *conceptelo*r specifice domeniului RT:
   - Proprietăți non-funcționale
   - Resurse
   - Mecanisme de concurență
   - Mecanisme de timp
2. Exprimarea de *aspecte cantitative*, utile pentru *analiza modelelor*.

Nivele de modelare :

- Descriere PIM (independent de platforma hardware) - instalare aplicație pe platformă RTOS.
- Descriere PSM - instalare PIM pe platformă hardware.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E
PROFIL UML

- Modelare *independentă* a elementelor *hardware* și *software* și a *relațiilor* dintre acestea.
- *Construcțiile de modelare* ale procesului de dezvoltare a sistemelor RT/E: *high-level* - specificare și *low-level* - implementare :
  - Calitative : paralelism, sincronizare, comunicare
  - Cantitative : termen, periodicitate

Permite *alegerea stilului* și ***construcțiilor de modelare*** preferate (incluzând modele din UML standard).

- Permite utilizarea diferitelor tipuri de *tehnici de analiză a modelelor* fără a le cunoaște detaliile.
- Utilizare modele :
  - ***predicții*** cantitative și de partiționare
  - ***analize*** referitoare la caracteristicile hardware și software ale sistemului RT/E.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

MODELAREA SISTEMELOR SOFTWARE RT/E
AADL

AADL (**A**rchitecture and **A**nalysis **D**esign **L**anguage) – limbaj de modelare pentru sisteme RT/E.

Modelele produse sunt utilizate de *instrumente software* pentru :

- *analiza specificației* sistemului
- *verificarea proprietăților* funcționale și extra-funcționale ale sistemului
  - consum energie
  - analiza de timp
  - ...
- *generare cod* pentru platforma hardware țintă.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

PROGRAMAREA PENTRU TIMP REAL

- Limbajele de programare pentru dezvoltarea sistemelor de timp real trebuie să :
    - includă facilități de *acces la componentele hardware*
    - ofere posibilitatea de a *cunoaște timpul de execuție* a unor operații particulare
- Limbaje preferate: limbaj de asamblare sau limbaj la nivel de sistem care permite generare de cod eficient (ex. C)
- Sincronizarea proceselor – *apel sistem* la construcții (ex. semafor) oferite de sistemul de operare de timp real.
- Limbajele orientate obiect introduc *întârzieri* din cauza codului suplimentar necesar medierii *accesului la atribute* și *manipulării apelurilor de operații.* Aceasta poate conduce la imposibilitatea îndeplinirii termenelor de timp real impuse (în special pentru sistemele de timp real hard).

Obs. Există o specificație Java pentru dezvoltarea sistemelor embedded.
https://jcp.org/en/jsr/detail?id=282

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

Evaluare formativă

1. Tipurile fundamentale de procese ale unui sistem RT/E.
2. Enumerați activitățile de proiectare pentru sistemele RT/E.
3. Realizați corespondența corectă între model și diagramele UML folosite.
4. Ce diagrame UML se folosesc în mod special pentru sistemele RT/E ?

https://forms.gle/KFEHCE4YkvRLf4tu6

PLAN CURS

Proiectarea sistemelor embedded

Șabloane arhitecturale pentru aplicații RT/E

Analiza de timp

Sisteme de operare de timp real

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ȘABLOANE ARHITECTURALE PENTRU APLICAȚII RT/E

ARHITECTURI CARACTERISTICE SISTEMELOR RT/E

Specific : orientate pe proces.

- ***Observe and React* –** un set de senzori sunt monitorizați și afișați.

- ***Environmental Control* –** sistemul include :
  - senzori care oferă informații despre context,
  - elemente de acționare care modifică contextul.

- ***Process Pipeline* –** transformare rapidă de date dintr-o reprezentare în alta aplicând procesări succesive într-o conductă de elemente de procesare executate în paralel.

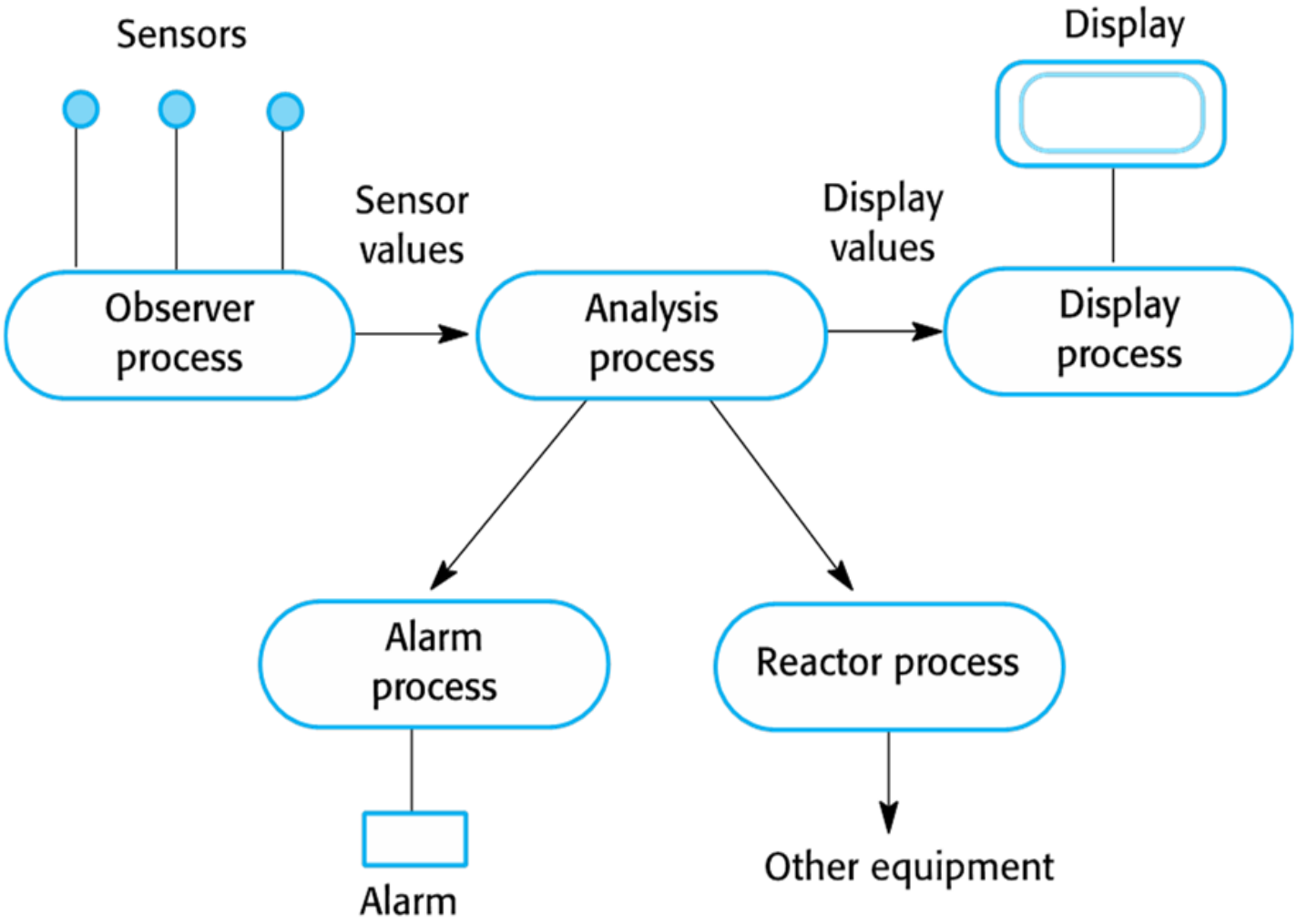
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ȘABLONUL Observe and React

NUME	OBSERVE and REACT
Descriere	Valorile de intrare de la un set de *senzori* de același tip sunt *colectate* și analizate. Aceste valori sunt ***afișate***. Dacă valorile senzorilor indică apariția unor *condiții excepționale*, atunci sunt inițiate acțiuni prin care se atrage atenția operatorului care, în anumite cazuri, va realiza acțiuni specifice.
Stimuli	Valori preluate de la senzorii atașați la sistem.
Răspunsuri	Ieșirile de afișat. Declanșatoare de alarmă. Semnale la sisteme de reacție.
Procese	Observer, Analysis, Display, Alarm, Reactor
Utilizare	Sisteme de monitorizare. Sisteme de alarmă

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

SABLONUL Observe and React
STRUCTURA DE PROCESE



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

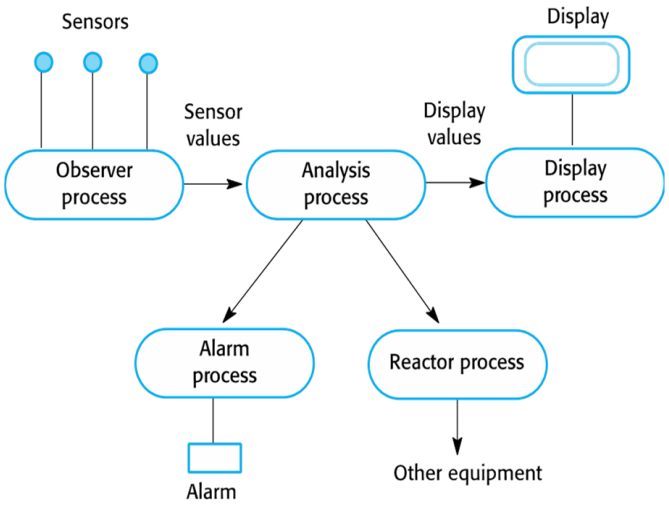
Exemplu : SISTEM DE ALARMĂ ANTI-FURT

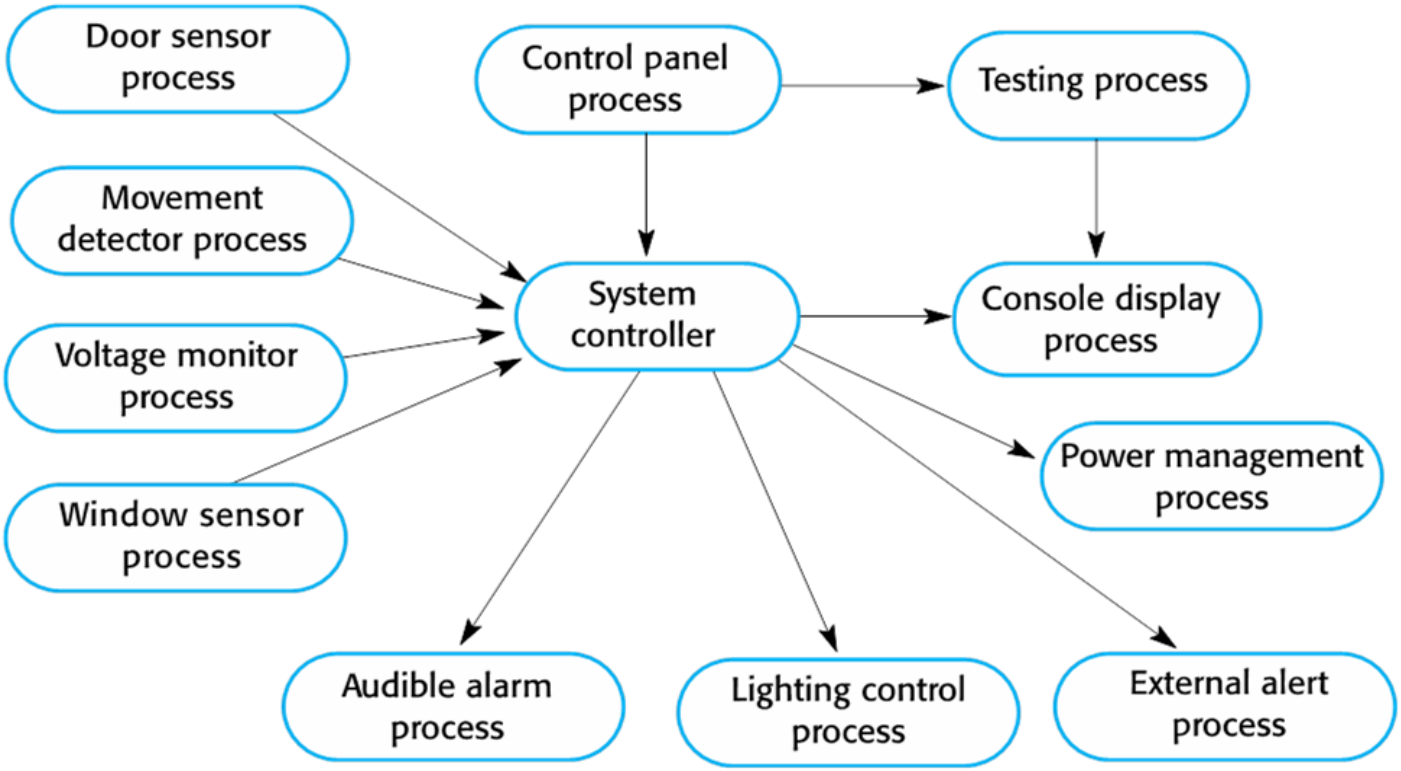
Trebuie implementat un sistem de alarmă ca parte a unui sistem anti-furt pentru clădiri comerciale. Acesta utilizează diferite tipuri de senzori : detectori de mișcare în camere, detectori de sesizare a deschiderii ușilor către coridor, detectori de deschidere a ferestrelor de la parter.

Când un senzor detectează prezența unui intrus, sistemul apelează automat poliția locală și, folosind un sintetizator de voce, raportează localizarea alarmei. Aprinde luminile în camerele din jurul senzorului activat și pornește o alarmă sonoră.

Sistemul de senzori este alimentat de la rețeaua generală dar este echipat și cu o baterie de rezervă. Pierderea de tensiune este detectată folosind un circuit separat care monitorizează alimentarea generală. Dacă este detectată o scădere de tensiune, sistemul presupune că intrusul a întrerupt alimentarea cu energie electrică și pornește o alarmă.

Exemplu :
STRUCTURA PROCESELOR DIN SISTEMUL DE ALARMĂ





Implementare sisteme cu mai multe instanțieri ale șablonului *Observe and React,* câte una pentru fiecare tip de senzor.
Optimizări : Unificare procese de analiză și procese de afișare.

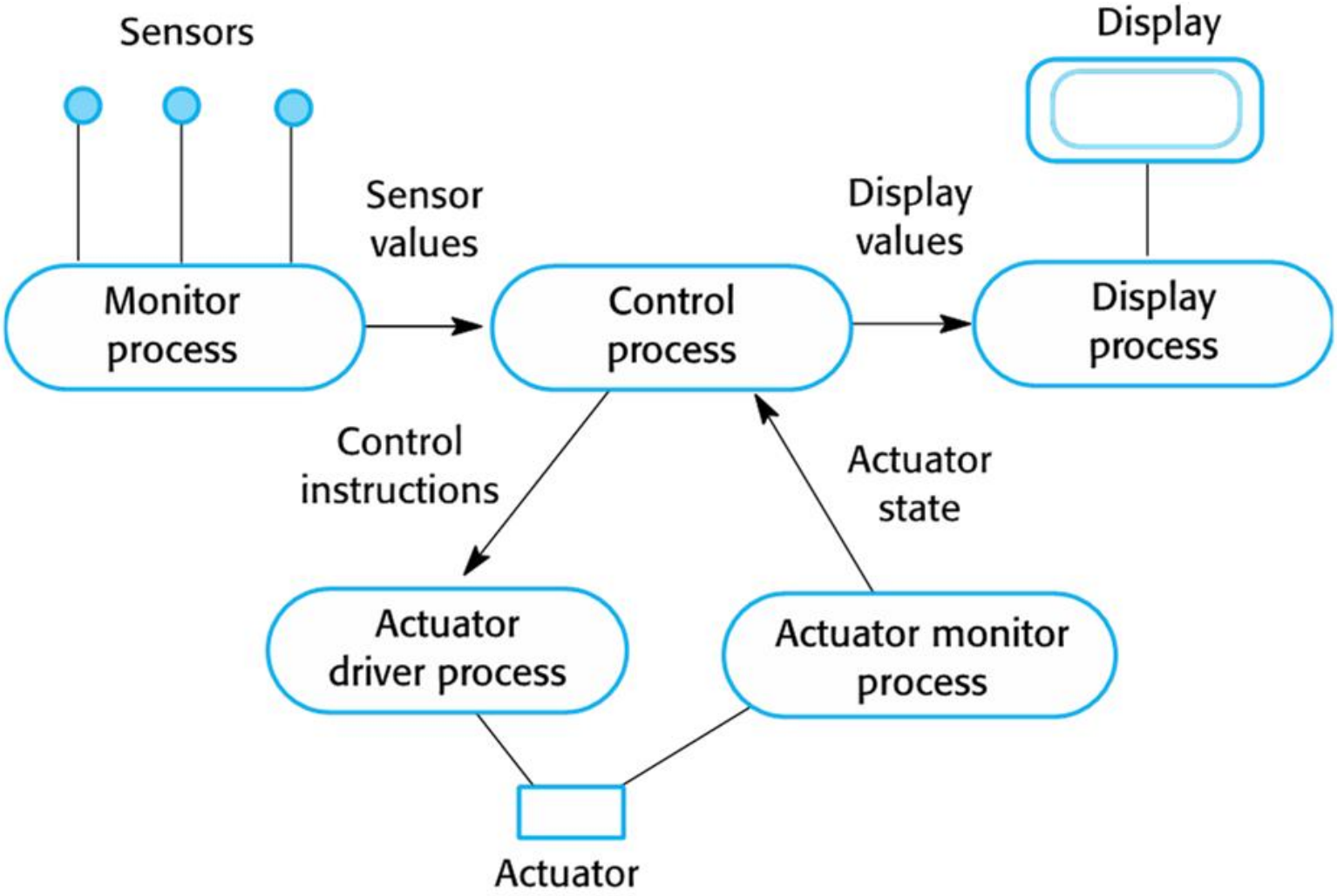
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ȘABLONUL Environmental Control

NUME	ENVIRONMENTAL CONTROL
Descriere	Sistemul analizează informații de la un set de *senzori* care *colectează date* din contextul sistemului. De asemenea, pot fi colectate informații de *stare* de la *elementele de acționare* conectate la sistem. Pe baza acestor date se trimit *semnale de control* către elementele de acționare care vor provoca modificări în contextul sistemului. Aceste informații pot fi, de asemenea, *afișate*.
Stimuli	Valori preluate de la senzorii atașați la sistem. Informații de stare preluate de la elementele de acționare conectate la sistem.
Răspunsuri	Semnale de control către elementele de acționare. Afișare informații.
Procese	Monitor, Control, Display, Actuator driver, Actuator monitor
Utilizare	Sisteme de control

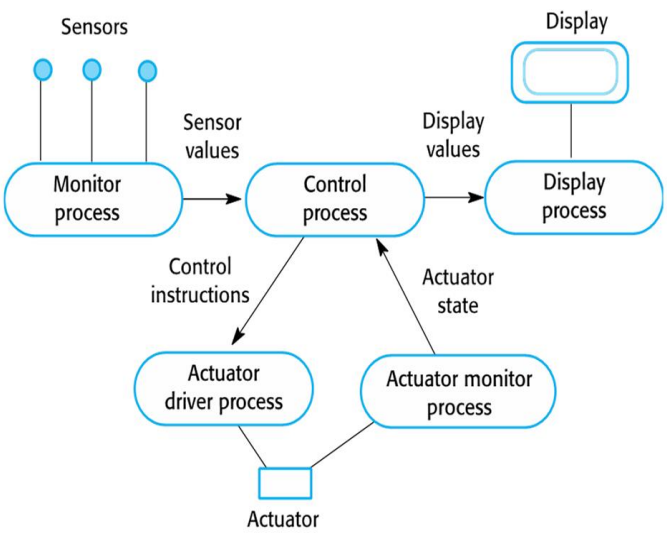
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

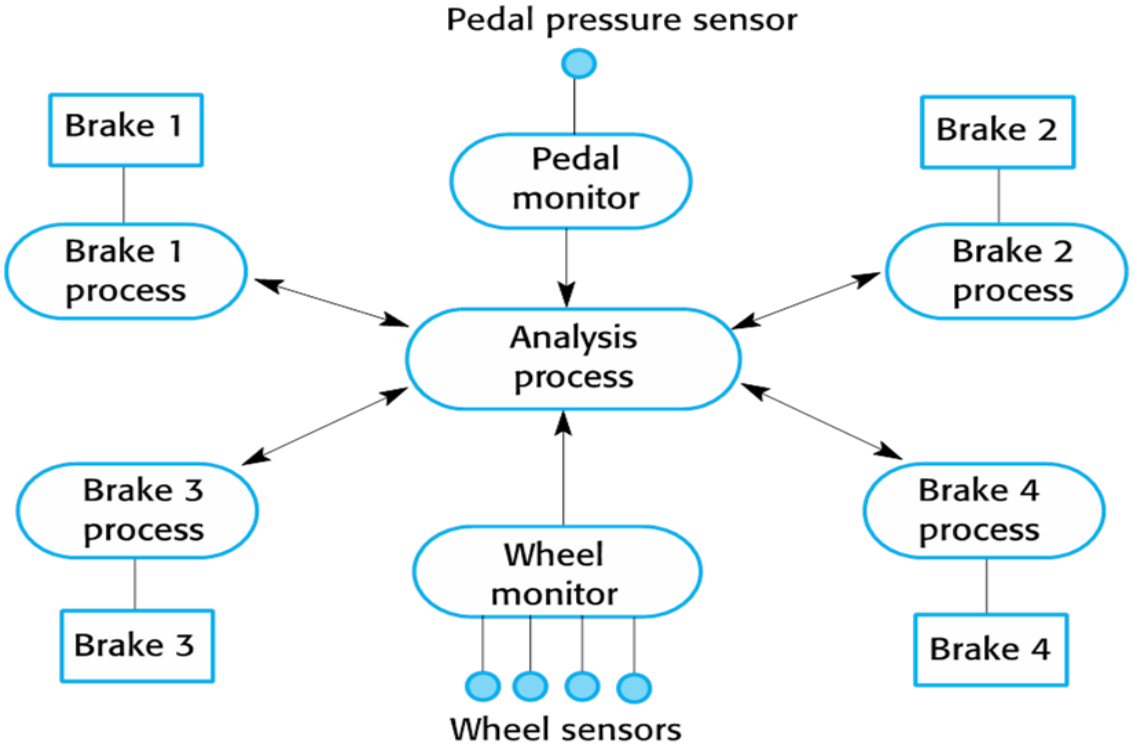
ȘABLOANUL Environmental Control
STRUCTURA DE PROCESE



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

Exemplu
ARHITECTURA SISTEMULUI DE CONTROL ANTI-DERAPARE





Implementatare sisteme cu câte o instanțiere a șablonului *Environmental Control* pentru fiecare tip de element de acționare.
Optimizări : variante de combinare procese de monitorizare și de control.

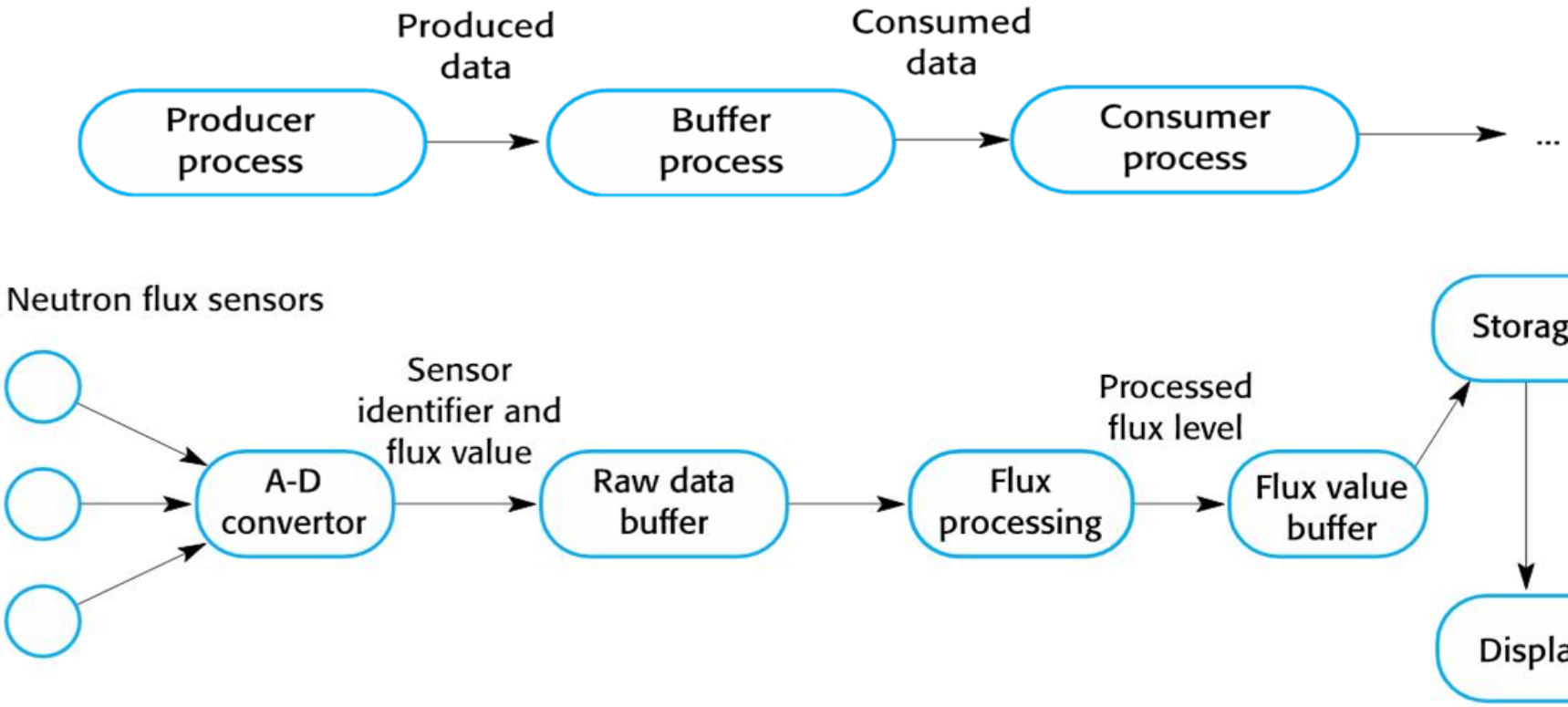
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ȘABLOANUL Process Pipeline

NUME	PROCESS PIPELINE
Descriere	Se configurează o conductă de procese cu transferul datelor în secvență de la un capăt la altul al conductei. Procesele sunt de obicei legate prin buffer-e sincronizate pentru a permite proceselor producător și consumator să ruleze la viteze diferite. Conducta se poate termina cu afișarea datelor, memorarea datelor, activarea unui element de acționare.
Stimuli	Valori de intrare preluate din context sau de la alt proces
Răspunsuri	Valori de ieșire către context sau către un buffer partajat.
Procese	Producer, Buffer, Consumer
Utilizare	Sisteme de achiziție date. Sisteme multimedia.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ȘABLOANUL Process Pipeline
STRUCTURA DE PROCESE
Exemplu : Colectare date despre densitatea de neutroni



Exemplu de colectare în timp real, datele fiind memorate pentru aplicare de analize ulterioare.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

SABLOANE ARHITECTURALE
pentru sisteme embedded

Șabloane arhitecturale = elemente de pornire în descrierea structurii de ansamblu a unui sistem (embedded).

Pot fi combinate.

Exemplu : *Environmental Control* combinat cu *Observe and React* pentru elementele de acționare din sistem.

În sistemele finale se folosesc *șabloane low-level* orientate pe controlul execuțiilor, pe comunicare, pe alocarea resurselor, pe siguranță și pe fiabilitate.

Exemple:

http://www.uml.org.cn/UMLApplication/pdf/rtpatterns.pdf

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

PLAN CURS

Proiectarea sistemelor embedded

Șabloane arhitecturale pentru aplicații RT/E

Analiza de timp

Sisteme de operare de timp real

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ANALIZA DE TIMP

- Corectitudinea unui sistem de timp real depinde atât de *corectitudinea ieșirilor* cât și de *momentul* la care aceste ieșiri sunt produse.

- În analiza de timp se calculează cât de des trebuie executat fiecare proces *(frecvența)* din sistem astfel încât toate intrările să fie procesate și toate ieșirile să fie produse *la timp*.

⇓

- Rezultatele analizei de timp sunt folosite pentru a decide *frecvența de execuție* a fiecărui proces și cum trebuie *planificate aceste procese* de către sistemul de operare de timp real.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

FACTORII ANALIZEI DE TIMP

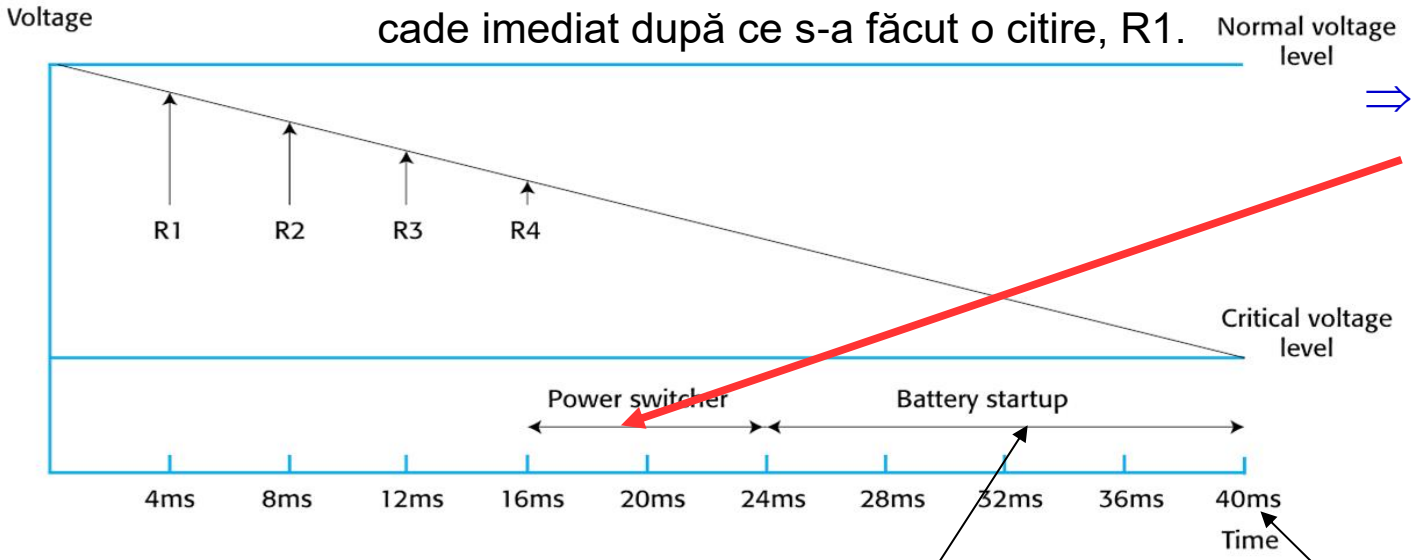
- Termen (deadline) la nivel de sistem
  - Termenele de finalizare a procesării stimulilor și producerii răspunsului de către sistem.
- Frecvență la nivel de proces
  - Numărul de execuții în unitatea de timp ale unui proces astfel încât întotdeauna termenele să fie îndeplinite.
- Timp de execuție la nivel de proces
  - Timpul necesar procesării unui stimul și producerii rezultatului corespunzător.
    - Sisteme de timp real hard – cazul cel mai defavorabil
    - Sisteme de timp real soft – timpul mediu.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ANALIZA DE TIMP
Exemplu : Căderea tensiunii de alimentare

Problemă de alimentare = scădere semnificativă de tensiune între citiri menținută pe parcursul a 3 citiri.

Scenariul cel mai defavorabil :Tensiunea cade imediat după ce s-a făcut o citire, R1.



⇒ Procesul de pornire a bateriei trebuie să dureze max. 8 msec.

Proces de verificare lansat de 250 ori pe secundă (la 4 msec.).

16 msec de la pornirea bateriei de rezervă până la funcționarea sa completă.

Sunt necesare 50 msec pentru ca tensiunea de alimentare să cadă la un nivel la care echipamentul este avariat => Bateria de rezervă trebuie activată și pusă în stare de funcționare în max. 40 msec. (rezervă pentru variațiile fizice din echipament).

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ANALIZA DE TIMP
Exemplu : Căderea tensiunii de alimentare

- Sunt necesare 50 msec pentru ca tensiunea de alimentare să cadă la un nivel la care echipamentul este avariat => Bateria de rezervă trebuie activată și pusă în stare de funcționare în max. 40 msec. (rezervă pentru variațiile fizice din echipament).
- Sunt necesare 16 msec de la pornirea bateriei de rezervă până la funcționarea sa completă.
- Există un proces de verificare lansat de 250 ori pe secundă (la 4 msec.). Acesta presupune că există o problemă de alimentare dacă are loc o scădere semnificativă de tensiune între citiri menținută pe parcursul a 3 citiri.
- Scenariul cel mai defavorabil :
    - Tensiunea cade imediat după ce s-a făcut o citire, R1. Tensiunea continuă să scadă la citirile R2-R4 astfel încât se detectează o posibilă cădere de tensiune după 16msec.
    - După lansarea procesului de comutare pe bateria de rezervă, bateria necesită 16msec pentru ajunge în stare operațională.

Concluzie : procesul de pornire a bateriei trebuie să dureze max. 8 msec.

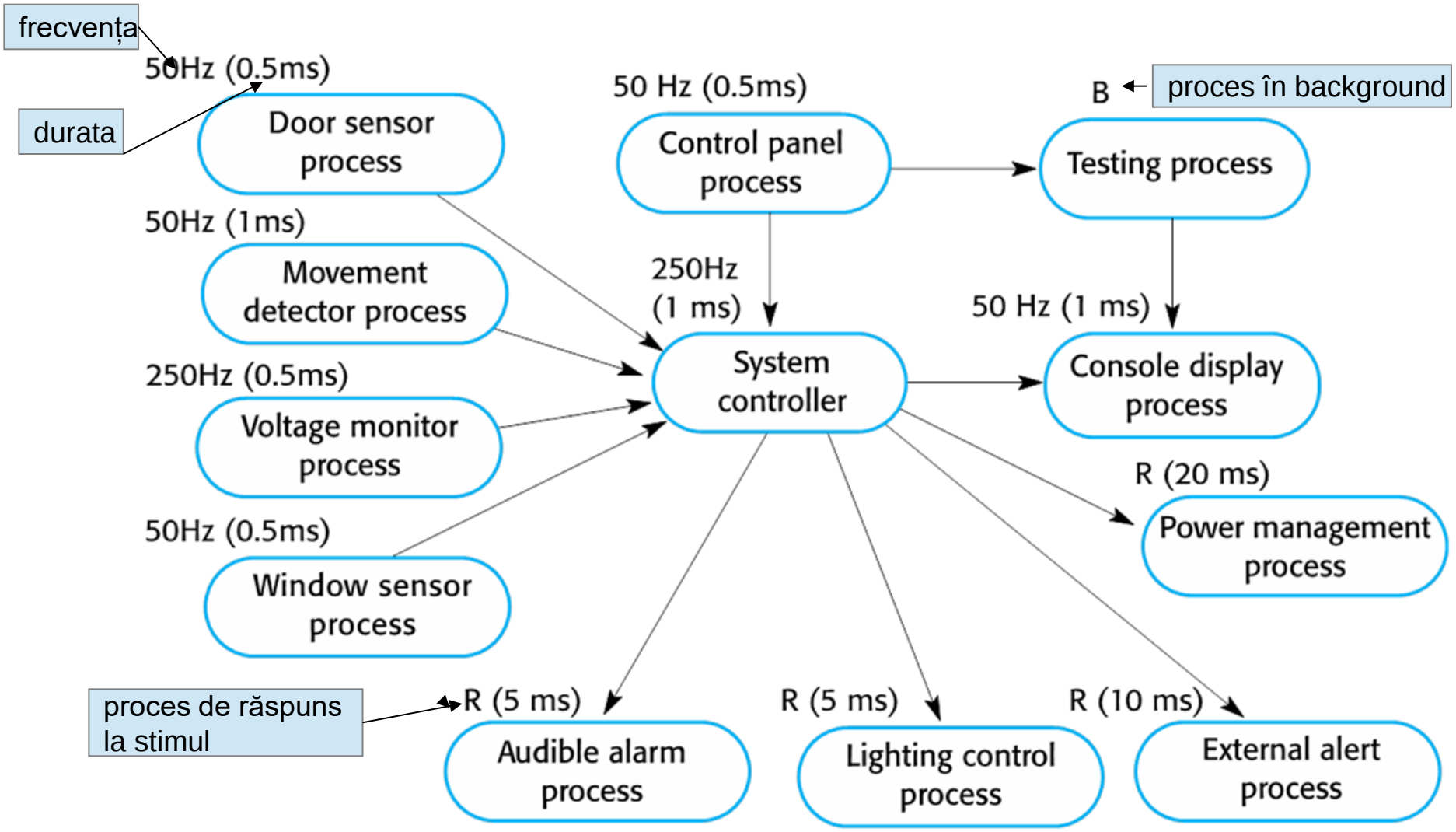
Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CERINȚE DE TIMP
PENTRU SISTEMUL DE ALARMĂ ANTI-FURT

STIMUL	RĂSPUNS
Alarmă sonoră	Alarma sonoră trebuie pornită în termen de 0.5 sec de la pornirea unei alarme de către un senzor.
Comunicații	Apelul la poliție trebuie lansat în max. 2 sec de la pornirea unei alame de către un senzor.
Alarmă de ușă	Fiecare alarmă de ușă trebuie sondată de 2 ori pe secundă.
Comutator lumină	Luminile trebuie aprinse în max. 0.5 sec de la pornirea unei alarme de către un senzor.
Detector mișcare	Fiecare detector de mișcare trebuie sondat de 2 ori pe secundă.
Cădere tensiune	Comutarea la alimentarea de rezervă trebuie realizată în termen de 50 msec.
Sintetizator voce	Mesajul sintetizat trebuie să fie disponibil în 2 sec de la pornirea unei alarme de către un senzor.
Alarmă de fereastră	Fiecare alarmă de fereastră trebuie sondată de 2 ori pe secundă.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CERINȚELE DE TIMP
PENTRU PROCESELE SISTEMULUI DE ALARMĂ ANTI-FURT



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

STIMULII DE PROCESAT

1. Căderea alimentării - detectată prin observarea unei căderi de tensiune de peste 20%.

Răspunsul : comutarea către bateria de rezervă, declanșată de un semnal către circuitul de comutarea pe bateria de rezervă.

2. Alarma de intrus - stimul generat de unul din senzorii sistemului.

Răspunsul : calcularea numărului camerei în care se află senzorul activ, lansarea unui apel către poliție, inițierea sintetizatorului de voce pentru acest apel, pornirea alarmei sonore și aprinderea luminilor din zonă.

FRECVENȚE ȘI TIMPI DE EXECUȚIE

Planificarea proceselor se va face de către sistemul de operare de timp real (*ROTS*) pe care se instalează aplicația de timp real.

Prioritățile acordate proceselor depind de *importanța* și *criticalitatea* acestora.

Exemplu : procesul de monitorizare a tensiunii de alimentare va avea prioritate superioară celei a proceselor de monitorizare senzori.

Evaluare formativă

1. Enumerați și descrieți pe scurt șabloanele arhitecturale specifice sistemelor RT/E.

2. Sistemele de timp real sunt proiectate ca *procese cooperante* controlate de un *executiv de timp real* (real-time executive). Care este rolul analizei de timp și care sunt factorii considerați în analiza de timp a acestor procese cooperante.

https://forms.gle/5SHHuntJMeWVRkKcA

PLAN CURS

Proiectarea sistemelor embedded

Șabloane arhitecturale pentru aplicații RT/E

Analiza de timp

Sisteme de operare de timp real

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

SISTEME DE OPERARE DE TIMP REAL

Platforme de execuție pentru sisteme embedded :

A. *bare metal* – platforma hardware

- Funcții incluse :
  - startup, shutdown
  - management procese și resurse
  - planificare procese

B. Sisteme de operare de timp real (RTOS)

- SO eficient, adaptat cerințelor de timp real
- Exemple : Windows CE, VxWorks, RTLinux, FreeRTOS

https://curlie.org/Computers/Software/Operating_Systems/Realtime

SISTEME DE OPERARE DE TIMP REAL

- Sistem de operare de timp real = sistem de operare specializat care gestionează procesele din sistemele de timp real.
- Responsabil de *administrarea proceselor* și de *alocarea resurselor* (procesor și memorie).
- Poate fi bazat pe un nucleu (kernel) standard utilizat ca atare sau modificat pentru o aplicație particulară.
- În general nu include facilități de administrare de fișiere.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

COMPONENTE SISTEM DE OPERARE DE TIMP REAL

- Real-time clock
  - Oferă informații pentru planificarea proceselor
- Interrupt handler
  - Administrează cererile aperiodice de servicii
- Scheduler
  - Alege procesul ce urmează să ruleze
- Resource manager
  - Alocă resurse de procesor și memorie
- Dispatcher
  - Pornește execuția proceselor

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

COMPONENTE SISTEM DE OPERARE DE TIMP REAL

Manager de configurare

Responsabilități :

–reconfigurarea dinamică a componentelor software și hardware ale sistemului.

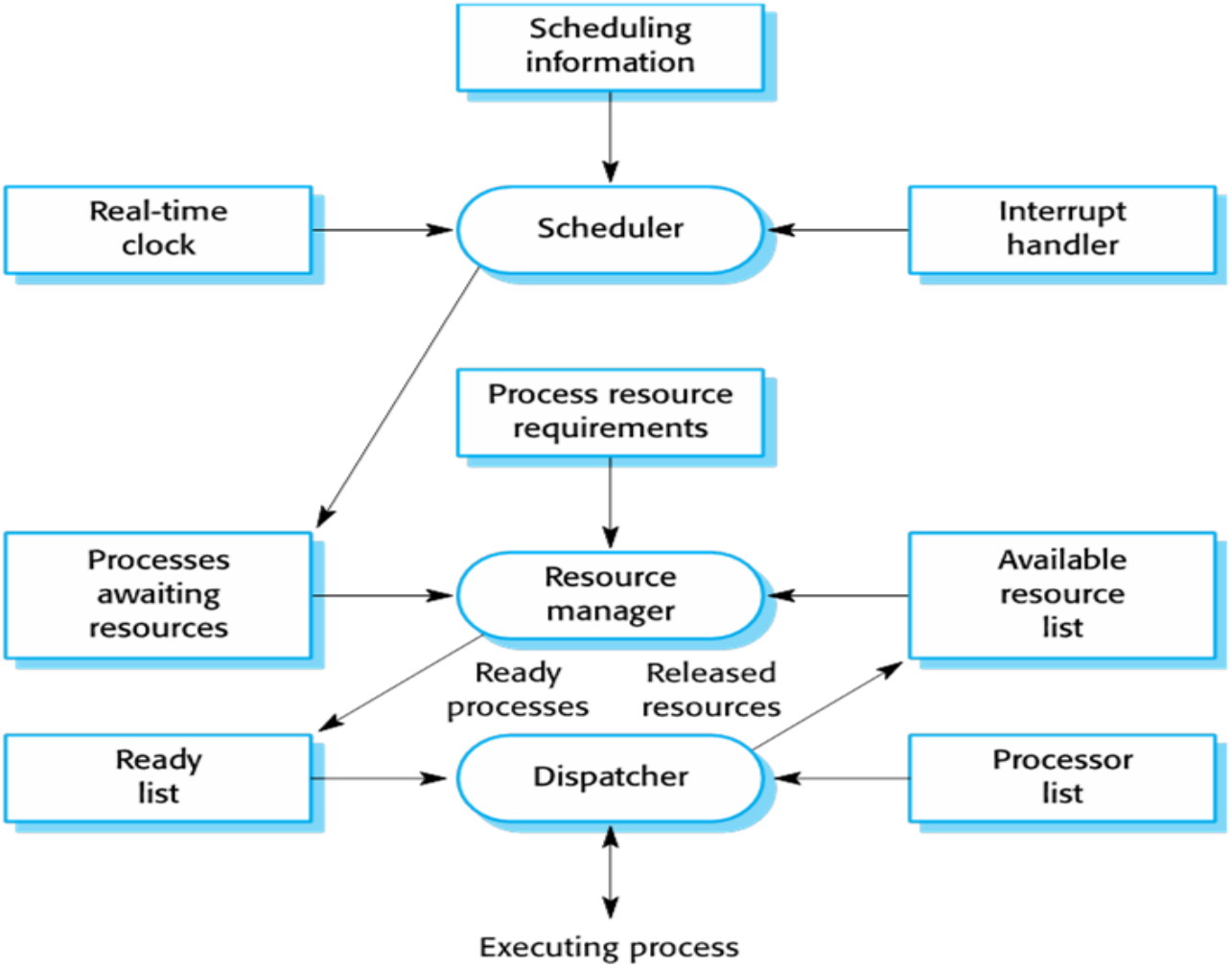
–fără a se opri sistemul pot fi înlocuite module hardware și pot fi actualizate componente software.

Manager defecte

Responsabilități :

- detectarea defectelor în hardware și în software
- realizarea acțiunilor corespunzătoare pentru a continua funcționarea sistemului.

COMPONENTE SISTEM DE OPERARE DE TIMP REAL



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

ADMINISTRAREA PROCESELOR

Administrarea unui *set de procese concurente*.

Procesele periodice sunt executate la intervale de timp pre-determinate.

- Sistemul de operare de timp real (ROTS)
  - folosește ceasul de timp real pentru a determina când să execute un anumit proces periodic,
  - alocă fiecărui proces resurse suficiente.

  în funcție de :
    - perioada procesului (timpul dintre 2 execuții),
    - timpul de execuție a procesului,
    - termenul procesului – timpul până la care trebuie să fie finalizat.

ADMINISTRAREA PROCESELOR

Cerință : procesele asociate *evenimentelor excepționale și critice* trebuie executate imediat ce apar stimulii respectivi.

Soluție : alocare *nivele de prioritate*.

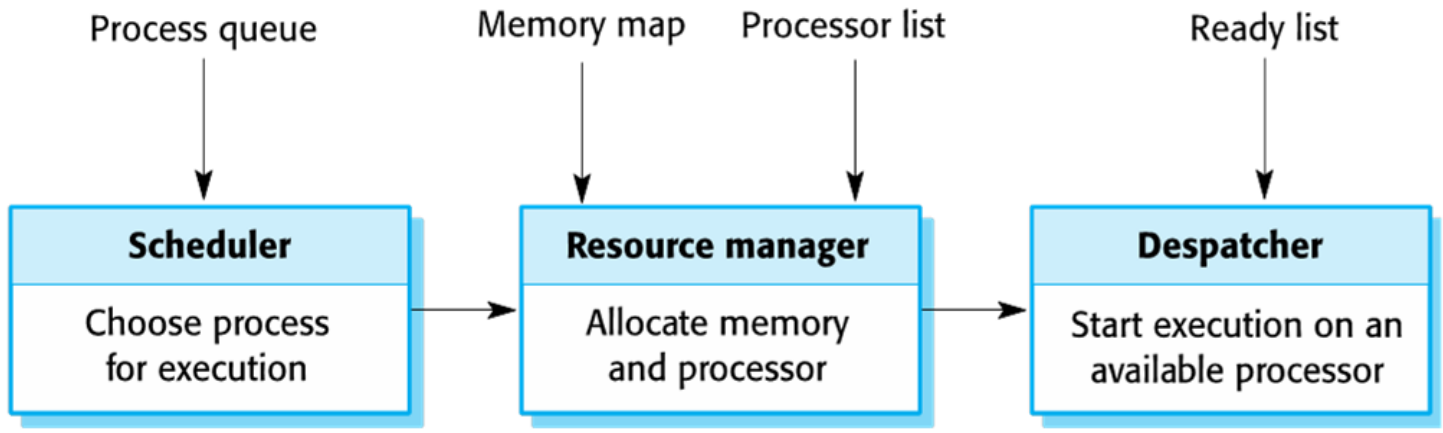
Clase de nivele de prioritate :

- Prioritate la nivel de *întrerupere* - prioritate mare alocată proceselor ce trebuie să răspundă rapid (inclusiv ceasului de timp real).
- Prioritate la nivel de *ceas* – alocată proceselor periodice.
- Prioritate de *background* – alocată proceselor de background, care vor fi planificate pentru execuție atunci când procesorul e liber.

În cadrul unei clase pot exista mai multe nivele de prioritate.

SERVIREA PROCESELOR PERIODICE

- Majoritatea sistemelor de timp real au câteva clase de procese periodice, fiecare cu *perioadă, timp de execuție și termen* specifice.
- Ceasul de timp real generează întreruperi cu o perioadă fixă, care lansează procesul planificator (*scheduler*) al proceselor periodice.
- Procesul planificator selectează procesul care va fi executat.

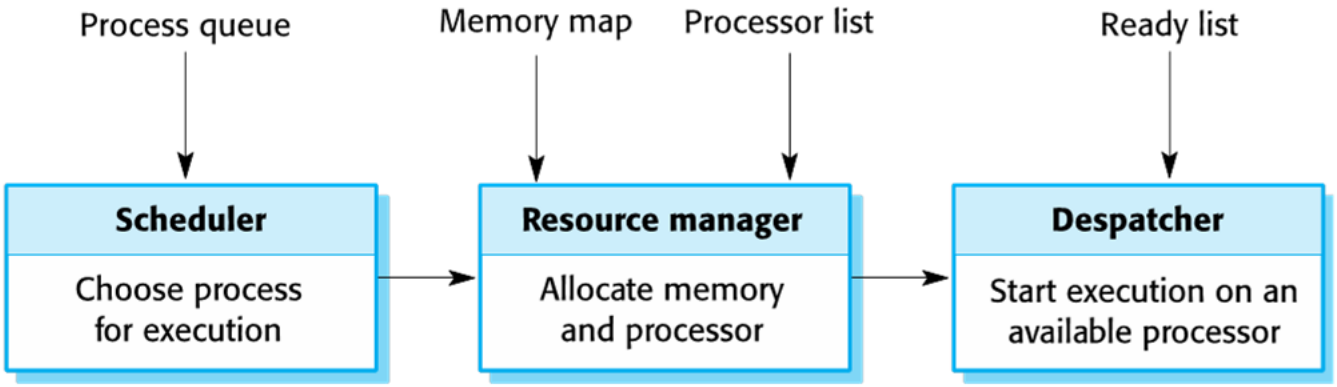


Etapele necesare pornirii unui proces periodic.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

COMUTAREA PROCESELOR

- Planificatorul alege următorul proces ce va primi procesor. Aceasta depinde de strategia de planificare, care poate lua în considerare prioritatea proceselor.
- Managerul de resurse alocă memorie și procesor pentru procesul ales
- Dispecerul ia procesul din lista proceselor gata de execuție, îl încarcă în procesor și pornește execuția.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

STRATEGII DE PLANIFICARE

- Planificare non-preemptivă
  - Un proces în execuție va rula până la terminarea sa sau până când se va bloca (ex. în așteptarea unei operații de I/E).
- Planificare preemptivă
  - Un proces în execuție poate fi oprit dacă un proces cu prioritate mai mare solicită servire.
- Algoritmi de planificare
  - Round-robin
  - Rate monotonic (priorități asignate static funcție de durata procesului)
  - Shortest deadline first

Cum se integrează în acest context
servirea proceselor care au priorități la nivel de întrerupere hard ?

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CONCLUZII

- Un sistem software embedded este o parte a unui sistem hardware/software care reacționează la evenimente din contextul său. Software-ul este înglobat în hardware. Sistemele embedded sunt, în mod normal, sisteme de timp real.
- Un sistem de timp real este un sistem software care trebuie să răspundă la evenimente în timp real. Corectitudinea sistemului depinde atât de rezultatele produse cât și de momentul producerii acestora.
- Sistemele de timp real sunt implementate în general pe un set de procese concurente și comunicante care reacționează la stimuli pentru a produce răspunsuri.
- Modelele de stări și tranziții sunt o reprezentare importantă din modelul sistemelor de timp real embedded. Acestea sunt utilizate pentru a ilustra modul în care sistemul reacționează la contextul său pe măsură ce evenimentele declanșează modificări în starea sistemului.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 21

CONCLUZII

- Există câteva șabloane standard care pot fi observate la diferite tipuri de sisteme embedded. Acestea includ un șablon de monitorizare a contextului sistemului referitor la evenimente adverse, un șablon pentru control al contextului cu elemente de acționare, și un șablon pentru procesare date.
- Proiectanții sistemelor de timp real trebuie să realizeze o analiză de timp, care depinde de termenele de procesare a stimulilor și de generare a răspunsurilor. Ei trebuie să decidă frecvențele de execuție ale proceselor din sistem și durata de execuție a acestora în cazul celor mai defavorabile scenarii.
- Un sistem de operare de timp real este responsabil cu managementul proceselor și al resurselor pe care acestea se execută. Include un proces planificator, care este componenta responsabilă să decidă care proces trebuie lansat în execuție.

Bibliografie

- Ian Sommerville, Software Engineering, ed 10, cap 21.
- UML Profile for MARTE

http://www.omg.org/spec/MARTE/

- SUPLIMENTAR :

Jim Cooling, *Software Engineering for Real Time Systems*, Addison-Wesley, 2003

Bruce Powe Douglas, *Real-Time Design Patterns. Robust Scalable Architecture for Real Time Systems*, Adison-Wesley, 2002