



UNIVERSITATEA  
DIN BUCUREȘTI

# Metode de dezvoltare software

---

Cerințe software

06.03.2019

Alin Ștefănescu

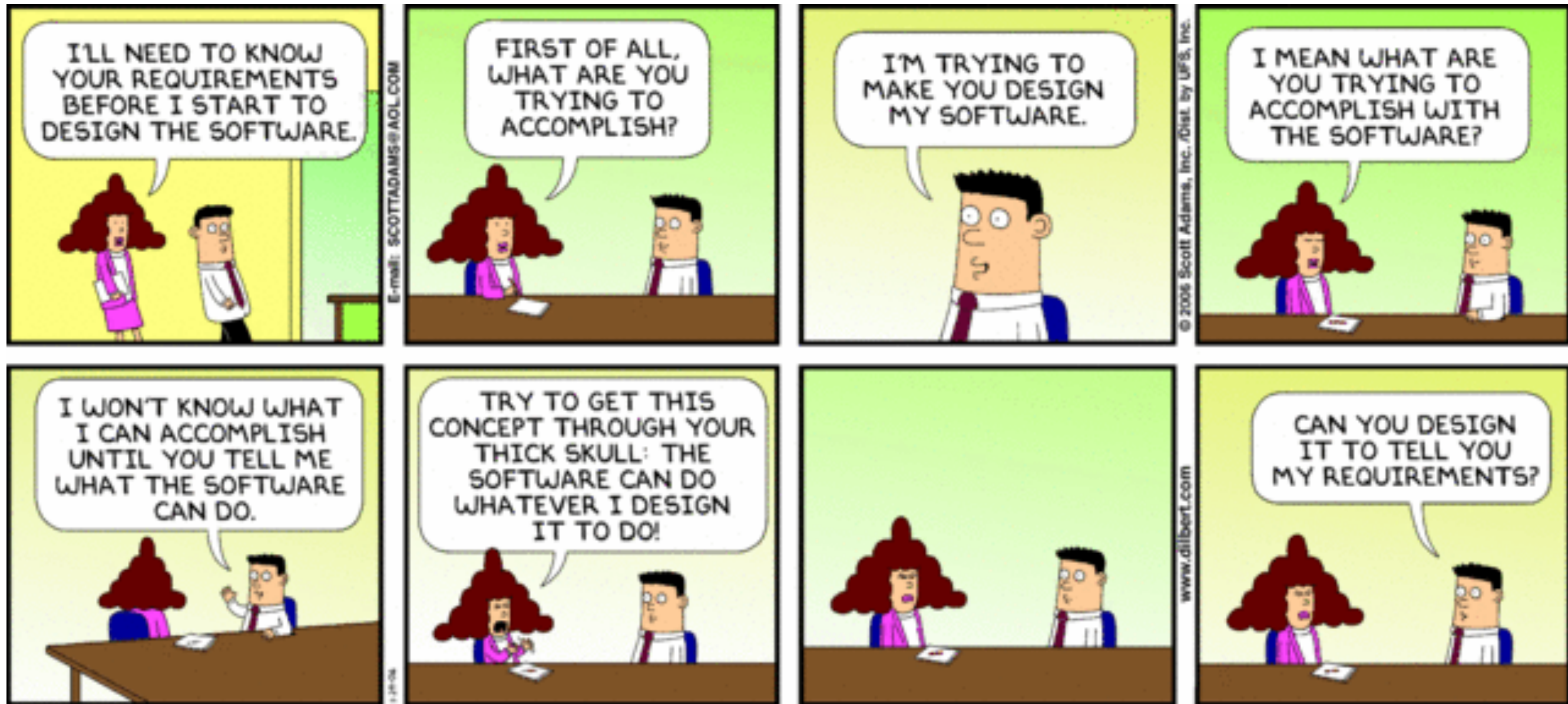
A black laptop is open, showing a bright orange screen. The word "Cerințe" is written in white, bold, sans-serif font in the center of the screen. The laptop is positioned against a dark, textured background.

# Cerințe

Prezentare bazată pe:  
“Software Engineering – 10th ed. (2015)” de Ian Sommerville  
<http://iansommerville.com/software-engineering-book>



# Negocierea cerințelor... în practică



de la [dilbert.com](http://dilbert.com)

# Despre cerințe (engl. *requirements*)

- **Cerințele** sunt descrieri ale serviciilor oferite de sistem și a constrângerilor sub care acesta va fi dezvoltat și va opera
- Cerințele pornesc de la afirmații abstracte de nivel înalt până la specificații matematice funcționale detaliate
- Cerințele îndeplinesc mai multe funcții:
  - pot constitui baza negocierilor pentru un contract – ca atare, trebuie să fie deschise diverselor interpretări
  - pot constitui baza contractului în sine – ca atare, trebuie definite cât mai detaliat

# Tipuri de cerințe

## USER STORIES

### ■ Cerințe utilizator

- afirmații în limbaj natural și diagrame ale serviciilor oferite de sistem laolaltă cu constrângerile operaționale.
- scrise pentru clienți.

### ■ Cerințele sistemului

- un document structurat stabilind descrierea detaliată a funcțiilor sistemului, serviciile oferite și constrângerile operaționale.
- poate fi parte a contractului cu clientul.

# Utilizarea cerințelor

## ■ **Cerințele utilizator** se adresează:

- utilizatorilor finali
- inginerilor si managerilor clientului
- proiectanților de sistem (arhitecți)
- managerilor de contracte

## ■ **Cerințele de sistem** se adresează:

- utilizatorilor finali
- inginerilor clientului
- proiectanților de sistem (arhitecți)
- programatorilor

# Utilizatorii documentului cu cerințe

- **clienții:** impun cerințele și verifică apoi dacă acestea sunt conforme cu nevoile lor. Pot cere modificări ale cerințelor.
- **managerii:** utilizează documentul pentru a stabili termenii contractului și a planifica procesul de dezvoltare
- **inginerii de sistem:** utilizează documentul pentru a înțelege ce trebuie dezvoltat
- **inginerii de la testare:** utilizează documentul pentru a proiecta testele de validare

# Cerințe funcționale și non-funcționale

## ■ Cerințe funcționale:

- *afirmații despre servicii* pe care sistemul trebuie să le conțină, cum trebuie el să răspundă la anumite intrări și cum să reacționeze în anumite situații.

## ■ Cerințe non-funcționale:

- *constrângeri* ale serviciilor și funcțiilor oferite de sistem cum ar fi: constrângeri de timp, constrângeri ale procesului de dezvoltare, standarde, etc.



# Cerințe funcționale

- Descriu funcționalitatea sistemului și serviciile oferite
- Depind de tipul softului, de utilizatorii avuți în vedere și de tipul sistemului pe care softul este utilizat
- Cerințele funcționale ale utilizatorilor pot fi descrieri de ansamblu, dar cerințele funcționale ale sistemului trebuie să descrie în detaliu serviciile oferite

# Cerințe non-funcționale

- Definesc *proprietăți și constrângeri ale sistemului*.  
D.ex.: fiabilitatea, timpul de răspuns, cerințele pentru spațiul de stocare, cerințe ale sistemului de intrări-ieșiri etc.
- La întocmirea lor se va ține cont de un anumit mediu de dezvoltare, limbaj de programare sau metodă de dezvoltare
- Cerințele non-funcționale pot fi mai importante decât cele funcționale. Dacă nu sunt îndeplinite, sistemul nu va fi util scopului pentru care a fost dezvoltat.

# Tipuri de cerințe non-funcționale

## ■ Cerințe ale **produsului**

- Cerințe care specifică un anumit comportament al produsului. D. ex.: gradul de utilitate, eficiență (viteză de execuție), fiabilitate, portabilitate etc.

## ■ Cerințe legate de **organizare**

- Cerințe care sunt consecințe ale politicilor de organizare a producției software. D. ex.: standarde utilizate, cerințe de implementare, cerințe de livrare, securitate etc.

## ■ Cerințe **externe**

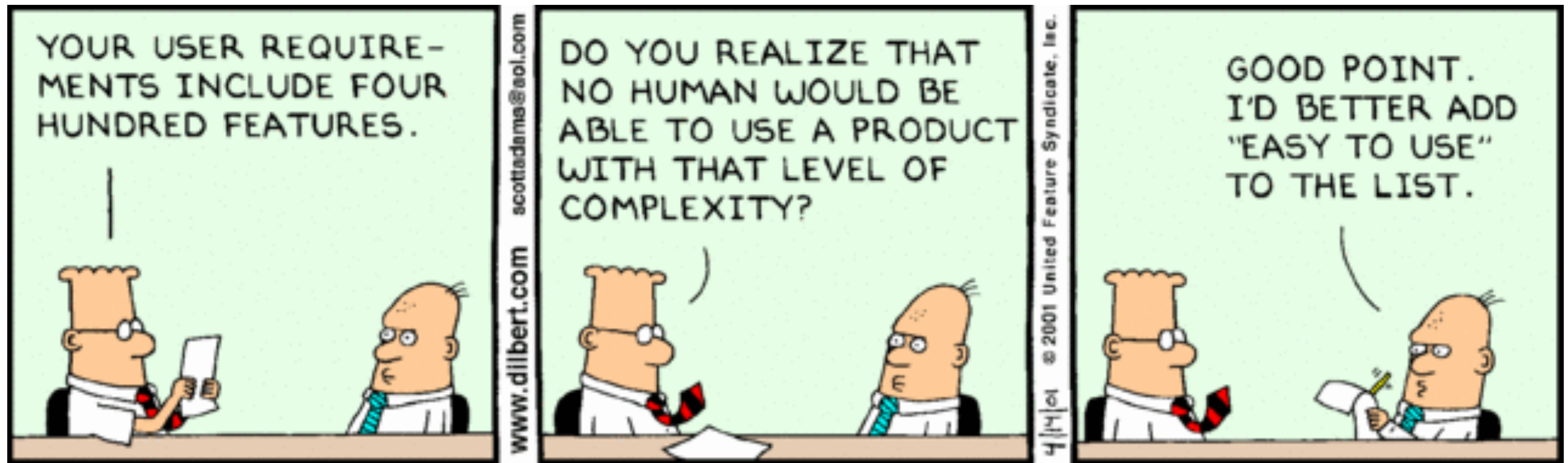
- Cerințe asociate unor factori externi. D. ex.: cerințe de interoperabilitate, cerințe legislative etc.

# Tipuri de cerințe non-funcționale

Proprietate	Exprimare (exemple)
Viteză	Tranzacții/sec; Timp de răspuns la utilizator
Dimensiune	MB; Nr. de cipuri necesare
Ușurința utilizării	Timp de învățare; dimensiunea “help”-ului
Fiabilitate	Timpul mediu dintre două defecte; Rata de apariție a defectelor
Robustețe	Probabilitatea de corupere a datelor la eroare; Timpul de restart după apariția unui defect
Portabilitate	Procentul de linii de cod dependente de ținta implementării; Numărul de sisteme țintă

# Cerințe ale utilizatorilor

- Cerințele utilizatorilor trebuie să descrie cerințe funcționale și non-funcționale pe înțelesul utilizatorilor sistemului care nu dețin cunoștințe tehnice detaliate.



de la [dilbert.com](http://dilbert.com)



# Recomandari pentru scrierea cerințelor

- adoptarea unui **format standard** și utilizarea lui la toate cerințele
- utilizarea limbajului într-un format consistent. E bine să se folosească **trebuie** pentru cerințele obligatorii și **ar trebui** pentru cele opționale
- **evidențierea textului** care identifică părțile importante ale cerințelor
- evitarea jargonului
- dacă este cazul, includerea unei explicații de ce este necesară o anumită cerință

# Cerințele sistemului

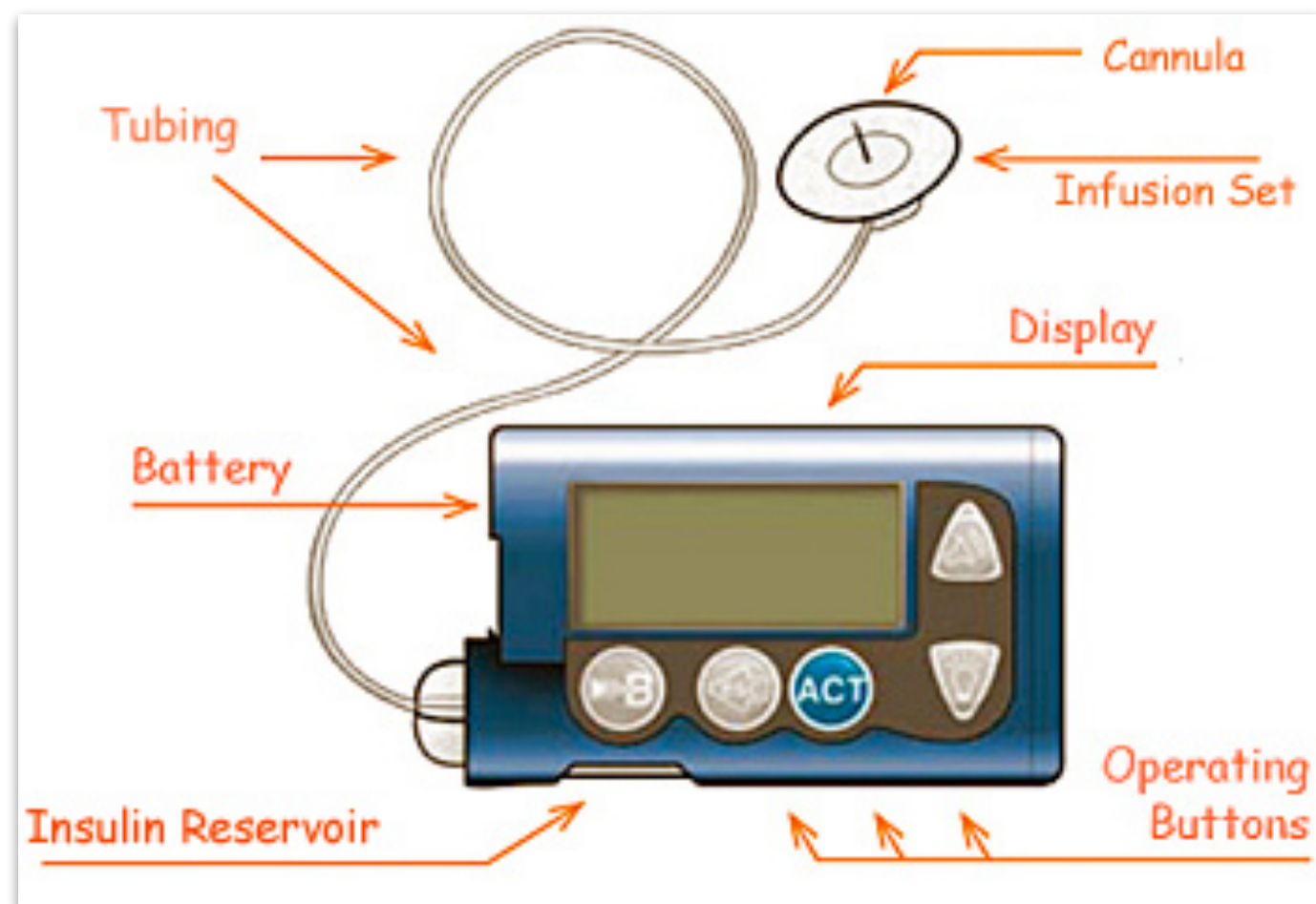
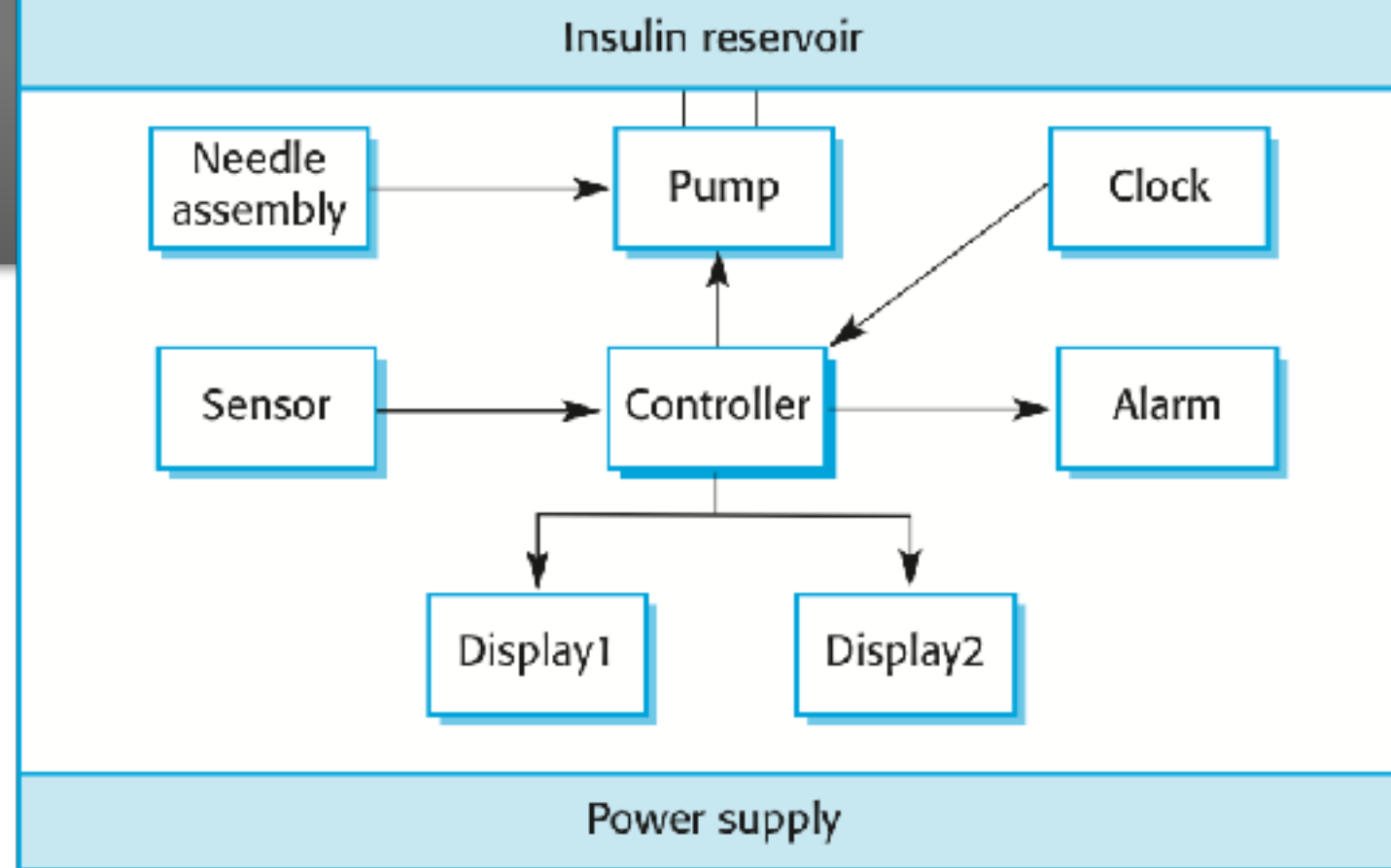
- cerințele sistemului sunt specificate mai detaliat decât cerințele utilizatorilor
- scopul principal al lor este acela de a fi baza proiectării sistemului
- cerințele trebuie să exprime **ce poate face** sistemul, iar proiectul trebuie să exprime **cum se poate implementa** sistemul
- ele pot fi incorporate în contract

# Posibilități de reprezentare a cerințelor

- **limbaj natural:** Cerințele sunt organizate în paragrafe numerotate.
- **limbaj natural structurat:** utilizarea unui format standard sau a machetelor în conjuncție cu limbajul natural
- **limbaj grafic** suplimentat cu adnotări textuale (mai ales pentru cerințe sistem), cum ar fi UML.
- **specificații matematice:** concepte matematice lucrând cu mașini cu stări finite sau relații peste mulțimi (limbajul Z). Elimină ambiguitățile, dar pot fi dificil de înțeles.

# Exemplu - pompă de insulină

- Colectează date de la senzorul pentru glicemie (zahărul din sânge) și calculează cantitatea de insulină care trebuie injectată
- Calculul se bazează pe diferențele glicemiei în timp.
- Trimite semnale cu valoarea dozei de insulină unei micropompe.
- Funcționarea corectă este critică pentru sănătatea pacientului.



# Exemple de cerințe în limbaj natural

- Cerința 3.2: Sistemul măsoară glicemia la fiecare 10 minute și, dacă e nevoie, pompează insulină.

*(Explicație: Modificarea glicemiei e relativ lentă, astfel că măsurători mai frecvente nu sunt necesare. Pe de altă parte, măsurători prea rare pot rata valori periculoase ale glicemiei.)*

- Cerința 3.6: Sistemul va rula un test intern o dată pe minut pentru a verifica funcționarea corectă a pompei, conform tabelului 1.

*(Explicație: Un test intern poate detecta probleme hardware sau software, alertând utilizatorul în cazul unor disfuncționalități.)*



# Exemplu de cerință structurată

## Insulin Pump / Control Software / SRS / 3.3.2

**Function:** Calculează doza de insulină pentru a menține glicemia în limite normale.

**Description:** Calculează doza de insulină care trebuie injectată pentru a menține glicemia în limitele normale de 3-7 unități.

**Inputs:** Valoarea curentă (*r2*); două valori anterioare (*r0* și *r1*).

**Source:** Valoarea citită de senzor. Celelalte valori din memorie.

**Outputs:** *CompDose* - doza de insulină care trebuie injectată.

**Destination:** Ciclul de control al pompei.

# Exemplu de cerință structurată continuat

## Insulin Pump / Control Software / SRS / 3.3.2 (continuare)

**Action:** *CompDose* este zero dacă glicemia e stabilă sau scade, sau dacă nivelul ei crește, dar rata de creștere e în scădere. Dacă nivelul crește și rata de creștere de asemenea crește, atunci *CompDose* se calculează împărțind la 4 diferența dintre  $r2$  și  $r1$  și rotunjind rezultatul. Dacă rezultatul rotunjit este zero, atunci *CompDose* ia valoarea predefinită a dozei minime.

**Requirements:** Două valori anterioare pentru a putea calcula rata de schimbare a glicemiei.

**Pre-condition:** Rezervorul de insulină conține cel puțin valoarea maximă permisă a unei doze de insulină.

**Post-condition:**  $r0$  este înlocuit cu  $r1$  și  $r1$  cu  $r2$ .

**Side effects:** Niciunul.

# Specificație tabelară (completează limbajul natural)

Condiție	Acțiune
Nivelul glicemiei scade: $(r2 < r1)$	$CompDose := 0$
Nivelul glicemiei stabil: $(r2 = r1)$	$CompDose := 0$
Nivelul glicemiei crește și rata de creștere descrește: $0 < (r2 - r1) < (r1 - r0)$	$CompDose := 0$
Nivelul glicemiei crește și rata de creștere e stabilă sau crește: $(r2 - r1) \geq (r1 - r0) > 0$	$CompDose := round((r2 - r1)/4)$ Dacă rezultatul de mai sus e 0, $CompDose := MinimumDose$

# Tooluri pentru documentarea cerintelor

- **GitLab**: <https://about.gitlab.com/solutions/project-management>

Licente academice pentru UB - **recomandat**

Pentru a obtine o licenta academica, completati acest formular:

<http://bit.do/cere-cont-ub-gitlab>

- **GitHub**: <https://github.com/features/project-management>

Licente academice pentru UB

- **Jira**: <https://www.atlassian.com/software/jira>

- **Trello**: <https://trello.com>

- **PivotalTracker**: <https://www.pivotaltracker.com>

- etc.