

# CURS 01A.

# VERIFICARE ȘI VALIDARE

---

**Verificarea și validarea sistemelor soft**  
**[25 Februarie 2020]**

Lector dr. Camelia Chisăliță-Crețu  
Universitatea Babeș-Bolyai

# Conținut

- Calitatea produselor soft
  - Stakeholders
  - Definiții ale calității produselor soft
  - Activități asociate controlului calității
- Verificare și validare
- Defect software
  - Terminologie
  - Costul unui bug software
- Bug-uri software celebre
- Bibliografie

# CALITATEA PRODUSELOR SOFT

---






Stakeholders

Definiții ale calității produselor soft

Activități asociate controlului calității unui produs soft

# Managementul publicațiilor la BJ

- O aplicație pentru gestionarea publicațiilor la Biblioteca Județeană.

				
Emil (47), Director General BJ	Clara (44), Bibliotecar	Anna (23), Bibliotecar	Alex (37), Programator, Manager firma IT	George (29), Programator Software

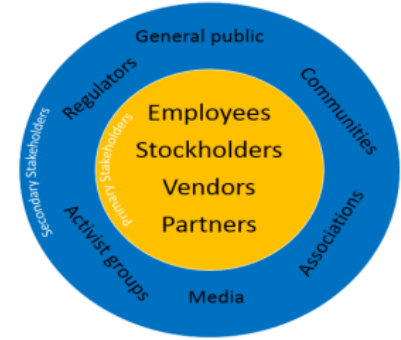
# Stakeholders

- **stakeholder** (*rom. beneficiar, utilizator*)
  - o persoană care manifestă un interes particular pentru **succesul** sau **eșecul** unui produs soft [\[BBST2010\]](#).



[\[CFI2019\]](#)






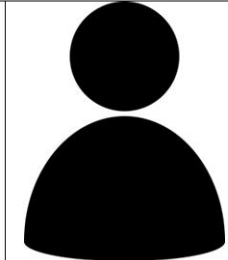
# Tipuri de Stakeholders



- **primar/secundar** (*engl.*, primary, secondary) [\[StakeholderMap2019\]](#):
  - **beneficiar primar** – direct afectat de succesul sau eșecul produsului;
  - **beneficiar secundar** – nu este afectat direct de succesul sau eșecul produsului.
- **preferat/nedorit** (*engl.*, favored, disfavored) [\[GauseWeinberg2011\]](#), [\[KanerBach2005\]](#):
  - **beneficiar preferat (avantajat)** – produsul este proiectat pentru a fi utilizat de acesta;
  - **beneficiar nedorit (dezavantajat)** - produsul este proiectat sa creeze dificultăți în utilizare;
  - **beneficiar neutru** – produsul nu este proiectat pentru acesta și nu îl poate influența;
  - **beneficiar ignorat (neglijat)** – produsul nu este proiectat pentru fi utilizat de acest utilizator.

# Exercițiu

- Clasificați următorii beneficiari pe baza categoriilor:
  - A. beneficiar primar/secundar;
  - B. beneficiar preferat/ nedorit/ neutru/ ignorat.

 <p>Emil (47), Director General BJ</p>	 <p>Clara (44), Bibliotecar</p>	 <p>Anna (23), Bibliotecar</p>	 <p>Alex (37), Programator, Manager firma IT</p>	 <p>George (29), Programator Software</p>	 <p>???</p>
---	--	---	--	--	--

# Calitatea produselor soft. Definiții (1)

- ***“produsul soft este conform cu cerințele documentate”*** [[Pressman2005](#)]:
  - conformitatea cu cerințele funcționale și de performanță precizate și documentate explicit în standarde de dezvoltare și caracteristicile implicite pe care un produs soft dezvoltat le are;
- ***“produsul soft este conform cu cerințele reale ale utilizatorului”*** [[Crosby1980](#)]:
  - conformitatea cu cerințele reale ale utilizatorului care pot fi incluse sau nu în specificațiile scrise;
  - **conformitate cu cerințele (nevoile) reale, nu doar cu cerințele documentate;**





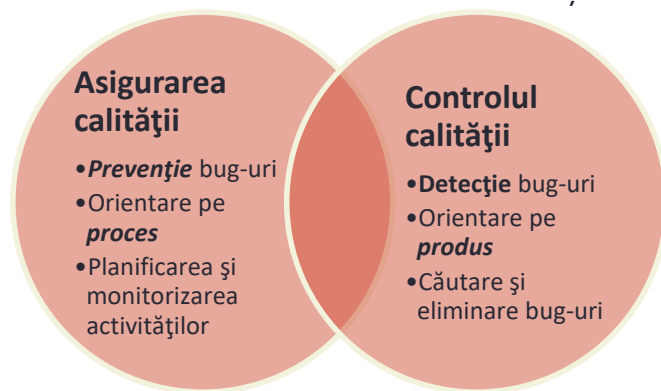
# Calitatea produselor soft. Definiții (2)

- ***“produsul soft este adecvat pentru a fi utilizat”*** [[Juran1998](#)]:
  - *satisfiers* – orice aspect care îl mulțumește pe beneficiar;
  - *dissatisfiers* – orice aspect care îl nemulțumește pe beneficiar;
- ***“produsul soft este relevant/important pentru o persoană”*** [[Weinberg1992](#)]:
  - **calitatea este subiectivă;**
  - un aspect care are relevanță/importanță însemnată pentru un utilizator poate fi mai puțin important pentru un alt utilizator din aceeași categorie de utilizatori.



# Activități asociate calității

- *în procesul de dezvoltare, calitatea este abordată din perspectiva:*
  - **procesului** ==> **asigurarea calității** (*engl. quality assurance*):
    - **Obiectiv:** asigură respectarea standardelor, planurilor și etapelor proceselor de dezvoltare necesare elaborării adecvate a produsului cerut;
    - **Întrebare:** Cum se asigură calitatea activităților desfășurate în procesul dezvoltare?
  - **produsului** ==> **controlul calității** (*engl. quality control*):
    - **Obiectiv:** identifică deficiențele în produsul obținut;
    - **Întrebare:** Cum se controlează calitatea rezultatelor obținute (e.g., work products) în urma activităților desfășurate?



# VERIFICARE ȘI VALIDARE

---

Verificare

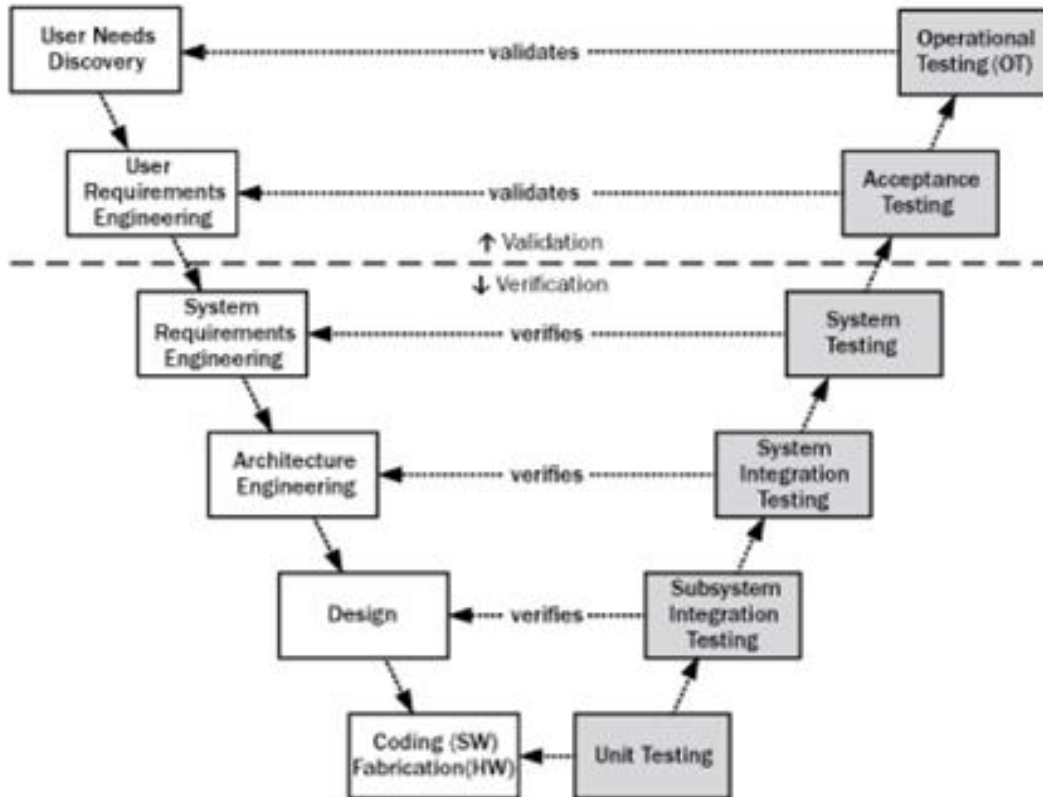
Validare

Verificare vs. Validare

# Verificare și Validare. Definiție SEI

- SEI (Software Engineering Institute) [[NT2005](#)]
- **Verificare**
  - procesul prin care se asigură că produsul este dezvoltat conform cerințelor, specificațiilor și standardelor;
  - întrebare asociată: **Dezvoltăm corect produsul?** (*Are we building the product right?*)
- **Validare**
  - procesul prin care se asigură că produsul dezvoltat satisface cerințele utilizatorului;
  - întrebare asociată: **Dezvoltăm produsul corect (de care are nevoie clientul)?**  
(*Are we building the right product?*)

# Verificare și Validare în modelul V



• sursa: [Firesmith2015](#)

# Verificare vs. Validare

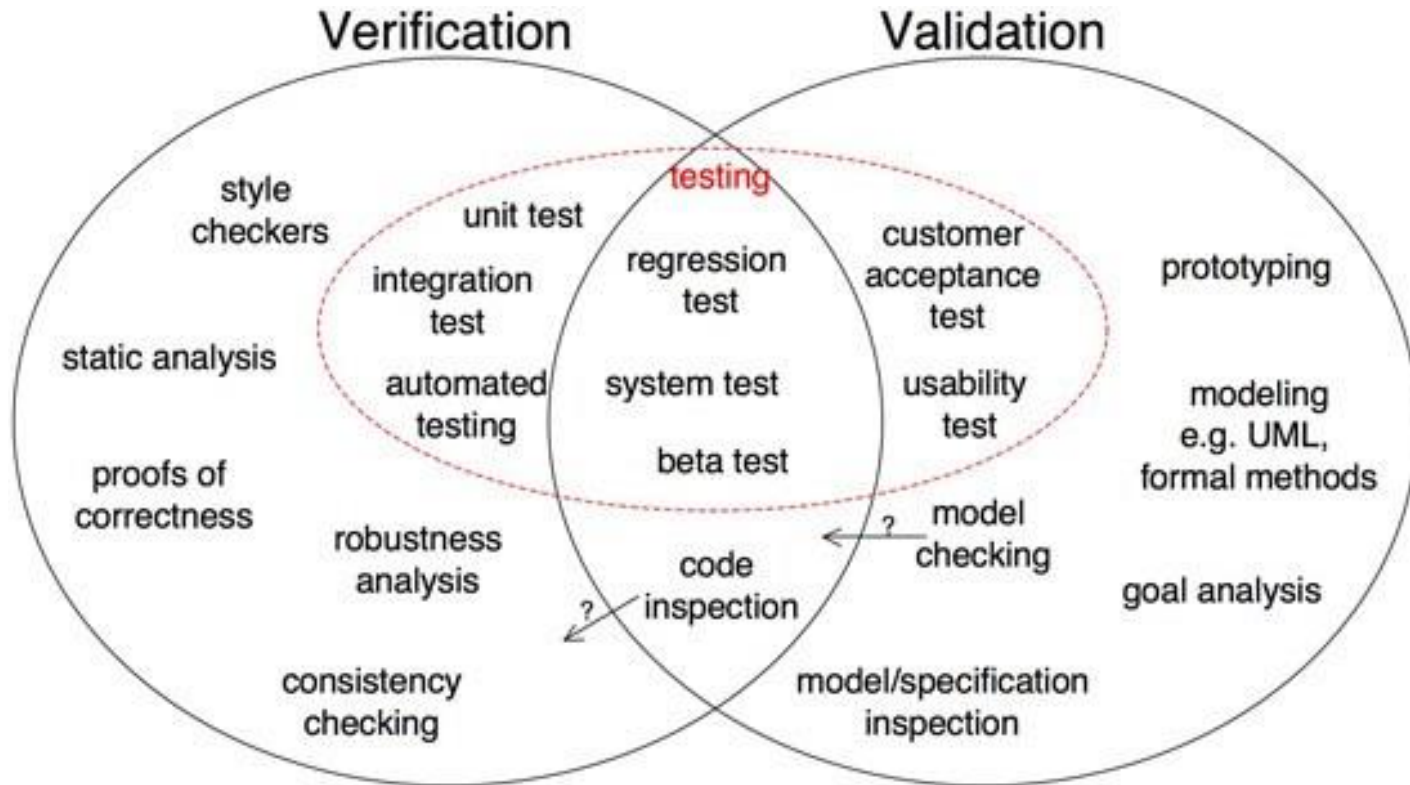
## Verificare

- stabilește dacă rezultatul unei etape de dezvoltare satisface cerințele acelei etape;
- asigurare a consistenței, completitudinii, corectitudinii;
- **aplică metode de control al calității;**

## Validare

- **confirmă dacă produsul satisface cerințele de utilizare;**
- se desfășoară spre sfârșitul procesului de dezvoltare, cu scopul de a demonstra că întregul sistem satisface nevoile și așteptările;
- se aplică asupra întregului sistem, în contextul real în care va funcționa, folosind diferite tipuri de testare.

# Activități de Verificare și Validare



# DEFECT SOFTWARE

---

Terminologie

Când apare un bug într-un produs soft?

De ce apare un bug în procesul de dezvoltare software?

Costul unui bug software

Defecte/Buguri software celebre



# Terminologie (1)

- **eroare** (*engl. error, mistake*; greșeală):
  - o acțiune umană care are ca rezultat un defect în produsul software [[Patton2005](#)];
- **defect** (*engl. fault*, i.e., **bug**):
  - consecință a unei erori [[Patton2005](#)];
  - un defect poate fi latent: nu cauzează probleme până când nu apar anumite condiții (*engl. failure triggers*) care determină execuția anumitor linii de cod sursă;
- **defecțiune** (*engl. failure*):
  - devierea de la comportamentul obișnuit al unei componente software;
  - apare atunci când comportamentul observabil al programului nu corespunde specificației sale;
  - procesul de manifestare a unui defect: când execuția programului întâlnește un defect, acesta provoacă o defecțiune [[Patton2005](#)];

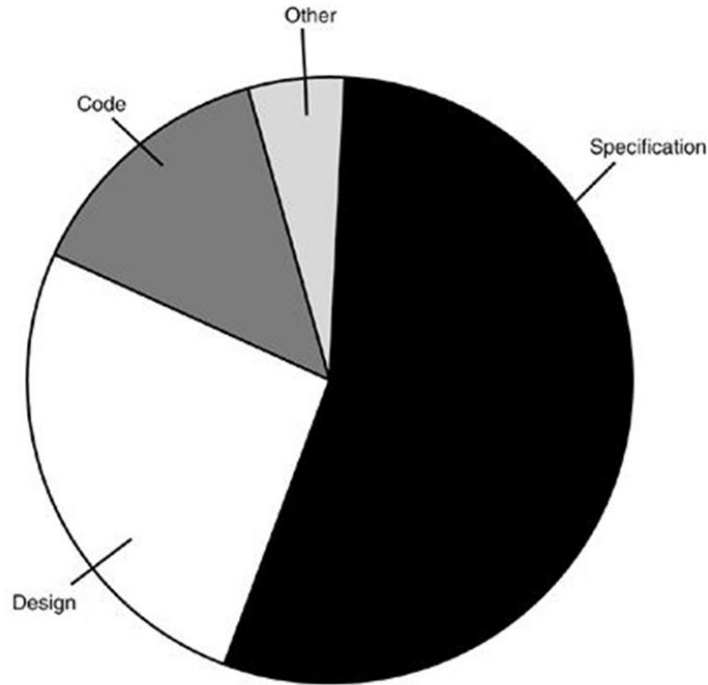
# Terminologie (2)

- **defect** (*engl.* **bug, software error**)
  - **orice aspect al unui produs soft care**
    - **cauzează reducerea inutilă și inadecvată a calității produsului soft** [\[BBST2008\]](#);
    - **constituie o amenințare asupra imaginii produsului** [\[BBST2008\]](#);
  - **exemple:** deficiențe de proiectare, greșeli în documentații, utilizare cu dificultate a programului;
  - **totuși, anumite aspecte ale produsului pot limita calitatea acestuia, dar nu pot fi considerate defecte!**
  - **exemplu:** constrângeri de utilizare precizate sau nu în specificații;
  - În cadrul acestui curs, orice deficiență sau problemă a produsului soft este denumită **bug (defect)**.
  - sinonime pentru bug: *engl.* **variance, problem, inconsistency, error, incident, anomaly** [\[Patton2005\]](#).

# De ce apare un bug într-un produs soft?

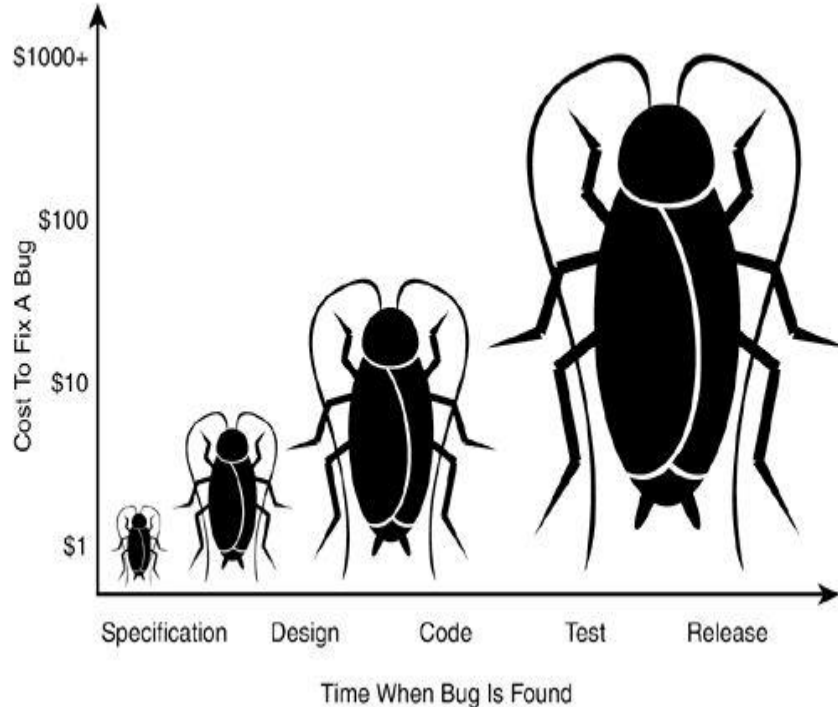
- Un bug software apare atunci când cel puțin una din următoarele situații are loc [[Patton2005](#)]:
  - Produsul soft nu face ce este precizat în specificația lui.
  - Produsul soft face ce nu este precizat în specificație.
  - Produsul soft face ce specificația precizează că **nu** trebuie făcut.
  - Produsul soft nu face ceea ce specificația ar trebui să precizeze.
  - Produsul soft este dificil de înțeles, greu de utilizat, lent. Testerul pune în evidență perspectiva utilizatorului final asupra produsului soft, adică produsul nu funcționează conform așteptărilor lui.

# În ce etapă a procesului de dezvoltare software apar bug-urile?



- **specificarea cerințelor:**
  - nu se scriu specificațiile, sunt superficiale, se schimbă continuu, nu sunt comunicate corespunzător întregii echipe de dezvoltare;
- **proiectare:**
  - sunt superficiale, nu se comunică eficient, se modifică;
- **implementare:**
  - complexitatea produsului soft, lipsa documentației (pentru codul sursă îmbunătățit), erori de redactare, presiunea termenului limită.
- **Care este etapa de dezvoltare în care se introduc cele mai multe defecte?**

# Cât costă eliminarea unui bug?



- Care sunt costurile de eliminare a unui bug software?
- costul eliminării bug-urilor crește pe măsură ce produsul soft este dezvoltat.

# BUG-URI SOFTWARE CELEBRE

---

10 bug-uri software celebre

# Buguri software celebre (1)

- **Naveta spațială Mariner 1 – 1962**

- naveta spațială Mariner 1 a deviat de la traiectoria ei la scurt timp după lansare spre planeta Venus; a fost distrusă la 293 secunde după lansare;
- **cauza:** eroare la scrierea unei instrucțiuni în limbajul FORTRAN, determinând calculul eronat al traiectoriei;
- **cost:** 18.5 milioane \$



# Buguri software celebre (2)

- **Tratamente împotriva cancerului – 1985**

- dispozitivul Therac-25 fost folosit în terapia prin radiații;
- **cauza:** programul a calculat greșit doza de radiații pe baza datelor de intrare, unii pacienți primind o doză de câteva ori mai mare decât cea normală;
- **cost:** 3 pacienți decedați, 3 răniți prin iradiere.





# Buguri software celebre (3)

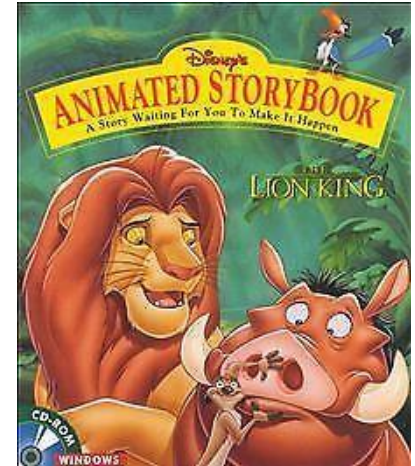
- **Sistemul de apărare american anti-rachetă – 1991**

- sistemul american de apărare antirachetă MIM-104 Patriot situat în Arabia Saudită nu a reușit să detecteze atacuri cu rachete Scud irakiene;
- **cauza:** o eroare de rotunjire la ceasul sistemului (un sfert de secundă) s-a cumulat, astfel încât la 14 ore, sistemul de urmărire își pierde acuratețea, devenind incapabil să localizeze și să intercepteze rachetele;
- **cost:** în atacul asupra unei cazarme din Dhahran au decedat 28 soldați americani;
- eroarea fusese deja remediată de experții armatei americane, iar noua versiune a softului urma să ajungă cu o zi mai târziu.



# Buguri software celebre (4)

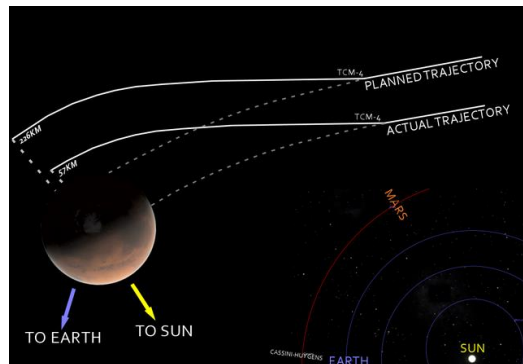
- **Jocul asociat desenului animat Disney Lion King – 1995**
  - la prima apariție pe piață a companiei Disney cu un joc pentru copii - ***The Lion King Animated Storybook*** - unii utilizatori nu au reușit să folosească produsul soft achiziționat;
  - **cauza:** compania Disney nu a testat produsul pe diferite modele de calculatoare personale existente pe piață;
  - **cost:** credibilitatea companiei, schimbarea unităților CD-ROM.



# Buguri software celebre (5)

- **Naveta spațială Mars Climate Orbiter – 1998**

- **obiectiv:** orbitarea planetei Marte și transmiterea informațiilor despre condițiile meteo;
- **eveniment:** după o călătorie de 286 zile de pe Pământ, la intrarea în atmosfera planetei Marte, motoarele au deviat traiectoria navei;
- **rezultat:** dezintegrarea navei în atmosferă;
- **cauza:** două dintre echipele implicate în dezvoltarea aplicației foloseau sisteme de măsurare a distanței diferite, imperial (**inch, feet**) și cel metric (**m, km**).



# Buguri software celebre (6)

- **Naveta spațială Mars Polar Lander – 1998**

- **obiectiv:** studierea solului și a climei din regiunea Planum Australe de pe Marte;
- pentru mecanismul de identificare a momentului când motorarele trebuie să fie oprite, NASA nu a folosit radare costisitoare, ci un senzor pe talpa picioarelor navetei, care determina oprirea alimentării cu combustibil;
- **eveniment:** la intrarea în atmosfera planetei Marte, programul a interpretat vibrațiile navetei – cauzate de turbulențele din atmosferă – că aceasta ar fi aterizat și a oprit motoarele navetei;
- **rezultat:** prăbușirea navetei de înălțimea de 40m față de suprafața planetei Marte;
- **cauza:** testare incompletă – procedura de aterizare a fost împărțită în două etape, care au fost testate independent; nu s-a realizat testarea de integrare.



# Buguri software celebre (7)

- **Knight Capital Group – 2012**

- casa de brokeraj Knight Capital Group a suferit o pierdere consistentă la bursa din New York;
- **cauza:** sistemul a introdus pe bursa de la New York tranzacții care au provocat fluctuații violente ale prețurilor multor acțiuni;
- **cost:** pierderi de 440 milioane \$ în doar 45 minute.



PHOTO: STAN HONDA/REUTERS

# Buguri software celebre (8)

- **Termostatul Nest – 2016**

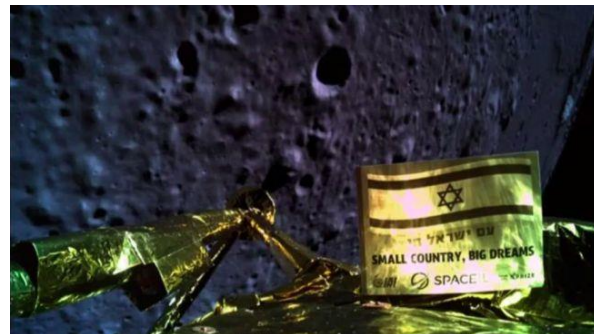
- termostatul Nest Learning Thermostat (achiziționat de Google în 2014 pentru 3.2 mld \$) nu a permis controlul temperaturii în locuințele în care a fost instalat – imposibilitatea de a-l utiliza pentru încălzire sau prepararea apei calde în timpul unui weekend friguros;
- **cauza:** update-ul de firmware pentru device împreună cu existența unor filtre necurățate și centrale termice incompatibile; acești factori au dus la descărcarea bateriei device-ului.



# Buguri software celebre (9)

- **Beresheet („In the beginning...”) – 2019**

- în 11 aprilie 2019 a avut loc tentativa eșuată a Israelului de a trimite pe Lună o naveta spațială fără oameni a bord;
- **cauza:** un bug la sistemul de control al motorului care l-a împiedicat să reducă viteza în timpul aselenizării;
- inginerii au încercat să corecteze bug-ul de la distanță prin restartarea motorului, dar la preluarea controlului asupra motorului era prea târziu pentru ca Beresheet să poată fi încetinită și s-a dezintegrat la prăbușire.



# Referințe bibliografice

- [Firesmith2015] Donald Firesmith, *Four Types of Shift Left Testing*, [https://insights.sei.cmu.edu/sei\\_blog/2015/03/four-types-of-shift-left-testing.html](https://insights.sei.cmu.edu/sei_blog/2015/03/four-types-of-shift-left-testing.html)
- [NT2005] K. Naik and P. Tripathy. *Software Testing and Quality Assurance*, Wiley Publishing, 2005.
- [NASA] NASA, <https://www.grc.nasa.gov/www/wind/valid/tutorial/glossary.html>.
- [Crosby1980] Philip B. Crosby, *Quality Is Free*, Signet Shakespeare, 1980.
- [Juran1998] A. Blanton Godfrey, Joseph Juran, *JURANS QUALITY HANDBOOK*, McGraw-Hill, 1998.
- [Weinberg1992] Gerald Weinberg, *Quality Software Management , Vol. 1: Systems Thinking*, Dorset House Publishing, 1992.
- [Pressman2000] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, Inc., 2000.
- [BBST] BBST – Bug Advocacy Course, [http://testingeducation.org/BBST/\(http://testingeducation.org/BBST/bugadvocacy/BugAdvocacy2008.pdf](http://testingeducation.org/BBST/(http://testingeducation.org/BBST/bugadvocacy/BugAdvocacy2008.pdf).
- [Patton2005] R. Patton, *Software Testing*, Sams Publishing, 2005.
- [Easterbrook2010] S. Easterbrook, *Software Testing*, <http://www.easterbrook.ca/steve/2010/11/the-difference-between-verification-and-validation/>.
- [CFI2019] Stakeholders, <https://corporatefinanceinstitute.com/resources/knowledge/finance/stakeholder/>.
- [StakeholderMap2019] Stakeholders, <https://www.stakeholdermap.com/primary-stakeholders.html>.
- [GauseWeinberg2011] Donald C. Gause, Gerald M. Weinberg, *Exploring Requirements: Quality Before Design*, Dorset House, 2011.
- [KanerBach2005] Kaner, C., Bach, J., *Requirements Analysis for Test Documentation*, <http://www.testingeducation.org/BBST/extras/BBSTTestDocs2005.pdf>.