Requirements Engineering: Setting the Scene

Leereenheid 6

Requirements Engineering: Setting the Scene

INTRODUCTIE

Bij deze leereenheid hoort hoofdstuk 1 van het tekstboek, dat het terrein afbakent.

Aan het eind van deze leereenheid geven we bovendien een korte inleiding in Systems Engineering, een onderwerp dat in dit boek op de achtergrond steeds aanwezig is.

LEERDOELEN

Na het bestuderen van deze leereenheid wordt verwacht dat u

- de betekenis kent van de termen WHY-, WHAT- en WHO-dimension
- kunt aangeven uit welke componenten een te ontwikkelen systeem (het system-to-be) is opgebouwd
- het verschil begrijpt tussen beschrijvende (descriptive) en voorschrijvende (prescriptive) uitspraken
- de betekenis begrijpt van de termen system requirement, software requirement, domain property, assumption en definition en het verband tussen deze typen uitspraken begrijpt
- weet wat monitored variables, controlled variables, input variables en output variables zijn en wat hun verband is met system en software requirements
- het verschil kunt aangeven tussen functionele en niet-functionele requirements en vier typen niet-functionele requirements kunt noemen
- enkele voorbeelden kunt geven van quality requirements
- het (spiraalvormige) verloop van het requirements proces kunt schetsen
- voorbeelden kunt noemen van eisen die aan een goed requirements document gesteld kunnen worden en gebreken die vermeden moeten worden
- kunt uitleggen waarom requirements engineering noodzakelijk is en welke obstakels er zijn voor een goed requirements engineering proces
- kunt aangeven waarin het systems engineering proces in bredere zin verschilt van softwareontwikkeling
- de verschillende stadia kunt noemen in het systems engineering process en kunt uitleggen wat hun functie is
- de betekenis kunt geven van de volgende termen: environmental phenomena, shared phenomena, software phenomena, four variable model, quality requirements, compliance requirements, architectural requirements, development requirements, systems engineering

Studeeraanwijzingen
Bestudeer hoofdstuk 1 van het tekstboek.

De studielast van deze leereenheid bedraagt circa 15 uur.

LEERKERN

1 Aanvullingen en opmerkingen

Figure 6.6 op pagina 34 van het tekstboek toont het requirements-proces als een herhaald doorlopen spiraal. Realiseer u dat deze spiraal de structuur van het eerste deel van het boek bepaalt:
– hoofdstuk 2 gaat over Domain Understanding and Elicitation
– hoofdstuk 3 gaat over Requirements Evaluation
– hoofdstuk 4 gaat over Specification and Documentation
– hoofdstuk 5 gaat over Quality Assurance
– hoofdstuk 6 gaat over Requirements Evolution, ofwel de verdere doorgangen door de spiraal

Hoofdstuk 7 is een overgang tussen het eerste en het tweede deel en valt hier buiten.

We behandelen overigens, zoals uiteengezet in de introductie-leereenheid, niet al deze hoofdstukken.

2 Opgaven

OPGAVE 6.1
Uit welke componenten is het *system-to-be* opgebouwd?

OPGAVE 6.2
Teken een wereld-machinediagram (als in figure 1.3 op pagina 18 van het tekstboek) voor een geldautomaat. Toon verschijnselen (phenomena) die te maken hebben met het uitnemen van de pas en het uittellen van het geld.

OPGAVE 6.3
De volgende uitspraken zijn afkomstig uit een systeem voor de toewijzing van ambulances (welke ambulance gaat naar een net gebeurd ongeluk). Geef voor elke uitspraak aan of het een system requirement, een software requirement, een domeineigenschap, een aanname of een definitie is.
a Binnen veertien minuten na binnenkomst van de melding van een incident bij de alarmcentrale moet er een ambulance ter plaatse zijn.
b Een ambulance kan slechts één gewonde tegelijk vervoeren.
c Een incident is een gebeurtenis waarbij minimaal één menselijke gewonde is gevallen.
d Na de invoer van een locatie van een incident zal het systeem de nummers en de gemeten locaties tonen van de vijf ambulances die volgens het systeem het dichtst bij de plaats van het incident zijn en beschikbaar zijn.
e Een incident wordt altijd binnen tien minuten gemeld bij de alarmcentrale.
f Voor elke ambulance en op elk moment wijkt de gemeten locatie niet meer dan 100 meter af van de werkelijke locatie.

OPGAVE 6.4
Wat is de categorie van de requirements uit de vorige opgave?

OPGAVE 6.5

Tekstboek pagina 59

Maak de eerste opgave op pagina 59 van het tekstboek ('Consider the case study description of the train control system….'). NB De verantwoordelijkheid voor het gesloten houden van de deuren bij de passagiers leggen, betekent dat er (zoals dat lang geleden het geval was) geen deurbeveiliging is.

OPGAVE 6.6

Tweede opgave op pagina 59 van het tekstboek (aangepast)

Noem voor het systeem voor het plannen van vergaderingen beschreven op pagina 9-12 van het tekstboek:

a twee strategische doelen (strategic objectives) van de *kopers* van dit systeem (het is ontwikkeld door een softwarehuis, zie pagina 9 van het tekstboek).

b twee functionele diensten (functional services)

c twee aannamen over de omgeving (environment assumptions).

OPGAVE 6.7

Zesde opgave op pagina 59 van het tekstboek (aangepast)

Bekijk een eenvoudig systeem met verkeerslichten om voetgangers een drukke weg te laten oversteken. Gegeven zijn een systeemeis en drie software-eisen als volgt:

(*SysReq:)* Het verkeerslicht zal voetgangers in staat stellen om veilig de weg over te steken door auto's te laten stoppen

(*Sofreq1:)* De schakelaar van het voetgangerslicht wordt op groen gezet binnen 30 seconden nadat er op de voetgangersknop is gedrukt

(*SofReq2:*) De schakelaar van het verkeerslicht voor auto's wordt op rood gezet ten minste 5 seconden voordat het voetgangerslicht groen wordt.

(*Sofreq3:*) Bij het starten van het systeem wordt de schakelaar van het voetgangerslicht op rood en die van het verkeerslicht op groen gezet.

a Welke domeineigenschappen en welke aannamen over de omgeving zijn nodig om het volgende te kunnen bewijzen:
(*SofReq1, SofReq2, SofReq3,* aannamen?, domeineigenschappen?) |= *SysReq*

b Toon dit ook aan.

c Zijn de aannamen over de omgeving realistisch?

OPGAVE 6.8

Noem minimaal vijf gewenste en vijf te vermijden eigenschappen van een requirements-document.

3 Systems engineering

In het tekstboek wordt, zoals ook al blijkt uit dit eerste hoofdstuk, het opstellen van software requirements in een veel breder kader geplaatst: het gaat bij de voorbeelden niet uitsluitend om 'zuivere' software-systemen. In de treincasus, beschreven op pagina 8-9 van het tekstboek, gaat het om een systeem waarin besturingsssoftware samenwerkt en afhankelijk is van sensoren en actuatoren in de trein en op de perrons. Agents in alle te specificeren systemen zijn niet alleen stukken software, maar mogelijk ook mensen en/of apparaten (devices) (zie pagina 16 van het tekstboek). Hetgeen waarvoor in deze cursus requirements worden opgesteld, is dus een *systeem* en niet uitsluitend een stuk software.

Er zijn enkele belangrijke verschillen tussen software ontwikkeling en systeemontwikkeling in bredere zin. Ten eerste is het nog moeilijker en duurder dan bij een zuiver softwaresysteem om de structuur van het systeem achteraf nog ingrijpend te wijzigen: hardwarecomponenten zijn nu eenmaal niet flexibel. Ten tweede zijn er bij het ontwikkelen van een hybride systeem allerlei verschillende ingenieursdisciplines betrokken, wat de kans op misverstanden en communicatiestoornissen vergroot.

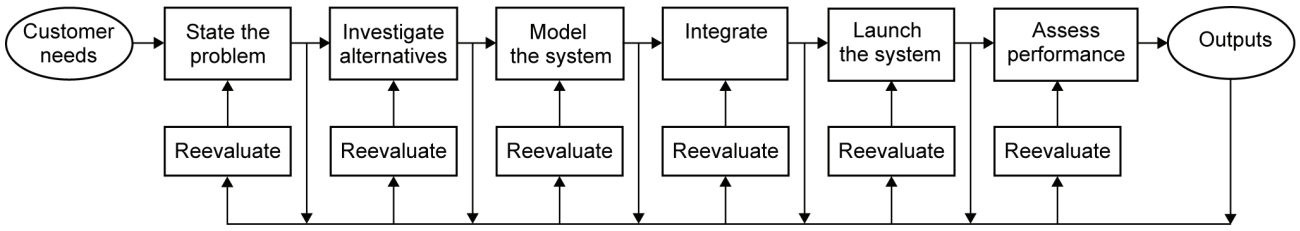
Om u bewust te maken van het feit dat de blik van een software engineer breder moet zijn dan alleen software, geven we hier een korte inleiding in het onderwerp systems engineering. De tekst is overgenomen van de website van INCOSE, The International Council on Systems Engineering (http://www.incose.org/practice/fellowsconsensus.aspx).

Definition of a system

A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected (Rechtin, 2000).

Systems Engineering

Systems Engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle. This process is usually comprised of the following seven tasks: **S**tate the problem, **I**nvestigate alternatives, **M**odel the system, **I**ntegrate, **L**aunch the system, **A**ssess performance, and **R**e-evaluate. These functions can be summarized with the acronym SIMILAR: **S**tate, **I**nvestigate, **M**odel, **I**ntegrate, **L**aunch, **A**ssess and **R**e-evaluate. This Systems Engineering Process is shown in Figure 6.1. It is important to note that the Systems Engineering Process is not sequential. The functions are performed in a parallel and iterative manner.



FIGUUR 6.1 The Systems Engineering Process from A. T. Bahill and B. Gissing, Re-evaluating systems engineering concepts using systems thinking, *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 28 (4), 516-527, 1998.

State the problem

The problem statement starts with a description of the top-level functions that the system must perform: this might be in the form of a mission statement, a concept of operations or a description of the deficiency that must be ameliorated. Most mandatory and preference requirements should be traceable to this problem statement. Acceptable systems must satisfy all the mandatory requirements. The preference requirements are traded-off to find the preferred alternatives. The problem statement should be in terms of *what* must be done, not *how* to do it. The problem statement should express the customer requirements in functional or behavioral terms. It might be composed in words or as a model. Inputs come from end users, operators, maintainers, suppliers, acquirers, owners, regulatory agencies, victims, sponsors, manufacturers and other stakeholders.

Investigate Alternatives

Alternative designs are created and are evaluated based on performance, schedule, cost and risk figures of merit. No design is likely to be best on all figures of merit, so multicriteria decision-aiding techniques should be used to reveal the preferred alternatives. This analysis should be redone whenever more data are available. For example, figures of merit should be computed initially based on estimates by the design engineers. Then, concurrently, models should be constructed and evaluated; simulation data should be derived; and prototypes should be built and measured. Finally, tests should be run on the real system. Alternatives should be judged for compliance of capability against requirements. For the design of complex systems, alternative designs reduce project risk. Investigating innovative alternatives helps clarify the problem statement.

Model the system

Models will be developed for most alternative designs. The model for the preferred alternative will be expanded and used to help manage the system throughout its entire life cycle. Many types of system models are used, such as physical analogs, analytic equations, state machines, block diagrams, functional flow diagrams, object-oriented models, computer simulations and mental models. Systems Engineering is responsible for creating a product and also a process for producing it. So, models should be constructed for both the product and the process. *Process* models allow us, for example, to study scheduling changes, create dynamic PERT charts and perform sensitivity analyses to show the effects of delaying or accelerating certain subprojects. Running the process models reveals bottlenecks and fragmented activities, reduces cost and exposes duplication of effort. *Product* models help explain the system. These models are also used in tradeoff studies and risk management.
As previously stated, the Systems Engineering Process is not sequential: it is parallel and iterative. This is another example: models must be created before alternatives can be investigated.

Integrate

No man is an island. Systems, businesses and people must be integrated so that they interact with one another. Integration means bringing things together so they work as a whole. Interfaces between subsystems must be designed. Subsystems should be defined along natural boundaries. Subsystems should be defined to minimize the amount of information to be exchanged between the subsystems. Well-designed subsystems send finished products to other subsystems. Feedback loops around individual subsystems are easier to manage than feedback loops around interconnected subsystems. Processes of co-evolving systems also need to be integrated. The consequence of integration is a system that is built and operated using efficient processes.

Launch the system

Launching the system means running the system and producing outputs. In a manufacturing environment this might mean buying commercial off the shelf hardware or software, or it might mean actually making things. Launching the system means allowing the system do what it was intended to do. This also includes the system engineering of deploying multi-site, multi-cultural systems.
This is the phase where the preferred alternative is designed in detail; the parts are built or bought (COTS), the parts are integrated and tested at various levels leading to the certified product. In parallel, the processes necessary for this are developed – where necessary - and applied so that the product can be produced. In designing and producing the product, due consideration is given to its interfaces with operators (humans, who will need to be trained) and other systems with which the product will interface. In some instances, this will cause interfaced systems to co-evolve. The process of designing and producing the system is iterative as new knowledge developed along the way can cause a re-consideration and modification of earlier steps.
The systems engineers' products are a mission statement, a requirements document including verification and validation, a description of functions and objects, figures of merit, a test plan, a drawing of system boundaries, an interface control document, a listing of deliverables, models, a sensitivity analysis, a tradeoff study, a risk analysis, a life cycle analysis and a description of the physical architecture. The requirements should be validated (Are we building the right system?) and verified (Are we building the system right?). The system functions should be mapped to the physical components. The mapping of functions to physical components can be one to one or many to one. But if one function is assigned to two or more physical components, then a mistake might have been made and it should be investigated. One valid reason for assigning a function to more than one component would be that the function is performed by one component in a certain mode and by another component in another mode. Another would be deliberate redundancy to enhance reliability, allowing one portion of the system to take on a function if another portion fails to do so.

Assess performance

Figures of merit, technical performance measures and metrics are all used to assess performance. Figures of merit are used to quantify requirements in the tradeoff studies. They usually focus on the product. Technical performance measures are used to mitigate risk during design and manufacturing. Metrics (including customer satisfaction comments, productivity, number of problem reports, or whatever you feel is critical to your business) are used to help manage a company's processes. Measurement is the key. If you cannot measure it, you cannot control it. If you cannot control it, you cannot improve it. Important resources such as weight, volume, price, communications bandwidth and power consumption should be managed. Each subsystem is allocated a portion of the total budget and the project manger is allocated a reserve. These resource budgets are managed throughout the system life cycle.

Re-evaluate

Re-evaluate is arguably the most important of these functions. For a century, engineers have used feedback to help control systems and improve performance. It is one of the most fundamental engineering tools. Re-evaluation should be a continual process with many parallel loops. Re-evaluate means observing outputs and using this information to modify the system, the inputs, the product or the process. Figure 1 summarizes the Systems Engineering Process. This figure clearly shows the distributed nature of the Re-evaluate function in the feedback loops. However, all of these loops will not always be used. The particular loops that are used depend on the particular problem being solved.

Variations

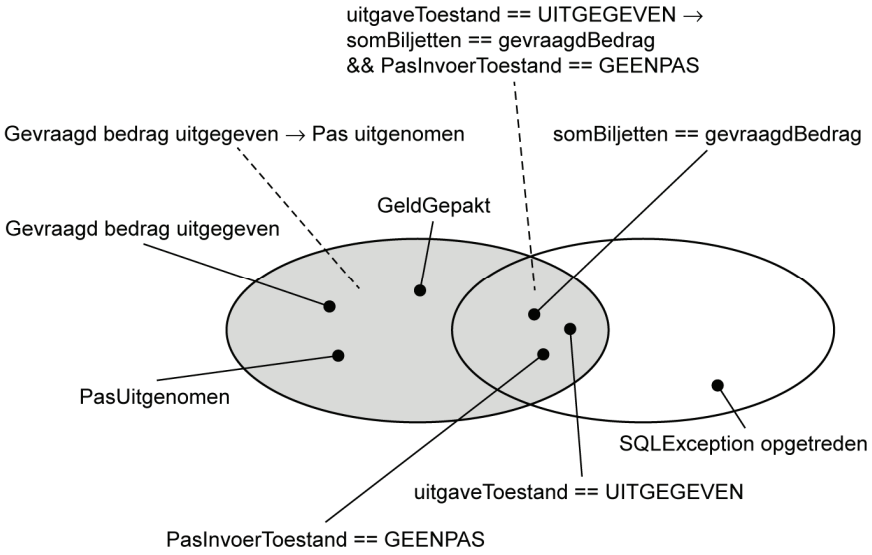
Like all processes, the Systems Engineering process at any company should be documented, measurable, stable, of low variability, used the same way by all, adaptive, and tailorable! This may seem like a contradiction. And perhaps it is. But one size does not fit all. The above description of the Systems Engineering process is just one of many that have been proposed. Some are bigger, some are smaller. But most are similar to this one.

TERUGKOPPELING

Uitwerking van de opgaven

6.1 Het *system-to-be* is opgebouwd uit de *software-to-be* (het te bouwen softwaresysteem) en componenten die samen de omgeving van die software vormen. Tot die omgeving behoren mensen, fysieke apparaten als sensoren en actuatoren en tot slot bestaande software waarmee het systeem moet communiceren.

6.2 Figuur 6.2 toont een mogelijke uitwerking.



FIGUUR 6.2 Wereld-machinediagram voor gelduitgifte door geldautomaat

Het belangrijke punt is, dat de software geen directe manier heeft om vast te stellen of de pas nog de automaat zit en of het gevraagde bedrag naar de gelduitvoer is getransporteerd. De software moet daarvoor afgaan op informatie afkomstig van sensoren. De eis "Gevraagd bedrag uitgegeven → Pas uitgenomen" is daarom een systeemeis, terwijl de eis "uitgaveToestand == UITGEGEVEN → somBiljetten == gevraagdBedrag & PasInvoerToestand == GEENPAS" software requirements zijn.

6.3 a Dit is een system requirement, een eis aan het systeem als geheel en niet alleen aan de toewijzingssoftware.
b Dit is een domeineigenschap (de waarheid ervan hangt niet van het systeem af maar alleen van de omgeving).
c Dit is een definitie (er is geen waarheidswaarde).
d Dit is een software requirement; het geeft een relatie tussen invoer (melding van de locatie van een incident) en uitvoer (de locaties van ambulances). Merk op dat het hier gaat om locaties en beschikbaarheid zoals bepaald door het systeem; dat dit klopt met de werkelijkheid moet apart gespecificeerd worden.
e Dit is een aanname, een eis waarvan we verwachten dat de omgeving er aan voldoet.

f Vanuit het systeem voor de toewijzing van ambulances gezien, is dit een aanname en wel een accuracy statement. Deze aanname is overigens verre van triviaal. In de totaal mislukte invoering van het London Ambulance System in 1992 werden de locaties van de ambulances bijgehouden door een autovolgsysteem, dat volstrekt onvoldoende functioneerde. De gemeten locatie week daardoor vaak kilometers af van de werkelijke locatie.

6.4 De system requirement uit onderdeel a is een performance requirement. De software requirement uit onderdeel d is een functionele requirement.

6.5 Zie onderstaande tabel.

	voordelen	*risico's*
bestuurder	Duidelijk wie verantwoordelijk is	Bestuurder ziet deuren niet en kan dus niet zeker weten of ze dicht zijn Bestuurder heeft ook andere taken, wat het risico van menselijk falen verhoogt
conducteur	Duidelijk wie verantwoordelijk is. Conducteur kan alle deuren langsgaan en ze van buiten sluiten.	Risico van menselijk falen. Het sluiten van de deuren neemt veel tijd in beslag.
passagiers	Eenvoudig en goedkoop.	Passagiers met haast openen de deur terwijl de trein nog (of al) rijdt, wat tot ernstige ongelukken kan leiden, met de bijbehorende hoge claims voor de vervoersmaatschappij.
software	Indien correct, zeer betrouwbaar	Softwarefout Fout in detectie of besturing

6.6 a Strategische doelen voor de kopers van het vergaderplannings-systeem zijn bijvoorbeeld:
– De deelname aan vergaderingen vergroten.
– Organisatorische overhead bij het plannen van vergaderingen terugdringen.
De andere doelen genoemd op pagina 10 en 11 zijn daarvan afgeleid. Het belang van een snelle scheduling dient bijvoorbeeld vooral om het risico terug te dringen dat mensen inmiddels al weer bezet zijn, en het terugbrengen van het aantal benodigde berichten is een zaak van efficiency.
b Enkele voorbeelden van services:
– Vergaderverzoek sturen aan alle potentiële deelnemers.
– Feschikte vergaderlocatie bepalen op grond van de deelnemers en hun standplaatsen.
– Gegeven een verzameling constraints, een tijdvak waarbinnen de vergadering moet vallen en de duur van de vergadering, een mogelijk tijdstip bepalen voor de vergadering.
– Deelnemers de tijd en locatie van de vergadering meedelen.
c Enkele voorbeelden van environment assumptions:
– Deelnemers reageren binnen n dagen op een vergaderverzoek.
– Een tijd die is opgegeven als beschikbaar, blijft minimaal m dagen (of uren) beschikbaar.
– Deelnemers kunnen op een dag niet meer dan x km reizen.

6.7 a Een mogelijke uitwerking is de volgende.

Domeineigenschappen:
– D1: De schakelaar van een verkeerslicht heeft twee standen: rood en groen.
– D2: Een oversteek van een weg is veilig als er geen auto's op de weg rijden.
– D3: Een auto die stilstaat, rijdt niet op de weg.
Aannamen over de omgeving:
– O1 (O2): Een verkeerslicht is rood (groen) dan en slechts dan als schakelaar van het licht op rood (groen) staat.
– O3: Voetgangers die willen oversteken op een moment dat het voetgangerslicht rood is, drukken op de voetgangersknop.
– O3: Voetgangers steken over als het voetgangerslicht groen is en steken niet over als het rood is.
– O4: Automobilisten rijden door als het verkeerslicht groen is.
– O5: Automobilisten stoppen als ze de oversteekplaats naderen terwijl het verkeerslicht al vijf seconden of langer rood is.

b Bewijs:
– Volgens Sofreq3 staat de schakelaar van het voetgangerslicht in de beginsituatie op rood. Volgens O1 is dan ook het voetgangerslicht rood en volgens O3 steken voetgangers dan niet over. Deze situatie is dus veilig.
– Volgens Sofreq1 wordt de schakelaar van het voetgangerslicht op groen gezet hoogstens 30 seconden nadat er op de knop is gedrukt; het licht wordt dan groen (O2) en de voetgangers steken over (O3).
– Volgens Sofreq2 is de schakelaar van het verkeerslicht al 5 seconden eerder op rood gezet en dus is volgens O1 ook het verkeerslicht al 5 seconden eerder op rood gegaan.
– Volgens O5 stoppen alle auto's dan bij het verkeerslicht.
– Volgens D3 rijden ze dan niet op de weg.
– Volgens D2 is de oversteek dan dus veilig.

Merk op, dat volgens deze specificatie het verkeerslicht mogelijk altijd rood blijft; daar spreken de software requirements zich niet over uit.
In het boek ontbreekt de derde requirement die de beginsituatie beschrijft. Dat zou echter kunnen betekenen dat beide lichten op groen staan bij de start van het systeem en dat is niet veilig.
Ook zeggen de eisen niets over andere weggebruikers; misschien is het realistischer om het overal over weggebruikers te hebben in plaats van over automobilisten.

c Uit de praktijk weten we natuurlijk dat niet alle voetgangers wachten tot het licht groen is en niet alle automobilisten stoppen voor een rood licht.

6.8 Zie de opsommingen op pagina 35-36 respectievelijk 37 van het tekstboek.