Analisi dei requisiti

Anno accademico 2023/2024

Ingegneria del Software

Tullio Vardanega, tullio.vardanega@unipd.it

Glossario

- **Secondo il glossario IEEE, "requisito" è**
  1. **La capacità necessaria a un utente per risolvere un problema o raggiungere un obiettivo: lato bisogno**
  2. **La capacità necessaria a un sistema per rispondere a una aspettativa: lato soluzione**
- **Il capitolato riflette il punto di vista di *lato bisogno***
  - **Ciò che il proponente si aspetta: «requisiti utente»**
  - **Anche laddove si addentri in raccomandazioni tecniche**
- **L'analisi dei requisiti esplora il punto di vista di *lato soluzione***
  - **Ciò che il prodotto deve saper fare per soddisfare i bisogni: «requisiti *software*»**

Acquisizione e fornitura

- **Nel processo di acquisizione, il committente studia un problema di proprio interesse e fissa le proprie aspettative su un prodotto che lo risolva**
  - **Fissando le aspettative in un capitolato che specifica i «requisiti utente»**
- **Nel processo di fornitura, il fornitore deve capire come soddisfare quelle richieste**
  - **Valutandone la fattibilità (tecnico-economica), insieme al proprio interesse strategico**
- **La prima *milestone* del calendario di progetto è quando committente e fornitore trovano accordo su**
  - **Gli obiettivi funzionali (analisi dei requisiti), gli obiettivi di qualità (piano di qualifica), i tempi e i costi del progetto (piano di progetto)**
  - **Nel mondo reale, questo è il vero momento di aggiudicazione**
- **Il contratto termina con la valutazione del committente sul prodotto**
  - **Accettazione condizionata a buon esito di collaudo**

Punto di partenza

- **I requisiti riflettono il contesto di lato utente, nel passaggio da *senza* a *con* il prodotto**
  - **Cosa voglio poter fare che prima non potevo**
  - **Capire il «senza» (prima) per capire il «con» (dopo)**
  - **Nel dominio d'uso di interesse**
- **L'analisi dei requisiti riflette la struttura funzionale del prodotto (non quella architetturale)**
  - **Lo scenario «Premere un pulsante scatena un calcolo che legge o scrive dati» va studiato come gerarchia, dall'esterno all'interno: che cosa deve succedere**
- **Sempre dal punto di vista di chi o cosa scateni l'azione in esame (per questo, detto «attore»)**
  - **L'attore è chi o cosa possa fare richieste alla parte di prodotto alla quale l'attore è esposto (sistema)**

Avvertenze

- L'analisi dei requisiti deve comprendere a fondo i bisogni
  - Cosa deve succedere per rispondere alle richieste fatte al prodotto
- I diagrammi dei casi d'uso hanno proprio questa prospettiva
  - Relazione tra «attore» e «sistema»
  - Cosa posso chiedere, cosa deve succedere
- Le «*user story*» sono il corrispondente XP di quei diagrammi
- L'analisi non deve costruire l'algoritmo di soluzione
  - Questo spetta alla progettazione
- L'analisi deve essere consapevole della fattibilità tecnologica

Fattori di rischio

- Le principali cause di abbandono di un progetto secondo (Standish Group 1995)
  - Requisiti incompleti
  - Insufficiente coinvolgimento del cliente/utente
  - Attese irrealistiche
  - Scarsità di risorse rispetto all'ampiezza del problema
  - Volatilità di specifiche e requisiti
  - Insufficiente competenza del fornitore
- Molte di queste cause originano da analisi dei requisiti difettose

Punto di arrivo

- Culmine del progetto è il collaudo del prodotto (validazione)
  - Il fornitore dimostra che tutti i requisiti utente siano soddisfatti
  - Accertando (verifica) che lo sviluppo abbia preso in carico tutti i requisiti di lato soluzione, senza dimenticarne alcuno
- Il «tracciamento dei requisiti» è la tecnica base per assicurarlo
- L'analisi dei requisiti deve facilitare tracciamento e verifica
- Chi fissa un requisito di lato soluzione deve anche stabilire come accertarne il soddisfacimento
  - Il metodo *Test-Driven Development* porta questo principio all'estremo
- Facendo attenzione al costo e complessità della verifica

Glossario

- **Verifica**
  - Accertare che lo svolgimento delle attività di sviluppo non introduca errori
  - *Did I build the system right?* - attenzione rivolta al *way of working*
- **Validazione**
  - Accertare che il prodotto corrisponda alle attese
  - *Did I build the right system?* - attenzione rivolta al prodotto finale
- **Piano di qualifica**
  - Dire come svolgeremo le attività di **V&V** nel progetto
  - E con quali obiettivi di qualità

V & V = Qualifica

Tracciamento dei requisiti

Esempio di tracciamento

Necessità: ogni requisito *software* soddisfa uno specifico requisito utente
(tutti i requisiti *software* sono utili)
Sufficienza: tutti i bisogni rilevanti corrispondono a qualche requisito *software*
(nessun bisogno dimenticato)

Attività di analisi dei requisiti

- Studio dei bisogni e del dominio d'uso
- Comprensione del problema dal lato dei bisogni
  - Approfondimenti tramite scenari di caso d'uso
- Raggruppamento degli scenari per affinità
  - Per individuare possibili parti del sistema
  - Approfondendole gerarchicamente
- Valutazione di fattibilità tecnologia
  - Noi la chiamiamo *Technology Baseline*
- Classificazione e tracciamento dei requisiti
  - In dialogo con il committente e gli altri *stakeholder*

Processi di supporto all'analisi

- **Documentazione**
  - **Per raccogliere i risultati dell'analisi**
  - **Per dialogare con il committente**
  - **Per avviare consapevolmente la progettazione (*design*)**
- **Verifica di adesione al *way of working***
- **Gestione e manutenzione dei prodotti dell'analisi**
  - **Tracciamento dei requisiti**
  - **Gestione di versione e configurazione**
  - **Gestione dei cambiamenti**

Per una buona analisi – 1/2

- La specifica dei requisiti deve essere
  - Priva di ambiguità
  - Corretta
  - Completa
  - Verificabile
  - Consistente
  - Modificabile
  - Tracciabile
  - Ordinata per rilevanza

IEEE 830-1998: *Recommended Practice for SW Requirements Specification*

- Per questo conviene ridurre i requisiti a «grana fine»
  - Affinando, e suddividendo i requisiti utente, aggiungendone ampliando lo sguardo sul problema
- Su questo poi agiscono verifica e controllo di qualità

Per una buona analisi – 2/2

GOOD REQUIREMENTS CHECKLIST

COMMENTARY BELOW THE BULLETED CHECKLIST ITEMS MAY GUIDE REQUIREMENTS IMPROVEMENTS EFFORTS

The following 10-item list (with supplemental notes) serves to assess areas where requirements quality may be improved, guide discussions among stakeholders, and—only if absolutely necessary—produce a metric to report to management. The statements below apply to both individual requirements and the specification as a whole.

- *Complete*: Contains sufficient information to drive forward the work of those who need to use the information at this point in the lifecycle, at an acceptable level of risk.

  "Completeness" cannot be a single binary toggle and is not merely a reflection of whether all fields in a template are filled in but must be assessed continually, based on the changing needs of both the project and developers over time. Who are the consumers of the artifacts you create? What do they need today to do their work, and what do you know today about the substance of the artifacts? How will your knowledge and their needs progress together across the lifecycle to produce the highest-quality, most useful content at precisely the right time?

- *Correct*: Stakeholder and SME reviews locate no errors, content is consistent with source materials, and artifacts have been reviewed and accepted by appropriate constituents.

  Who gets to decide if requirements are "done"? I ask this in every class I teach at work, usually to a few moments of silence before someone speaks up: "Maybe the person who has to actually use them?" Yes—consumers of the requirements are the ultimate arbiters of requirements. But see also *Traceable*, where we move a step beyond mere evidence that a requirement can be traced to source material.

- *Concise*: Addresses a single idea or concept, expressed in as few words as possible. Supporting information is separate from the requirement statement itself.

  The advent of commercially available natural language processing (NLP) requirements tools has made assessment of a statement's conciseness (also called "atomic" by some checklists) somewhat simpler. Mere mechanical assessment of conjunctions or verb counts provide insight, but what else can be explored through a discussion of the intent behind the statement? Does the level of abstraction of the requirement matter?

- *Feasible*: Requirements are known to be feasible through use in prior products, prototyping, or analysis.

  I have the privilege of working with some of the most brilliant people I've encountered anywhere in the world, people who take "not possible" as a challenge to make it happen anyway. Propose an idea to them and ask if it is feasible, and the answer will invariably be "Yes." Nothing's impossible, given enough time and resources—but time and resources are finite, so bound the discussion and, perhaps, discuss how to stage steps toward the end goal, particularly if it currently appears to violate all known practice.

- *Necessary*: Market segment, business strategy, usage model, technical feasibility, or legal or sustainability constraints require that this statement exist.

  Separating needs from wants is always a challenge. There are no free features—any requirement artifact is a claim on planning, architecture, development, test, and other resources, human as well as financial. Do all stakeholders share the same vision of product needs?

- *Prioritized*: Tradeoffs between requirements are clear, explicit, and understood by stakeholders, and they have been assessed across multiple dimensions of market and technical considerations.

  Some programs may find value in a strict high–medium–low prioritization scheme; others may simply

(Continued on next page)

Editor: Sarah Gregory
Intel Corporation
sarah.c.gregory@ieee.org

GOOD REQUIREMENTS CHECKLIST (CONT.)

COMMENTARY BELOW THE BULLETED CHECKLIST ITEMS MAY GUIDE REQUIREMENTS IMPROVEMENTS EFFORTS

assess whether requirements are present or absent for a particular variant—included, or left in a backlog to address later—if at all. More significant than whether a priority has been assigned to a requirement may be the question of whether prioritization is assessed frequently through a process of backlog grooming, ensuring that what is done at any particular time is the highest-value, greatest-need work.

- *Unambiguous*: Each requirement is well understood by the intended audience and has only one possible interpretation.

  NLP tools also shine at identifying ambiguity in requirements. However, just as requirements prioritization is not an objective science, neither is the effect of ambiguity, as increasing recent research suggests. If a set of requirements all contain the canonically ambiguous verb "support," must they all be revised? The only reasonable answer is, "It depends." Does the risk posed by the ambiguity of "support" exceed the risk of taking the time to explore a better verb to express a concept that may be well understood and even already coded and tested on other projects? Sometimes yes—but not always.

- *Verifiable*: Implementation of each requirement can be effectively established prior to product release, and qualitative requirements are quantified with a target value and, where applicable, a description of the scale and meter with which they will be measured.

  If you can't test it, how will you know if you've met it? Agile provides one possible answer in the form of a product owner or other empowered customer representative who can raise the "good enough" flag. Particularly in contexts in which tools and work process support extensive reuse, tailoring the target value while leaving the core of the requirement itself alone can accelerate progress.

- *Consistent*: Each requirement is represented only once in a specification and is internally and externally consistent with all others.

  In a complex, multicustomer system-of-systems context, we are exploring the concept that a colleague has dubbed "purposeful duplication." Different stakeholders may have the same requirements or different ones and, likely, a combination of each. Maintaining the tension of the inconsistency of asks while assessing their overlaps and commonality as well is a delicate balancing act, and it may also require some acceptance of inconsistency until the last responsible moment for a decision to be made about a release, which may be far beyond our traditional comfort zones. This tension can only be maintained with careful and well-executed traceability.

- *Traceable*: Requirements are traceable if they are concise (only one concept per requirement, please) and uniquely and persistently identified with a tool-generated or other identifier.

  However, mere traceability of requirements is insufficient to truly satisfy this attribute in a context that measures quality as "quality in use." On the programs with which I work now, we are beginning to assess the quality of a requirement's traceability if it is, in fact, in use and traced. Why is this artifact here? What need does it satisfy, and is that need itself traced to a higher-order request that includes a rationale for its claim on resources? Have the consumers of that requirement signed off on its fitness for purpose?

Tecniche di analisi

- Dominio d'uso come fonte di requisiti impliciti
  - A quali bisogni risponde il prodotto atteso?
  - Per che tipo di utenti, con quali aspettative?
- Vi sono molti requisiti «nascosti»
  - La classificazione dei requisiti aiuta a scovarli
- Acquisizione di conoscenze: interviste al committente
  - Costruzione, analisi e discussione di scenari: decisioni riassunte in verbali
- Acquisizione di conoscenze: *brainstorming*
- Consolidamento delle conoscenze di dominio: glossario
- Valutazione di fattibilità tecnologica (TB)

Classificazione dei requisiti – 1/2

Extra-functional requirement

M. Glinz,
On Non-Functional Requirements.
15th IEEE International Conference on Requirements Engineering.
October 2007, 21-26

Classificazione dei requisiti – 2/2

- **I requisiti hanno diversa rilevanza e utilità, che va negoziata e concordata con il committente**
  - **Obbligatori**
    - Irrinunciabili per qualcuno degli *stakeholder*
  - **Desiderabili**
    - Non strettamente necessari ma a valore aggiunto riconoscibile
  - **Opzionali**
    - Relativamente utili oppure contrattabili più avanti nel progetto
- **Non devono essere tra loro contraddittori**

Confine tra analisi e *design* – 1/3

Confine tra analisi e *design* – 2/3

©Ian Sommerville 2004 Software Engineering, 7th edition

Confine tra analisi e *design* – 3/3

- La separazione netta tra analisi e *design* si ha solo con lo sviluppo sequenziale
- In ogni altro caso le due attività evolvono in parallelo, in dialogo tra loro
  - Noi le facciamo avvicinare nella *Requirements and Technology Baseline*
- La prassi XP suggerisce che il *design* evolva per fattorizzazioni successive
- Perseguendo sempre semplicità nella soluzione
  - Un *design* semplice richiede poco codice (il meno possibile)
  - Questo è un ottimo indicatore di (assenza di) semplicità

Documentazione dell'analisi – 1/3

- La documentazione in linguaggio naturale genera rischi di ambiguità interpretativa
  - Servono norme redazionali per evitare espressioni ambigue
  - Glossario per garantire terminologia consistente
- L'uso di metodi (semi-)formali aiuta a ridurre gli errori di interpretazione
  - Diagrammi e formule (UML) invece di testo e disegni in stile libero
  - *User story* strutturate

Documentazione dell'analisi – 2/3

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Requirements
- 3.1 Functions
- 3.2 Performance requirements
- 3.3 Usability requirements
- 3.4 Interface requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

ISO/IEC/IEEE 29148:2018
System and software engineering – Life cycle processes – Requirements engineering

Example outline of Software Requirements Specification

Documentazione dell'analisi – 3/3

- **Ricercare chiarezza espressiva**
  - L'uso del linguaggio naturale rende difficile coniugare chiarezza con facilità di lettura
- **Ricercare chiarezza strutturale**
  - Separazione tra requisiti funzionali e non-funzionali
  - Classificazione precisa, uniforme e accurata
- **Ricercare atomicità e aggregazione**
  - Requisiti elementari
  - Correlazioni chiare ed esplicite

Verifica dell'analisi

- **Eseguita su un documento organizzato**
  - **Non ripete il lavoro di analisi, ma si accerta che esso sia stato svolto in modo conforme alle attese**
- **Tramite *walkthrough***
  - **Lettura a largo spettro**
- **Oppure ispezione**
  - **Lettura mirata e strutturata basata su liste di controllo (*checklist*)**
- **Accertando necessità e sufficienza dei requisiti specificati**
  - **Esaminando la matrice delle dipendenze (documentazione di tracciamento)**

Stati di progresso per SEMAT – 1/2

- ***Conceived***
  - **Il committente è identificato e gli *stakeholder* vedono sufficienti opportunità per il progetto**
- ***Bounded***
  - **I bisogni macro sono chiari, i meccanismi di gestione dei requisiti (configurazione e cambiamento) sono fissati**
- ***Coherent***
  - **I requisiti sono classificati e quelli essenziali (obbligatori) sono chiari e ben definiti**

Stati di progresso per SEMAT – 2/2

- ***Acceptable***
  - I requisiti fissati definiscono un sistema soddisfacente per gli *stakeholder*
- ***Addressed***
  - Il prodotto soddisfa i principali requisiti al punto da poter meritare rilascio e uso
- ***Fulfilled***
  - Il prodotto soddisfa abbastanza requisiti da meritare la piena approvazione degli *stakeholder*