Session #3

SOFTWARE ENGINEERING ANALYSIS

Software Engineering in Practical Approach: an Experience Sharing

Jurusan Teknik Informatika Politeknik Elektronika Negeri

ITS SURABAYA, Sept., 20-22, 2011

BAHTIAR H. SUHESTA

IT Practitioner, Writer-preneur, and Founder of An-Nabwah Group

ITS Institut Teknologi Sepuluh Nopember

ANNABWAH

Topics

- Chaos Report on Software Development
- The Rock Problem
- Requirement Engineering
  - Elicitation
  - Specification
  - Validation and Verification
- Communication Principles

“

The journey of a thousand miles begins with one step, Great acts are made up of small deeds.

~ Lao Tzu

Bridge vs Software

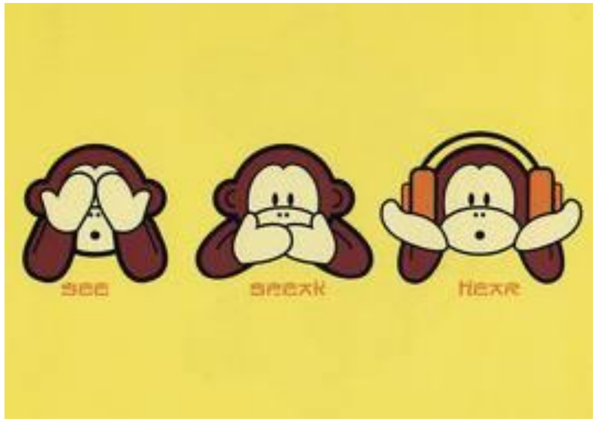
- When a bridge falls down, it is investigated and a report is written on the cause of the failure.
- In the computer industry, failures are covered up, ignored, and/or rationalized. As a result, we keep making the same mistakes over and over again.

Failure Record

Chaos Report of Standish Group, @1995

- In USA, there are approx. 175,000 projects on IT app development for more than $250 billion per year.
- The average cost for large company is $2,3 million.
- Great many of these projects will fail.
- SW development project are in chaos, and we can no longer imitate the three monkeys– hear no failures, see no failures, and speak no failures.

The Three Monkeys



Failure Record

Chaos Report of Standish Group, @1995

- The Standish Group research shows a staggering 31.1% of projects will be cancelled before they ever get completed → cancel
- Further results indicate 52.7% of projects will cost 189% of their original estimates → biaya membengkak
- The average overrun is 222% of the original time estimate. → waktu molor
- The success rate was only 16.2% → berhasil

Project Success Factors

Chaos Report of Standish Group, @1995

Project Success Factors	% of Responses
1. **User Involvement**	15.9%
2. **Executive Management Support**	13.9%
3. **Clear Statement of Requirements**	13.0%
4. Proper Planning	9.6%
5. Realistic Expectations	8.2%
6. Smaller Project Milestones	7.7%
7. Competent Staff	7.2%
8. Ownership	5.3%
9. Clear Vision & Objectives	2.9%
10. Hard-Working, Focused Staff	2.4%
Other	13.9%

Project Challenged Factors

Chaos Report of Standish Group, @1995

Project Challenged Factors	% of Responses
1. **Lack of User Input**	12.8%
2. **Incomplete Requirements & Specifications**	12.3%
3. **Changing Requirements & Specifications**	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

Project Impaired Factors

Faktor-Faktor Pengganggu Proyek

Chaos Report of Standish Group, @1995

Project Impaired Factors	% of Responses
1. **Incomplete Requirements**	13.1%
2. **Lack of User Involvement**	12.4%
3. **Lack of Resources**	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specifications	8.7%
7. Lack of Planning	8.1%
8. Didn't Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

Requirement Engineering

- *Requirements engineering* adalah fase terdepan dari proses software engineering, dimana software requirements (kebutuhan) dari user (pengguna) dan kustomer (pelanggan) dikumpulkan, dipahami, dan ditetapkan.
- Para pakar software engineering sepakat bahwa requirements engineering adalah suatu pekerjaan awal yang sangat penting.
- Kebanyakan kegagalan pengembangan software disebabkan karena adaya ketidakkonsistenan (*inconsistent), ketidaklengkapan (incomplete), maupun ketidakbenaran (incorrect)* dari spesifikasi kebutuhan (*requirements specification*).

Requirement Engineering

- The Standish Group mencatat bahwa persentase akumulatif kegagalan sebuah proyek pengembangan software sebagian besar disebabkan oleh masalah requirements dan spesifikasinya.
- Ed Yourdon menggunakan istilah *"the rock problem" (masalah batu) sebagai diskusi dasar masalah yang selalu muncul* dalam proses pengerjaan proyek software.

The Rock Problem

(Senada: “Yes, But Syndrome” / “Undiscovered Ruins Syndrome”)

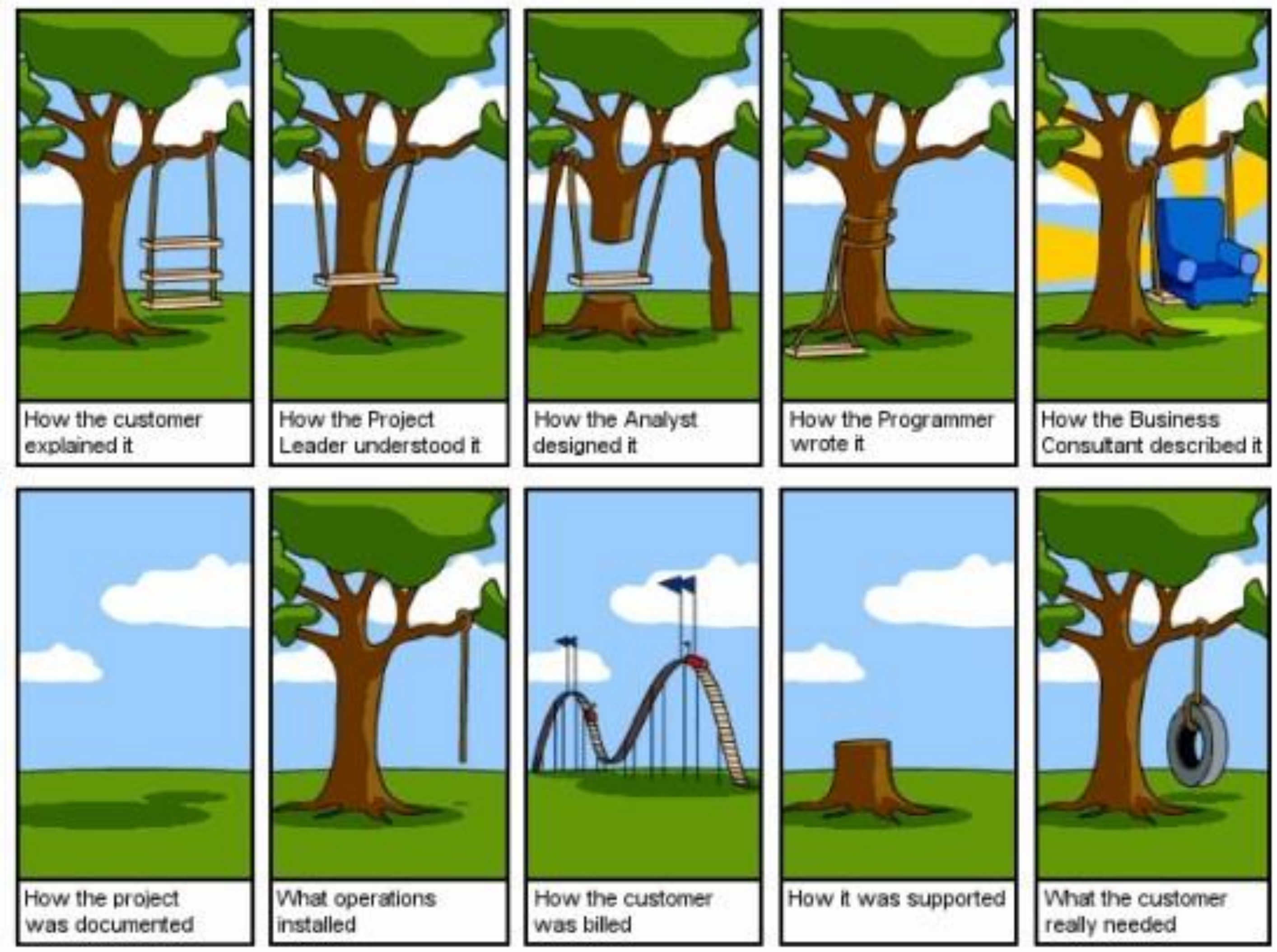
- Kustomer yang datang kepada kita ibarat mengatakan, “*Tolong buatkan saya batu*”.
- Ketika kita memberikan kepadanya sebuah batu, dia akan melihatnya sebentar dan mengatakan kepada kita, “*Ya, terima kasih, tapi sebenarnya yang saya inginkan adalah sebuah batu kecil berwarna biru*”.
- Ketika kita bawakan untuknya batu kecil berwarna biru, dia mengatakan bahwa yang diinginkan adalah “yang bentuknya bulat”.
- Demikian seterusnya proses iterasi (*iteration*) terjadi berulangkali sampai akhirnya kita dapatkan yang sebenarnya diinginkan kustomer adalah “batu pualam kecil berwarna biru berbentuk bulat telur”.

The Rock Problem

(Senada: “Yes, But Syndrome” / “Undiscovered Ruins Syndrome”)

- Meskipun mungkin sebenarnya bukan “tepat yang diinginkan”, tapi paling tidak “paling dekat” dengan yang diinginkan kustomer.
- Mungkin saja terjadi, kustomer kita mengubah pikiran tentang requirements pada saat proses interaksi dengan pengembang terjadi (dari iterasi pertama yang sekedar batu, sampai iterasi terakhir yang menghasilkan batu pualam kecil berwarna biru berbentuk bulat).

The Rock Problem


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Requirement Engineering

- Requirements engineering provides the appropriate mechanism for:
  - understanding what the customer wants,
  - analyzing need,
  - assessing feasibility,
  - negotiating a reasonable solution,
  - specifying the solution unambiguously,
  - validating the specification, and
  - managing the requirements as they are transformed into an operational system

Requirement Engineering

- Hasil dari fase requirements engineering terdokumentasi dalam *software requirements specification*.
- Requirements specification
  - berisi kesepakatan bersama tentang permasalahan yang ingin dipecahkan antara pengembang dan kustomer, dan
  - merupakan titik start menuju proses berikutnya yaitu *software design*.

Types of requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
  - A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Functional and non-functional requirements

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements
  - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain.

Functional requirements

- Fungsi teknis dari perangkat lunak yang akan dikembangkan
- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

Non-functional requirements

- Persyaratan yang bersifat kualitatif terhadap sistem atau perangkat lunak yang akan dikembangkan
- Biasanya mencakup batasan waktu, batasan proses pengembangan, penggunaan standar, dsb
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Domain Requirements

- Berasal dari domain aplikasi sistem.
- Dalam kasus sistem perpustakaan, misalnya karena masalah hak cipta maka beberapa dokumen dalam perpustakaan tidak boleh diakses oleh orang lain yang tidak berhak.
- Requirements ini bisa berupa functional maupun non-functional requirements

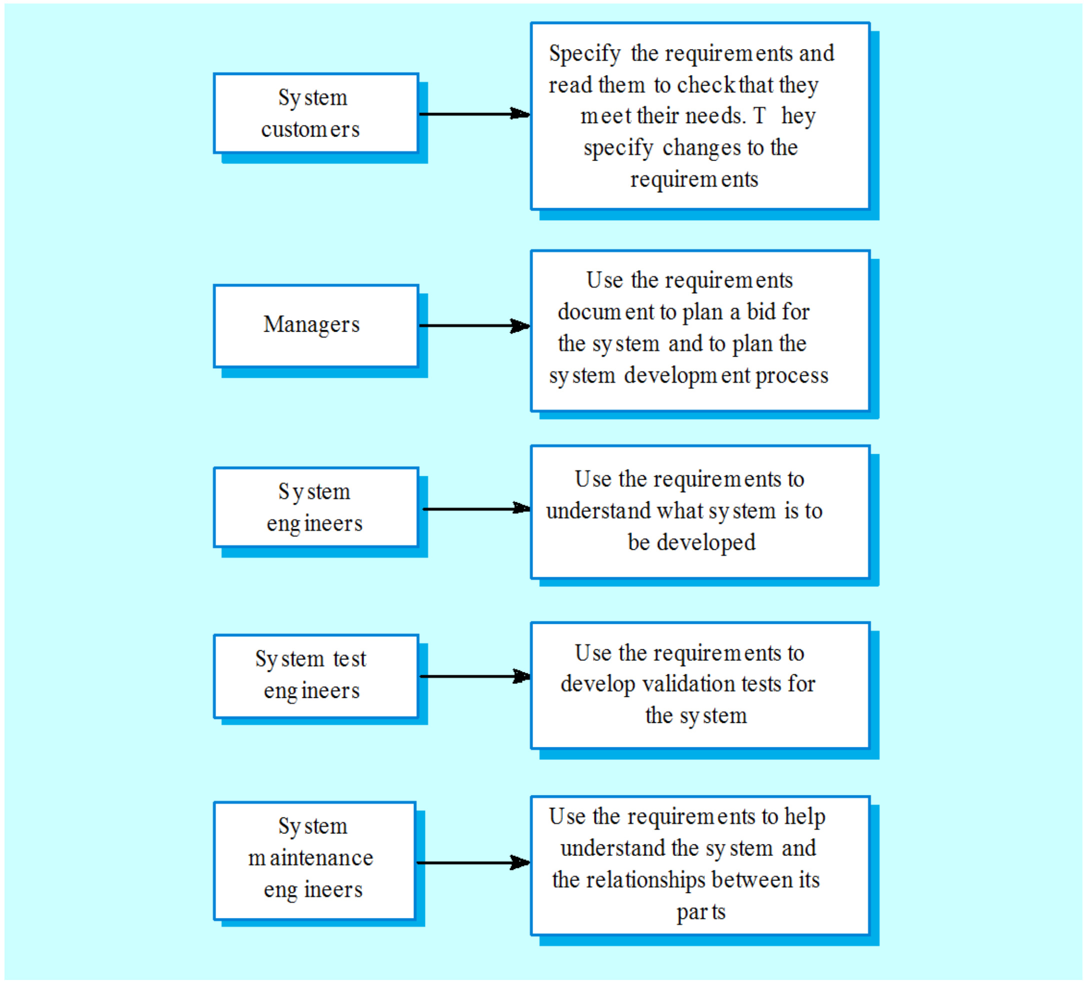
Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

Requirements interaction

- Conflicts between different non-functional requirements are common in complex systems.
- Spacecraft system
  - To minimise weight, the number of separate chips in the system should be minimised.
  - To minimise power consumption, lower power chips should be used.
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

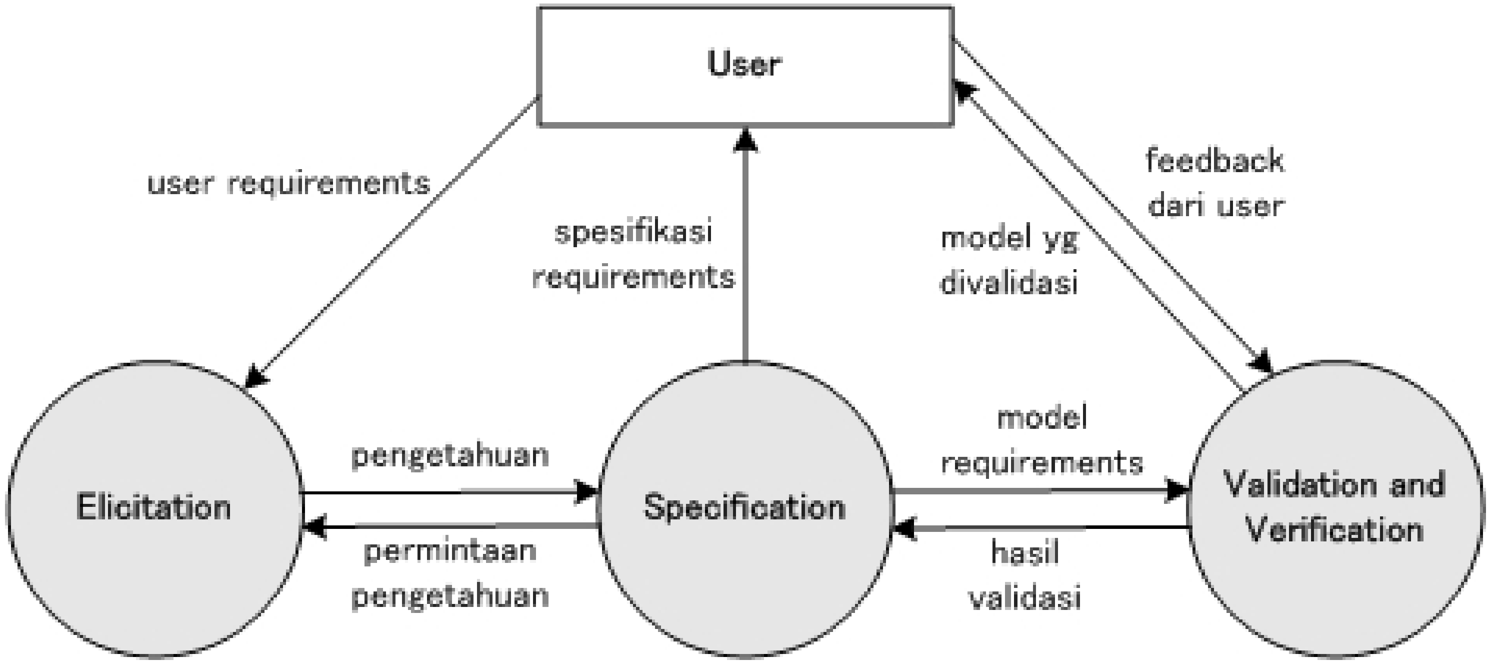
Users of a requirements document


System customers
Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements
Managers
Use the requirements document to plan a bid for the system and to plan the system development process
System engineers
Use the requirements to understand what system is to be developed
System test engineers
Use the requirements to develop validation tests for the system
System maintenance engineers
Use the requirements to help understand the system and the relationships between its parts

Requirements Engineering

- Frustasi terjadi baik di pihak pengembang maupun customer → The rock problem & Yes/but syndrome
- Dari kesulitan itu, muncul keinginan untuk menerapkan pendekatan engineering (engineering approach) untuk memecahkan masalah tersebut, yang akhirnya membawa arus deras kemunculan cabang ilmu requirements engineering.

3 Dimension of Req. Engineering



Requirement Elicitation

- Proses mengumpulkan dan memahami requirements dari user.
- Kustomer expert pada domain yang softwarenya ingin dikembangkan (domain specialist), di lain pihak requirements analyst relatif buta terhadap knowledge domain tersebut → gap knowledge.
- Gap knowledge bisa diatasi dengan adanya interaksi terus menerus dan berulang (iterasi) antara analyst dan kustomer.
- Proses interaksi tersebut kemudian dimodelkan menjadi beberapa teknik dan metodologi di antaranya adalah interviewing, brainstorming, prototyping, use case, dsb.

Requirement Specification

- Pasca masalah berhasil dipahami, analyst mendeskripsikannya dalam bentuk dokumen spesifikasi requirement.
- Spesifikasi ini berisi fitur dan fungsi yang diinginkan oleh kustomer, dan sama sekali tidak membahas bagaimana metode pengembangannya.
- IEEE mengeluarkan standard untuk dokumen spesifikasi requirements yang terkenal dengan nama IEEE Recommended Practice for Software Requirements Specifications [IEEE-830].
- Dokumen spesifikasi requirements bisa berisi functional requirements, performance requirements, external interface requirements, design constraints, maupun quality requirements.

Req. Validation & Verification

- Pasca spesifikasi requirement dibuat, perlu dilakukan dua usaha:
  - Validation (validasi), yaitu proses untuk memastikan bahwa requirement yang benar sudah ditulis.
  - Verification (verifikasi), yaitu proses untuk memastikan bahwa requirement sudah ditulis dengan benar.
- Proses validasi dan verifikasi ini melibatkan kustomer (user) sebagai pihak yang menilai dan memberi *feedback* berhubungan dengan requirements.

Requirement Engineering

- Requirement engineering biasa dilakukan dengan cara:
  - Survey Cek Fisik (lokasi, sistem eksisting, network dan infrastruktur + media komunikasi, database eksisting, user, dsb).
  - Wawancara + diskusi dengan user
  - Questionnaire
  - Dll.

Communication Principles

1. Listen
2. Prepare before you communicate
   - membawa kamera, alat perekam, kartu nama
   - lakukan pemisahan forum antar stakeholders yang berbeda, mis antara atasan/manajemen dan bawahan/operator
3. Someone should facilitate the activity
   - counterpart : seseorang/CP yang ada dalam sistem, utk memastikan semua proses lancar, bebas birokrasi
4. Face-to-face communication is best
   - untuk bisa membaca ekspresi (marah, enjoy, segan, dll) , lebih dari sekedar mendengar suara
5. Take notes and document decisions
6. - catatan, rekaman, dll

Communication Principles

6. Strive for collaboration
    - utk mencairkan suasana
7. Stay focused; modularize your discussion
    - harus punya target, sehingga bisa menggiring diskusi
8. If something is unclear, draw a picture
9. Once you agree to something OR If you can't agree to something OR If a feature or function is unclear and cannot be clarified at the moment, move on.
10. Negotiation is not a contest or a game. It works best when both parties win.

Communication Principles

- Before customer requirements can be analyzed, modeled, or specified they must be gathered through the communication activity.
- A customer has a problem that may be amenable to a computer-based solution. You respond to the customer's request for help.
- Communication has begun. But the road from communication to understanding is often full of potholes.

About SRS

- A *software requirements specification (SRS) is* a document that is created when a detailed description of all aspects of the software to be built must be specified before the project is to commence.
- It is important to note that a formal SRS is not always written. In fact, there are many instances in which effort expended on an SRS **might be better spent in other software engineering activities.**
- When software is to be developed by a third party, when a lack of specification would create severe business issues, or when a system is extremely complex or business critical, an SRS may be justified.

About SRS

- It is better to use a standardized SRS, such as SRS based on IEEE template.
- But, **it is better to write the only requirement specification needed**. Don’t write the useless things.

IEEE requirements standard

- Defines a generic structure for a requirements document that must be instantiated for each specific system.
  - Introduction.
  - General description.
  - Specific requirements.
  - Appendices.
  - Index.

Discussion