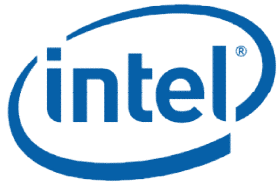


Specifying Effective Non-Functional Requirements

John Terzakis

Intel Corporation

June 24, 2012
ICCGI Conference
Venice, Italy

Version 1.0

Legal Disclaimers

Intel Trademark Notice:

Intel and the Intel Logo are trademarks of Intel Corporation in the U.S. and other countries.

Non-Intel Trademark Notice:

*Other names and brands may be claimed as the property of others

Agenda

- Requirements Overview
- Natural Language & Its Issues
- Natural Language Non-Functional Requirements
- Planguage: A Technique for Writing Effective Non-Functional Requirements
- Essential Planguage Keywords for Non-Functional Requirements
- Using Planguage to Rewrite the NFR Examples to be Verifiable
- Wrap up

Requirements Overview

What is a Requirement?

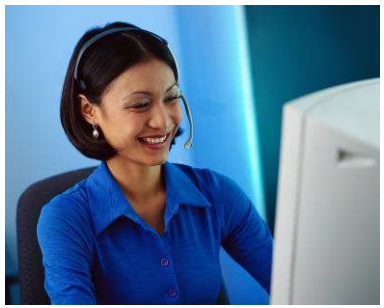
A **requirement** is a statement of one of the following:

1. What a system must do
2. A known limitation or constraint on resources or design
3. How well the system must do what it does

The first definition is for **Functional Requirements**

The second and third definitions are for **Non-Functional Requirements (NFRs)**

Examples of Functional and Non-Functional Requirements



Video over IP
Conference Calling

Functional Requirements

- Add Participant
- Count Participants
- Drop Participant
- Lock Call to New Participants
- Summon Operator
- Mute microphone

Non-Functional Requirements

- Voice and Video Quality
- Reliability
- Availability
- Ease of Use
- Cost
- Localization

Functional Requirements

A **Functional Requirement:**

- is a statement of what a system must do (#1)
- is measured in “yes” or “no” terms
- usually employs the word “shall”

Examples:

Add Participant

“The software shall display an option to add a participant”

Summon Operator

“The software shall summon the operator if the participant clicks the Operator Help icon.”

Non-Functional Requirements (1 of 2)

A **Non-Functional Requirement**:

- is a known limitation or constraint on resources or design (#2)
- usually measured in yes/no terms
- can include documentation, marketing collateral, product localization, legal compliance restrictions
- typically employs the word "must"

Examples:

Cost

"The retail cost of the software must be between $175 and $199."

Localization

"The help file must be released in English, French and Spanish."

Non-Functional Requirements (2 of 2)

A **Non-Functional Requirement**:

- is a measure of how well the system must do what it does (#3)
- Is measured over an interval or range
- usually employs the word “must”
- includes the “ilities” (e.g., quality, reliability, scalability, availability)

This type of requirement is problematic within most Requirements Engineering practices, and will be the focus of this tutorial. We’ll look at good examples later.

Natural Language & Its Issues

What is Natural Language?

Natural language is unconstrained, informal language as it is used in every day speech and writing (e.g., email)

Natural language is the most common medium for expressing requirements in most industries; It is flexible, easy to use and requires no additional training.

Exercise: Write Natural Language NFRs

The instructor will divide the class into small groups

1. Write 3-4 natural language non-functional requirements for the purchase of a new car.

   Example: The car must be reliable.

2. Discuss whether these non-functional requirements are verifiable or not

Issues with Natural Language NFRs

While useful in everyday interactions, natural language is fertile ground for a number of issues relating to requirements (functional as well as non-functional) including:

- Weak words
- Unbounded lists
- Implicit Collections
- Ambiguity
- Issues around verb choice, semantics, and grammar

Natural language tends to produce NFRs that are not verifiable

Weak Words

Weak words are subjective or lack a common or precise definition. Examples include:

- Quick, Quickly
- Easy, Easily
- Timely
- Fast
- Frequently
- Intuitive
- Feel, Feeling
- Normal
- Reliable
- State-of-the-art
- Effortless
- Friendly, User-friendly
- Secure
- Immediate

This is just a partial list. Don't use weak words – define what you mean using precise, measurable terms

Unbounded Lists

An unbounded list is one that lacks a starting point, an end point, or both

Examples include:

- At least
- Including, but not limited to
- Or later
- Such as

Unbounded lists are impossible to design for or to test against

For example, how would you design and test a system that "must maintain a list of at least 250 users"?

Or, how would you test software that "must install on Windows® Vista or later in under 5 seconds"?

Implicit Collections

Often, collections of objects within requirements are not explicitly defined anywhere

Without a definition, readers may assume an incorrect meaning

Example:

"The software must support 802.11 and other network protocols supported by competing applications under Linux."

- What is counted as a "competing application"?
- What belongs to the collection of "other network protocols"?
- What specific protocols of 802.11 are included?
- "Linux" is also a collection of OS vendors, versions, and revision levels

Ambiguity

Ambiguity occurs when a word or statement has multiple meanings or there is doubt about the meaning.

These problems (and others) create ambiguity:

- Vagueness
- Subjectivity
- Optionality
- Under-specification
- Under-reference

Ambiguity leads to differences in interpretation amongst the various stakeholders for a requirement.

Ambiguity Examples

Vagueness:

"The system must pass between 96-100% of the test cases using current standards for video encoding before launch."

Subjectivity:

"The debug code must easily and seamlessly integrate with the validation test automation software.

Optionality:

"The software should be tested under as many OSes as possible."

Under-specification:

"The software must support 802.11n and other network protocols"

Under-reference:

Users must be able to complete all previously-defined operations in under 5 minutes 80% of the time."

Issues With Verb Choice, Semantics, and Grammar

Be careful with verb choice

- What is difference between the words “enable”, “allow”, “assist”, “permit”, “authorize”, and “provide the capability to”?

Be careful with each, every, and only

- “Each” refers to one; “every” refers to all; both are universal qualifiers
- The placement of “only” can totally change the intent of the requirement

Avoid grammatical issues

- Use of a slash (e.g., “object1/object2”) creates confusion
  - Is it both terms (object1 and object 2) or just one (object 1 or object2)?
- Use of “and” with a preceding qualifier creates two options
  - Does the qualifier before the “and” apply to just the term before the “and” or both terms?

Verb Choice, Semantics, and Grammar Examples

Verb Choice:

- The SW must quickly provide the capability to users to access their invoices
- The SW must quickly authorize users to access their invoices

Each and every:

- Unless each user is authenticated, the SW must securely protect the data
- Unless every user is authenticated, the SW must securely protect the data

Placement of "only":

- Only authorized users can access medical information
- Authorized users can only access medical information

Grammatical Issues:

- The SW must email/log improper access attempts after 3 failures
- The SW must rapidly disable the accounts of unregistered users and guests

Exercise: Identify the Issues

The usability objective of the AlphaBeta Plus client is to be usable by the intended customer at a 5' distance. The client should be an integrated system that is both reliable and responsive. Reliability and responsiveness are more critical for this device than for PC desktop systems. Reliability should be as good as that of consumer home entertainment devices (e.g., TV or VCR) and response to user interaction should be immediate.

The applications should provide an easy-to-learn, easy-to-use, and friendly user interface, even more so than PC desktop applications. Users should be able to start using the application immediately after installation. Users should be able to satisfactorily use the device with little instruction.

Friendly means being engaging, encouraging, and supportive in use. Users must feel comfortable with the client and must not be given reason to worry about accidentally initiating a destructive event, getting locked into some procedure, or making an error. Feedback for interactions should be immediate, obvious, and appropriate.

Natural Language Non-Functional Requirements

Examples of Natural Language NFRs

Order processing must be fast

The software must support at least 25 users

Make the web site software reliable

The configuration software should be intuitive to use

The audio software must reproduce music nearly perfectly

Do you see any issues with these requirements?

Issues Identified

1. Order processing must be fast
   - How long is “fast”? Seconds, minutes or hours? Can we test “fast”?
2. The software must support at least 25 users
   - What is the meaning of “support”? Are these concurrent users or not?
   - How many is “at least” 25 users? 26 users? 200,000 users?
3. Make the web site software reliable
   - What is “reliable”? Can we test for it?
4. The configuration software should be intuitive to use
   - “should” implies optionality
   - What does “intuitive” mean? It is subjective (reader dependent)
5. The audio software must reproduce music nearly perfectly
   - What does “nearly perfectly” mean? An audiophile will have a different opinion than a causal listener.

Effective NFRs Must Be Verifiable

For a NFR to be effective, it must be verifiable.

A requirement is verifiable if it can be proved that the requirement was correctly implemented (i.e., we can test for correct implementation)

Requirements are often unverifiable because they contain weak words, utilize unbounded lists, include implicit collections, are ambiguous or have grammatical issues

Eliminating these issues is the first step towards writing effective NFRs

Planguage: A Technique for Writing Effective Non—Functional Requirements

What is Planguage?

Planguage is an informal, but structured, keyword-driven planning language

- Developed by Tom Gilb in 1988 and explained in detail in his book *Competitive Engineering* *
- Can be used to create all types of requirements
- Is a combination of the words *Plan*ning and Lan*guage*
- Is an example of a Constrained Natural Language

Planguage aids communication about complex ideas

* *Competitive Engineering,* Butterworth-Heinemann, 2005

Planguage

Planguage provides a rich specification of requirements that results in:

- Fewer omissions in requirements
- Reduced ambiguity and increased readability
- Early evidence of feasibility and testability
- Increased requirements reuse
- Effective priority management
- Better, easier decision making

Basic Planguage Keywords & Definitions

Tag: A unique, persistent identifier

Gist: A brief summary of the requirement or area addressed

Requirement: The text that details the requirement itself

Rationale: The reasoning that justifies the requirement

Priority: A statement of priority and claim on resources

Stakeholders: Parties materially affected by the requirement

Status: The status of the requirement (draft, reviewed, committed, etc.)

Owner: The person responsible for implementing the requirement

Author: The person that wrote the requirement

Continued…

Basic Planguage Keywords & Definitions

Revision: A version number for the statement

Date: The date of the most recent revision

Assumptions: All assumptions or assertions that could cause problems if untrue now or later

Risks: Anything that could cause malfunction, delay, or other negative impacts on expected results

Defined: The definition of a term (better to use a glossary)

Fuzzy concepts requiring more details: **<***fuzzy concept***>**

A collection of objects: **{***item1, item2, …***}**

The source for any statement: ←

Basic Planguage Keywords are useful for any requirement, and are sufficient for requirements measured as "present" or "absent"

A Simple Planguage Functional Requirement

Tag: Invoice ← {C. Smith, 07/06/05}

Requirement: When an Order is shipped and Order Terms are not "Prepaid", the system shall create an Invoice.

Rationale: Task automation decreases error rate, reduces effort per order. Meets corporate business principle for accounts receivable.

Priority: High. If not implemented, it will cause business process reengineering and reduce program ROI by $400K per year.

Stakeholders: Shipping, finance

Author, Revision, Date: Julie English, rev 1.0, 5 Oct 05

Choosing Planguage Keywords

Recall that requirements generally fall into two categories based on the nature of how they are measured

Functional Requirements are measured in Boolean (simple yes/no) terms as either present or absent in the completed system

- This category includes *system functions* and *constraints*

Non-Functional Requirements are typically measured on some scale or interval, not simply "present" or "absent"

- This category includes *system qualities* and *performance levels*

Because of the way they are measured, Non-Functional Requirements use some additional Planguage keywords

Additional Planguage Keywords for Non-Functional Requirements

Ambition	A description of the goal of the requirement (this replaces the Requirement keyword used in functional requirements)
Scale	The scale of measure used to quantify the requirement (e.g., time, temperature, speed)
Meter	The process or device used to establish location on a Scale (e.g., watch, thermometer, speedometer)
Minimum (Must)	The minimum level required to avoid political, financial, or other type of failure
Target (Goal)	The level at which good success can be claimed
Outstanding (Stretch)	A feasible stretch goal if everything goes perfectly

A Simple Planguage NFR

Tag: Menu Complexity

Ambition: Make Accessing Patient Dental History Menus easier

Scale: Number of menus

Meter: Measured from the login menu to display of the most recent patient dental visit

Minimum: 4

Target: 3

Outstanding: 2

Note: the term “easier” in the Ambition is OK since it is qualified by the keywords that follow

Notes on Planguage Keywords

- Use the keywords that add value to your statement - no more, no less
- There are many more keywords to Planguage than presented here – See *Competitive Engineering* for more examples
- Extend Planguage as needed with new keywords - but it's good to check to see whether there is already a keyword that will work

Exercise: Using Planguage for NFRs

The instructor will divide the class into the same groups as the previous car purchase exercise

Use the following template to write NFRs for the top speed and fuel economy for a new car purchase:

Ambition:
Scale:
Meter:
Minimum:
Target:
Outstanding:

Essential Planguage Keywords for Non-Functional Requirements

Focus on Essential NFR Planguage Keywords

The following Planguage keywords are important for specifying effective Non-Functional Requirements:

- Scale
- Meter
- Minimum
- Target
- Outstanding

Let's look at all five in detail

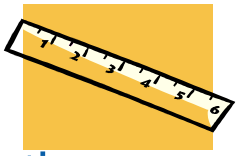
Scales

Scale: The scale of measure used to quantify the statement

There are three types of scales:

- Natural: Scales with obvious association to the measured quality
- Constructed: A scale built to directly measure a quality
- Proxy: An indirect measure of a quality

Examples of Scales

Natural	Time measured in seconds Number of users 
Constructed	A 5-point scale created to measure perceived sound quality  A 10-point scale created to register user satisfaction
Proxy	An in-field MTTF goal predicted using pre-release reliability test results “Critical” defect prediction for first year of released software based on defect trending during Beta testing 

Finding Scales

Start by looking for a natural scale. If none comes to mind:

- Create a constructed scale
- Look for a proxy scale
- Decompose the concept being measured into its parts and try again

Other hints:

- Use known, accepted scales of measure when possible
- Derive new scales from known scales by substituting terms
- Incorporate qualifiers in the scales to increase specificity
- Don’t confuse scale with meter
- Share effective scales with others

Meters

Meter: The process or device used to establish location on a Scale

Most meters have an obvious association with the scale they are measuring (e.g., time with a stop watch)

Some meters may require a process or test procedure to be utilized or created

Examples of Meters

Natural	A stopwatch  Log of users authenticated 
Constructed	"Double blind" tests  User satisfaction survey  
Proxy	Stress testing of pre-production software, analyzing results and predicting the Mean Time to Failure (MTTF) Validation testing of Beta software, analyzing results and predicting the number of critical defects in the first year of customer release 

Finding Meters

First, study the scale carefully. If no meter comes to mind:

- Look at references and handbooks for examples for ideas
- Ask others for their experience with similar methods
- Look for examples within test procedures

Once you have a candidate, check to see that:

- The meter is adequate in the eyes of all stakeholders
- There is no less-costly meter available that can do the same job (or better)
- The meter can be employed before product release or completion of the deliverable

Examples of Paired Scales and Meters

Tag: Response Time
Scale: Time in seconds
Meter: Measured from mouse click to display of next menu

Tag: Software Maintainability
Scale: Average engineering time from report to closure of defects
Meter: Analysis of 30 consecutive defects reported and corrected during product development

Tag: Market Share
Scale: % of Total Available Market
Meter: Quarterly market survey

Remember: Scale = units of measure,
Meter = Device or process to measure position on the Scale

Minimum, Target & Outstanding Keywords

Minimum: The minimum level required to avoid political, financial, or other type of failure

Target: The level at which good success can be claimed

Outstanding: A stretch goal if everything goes perfectly

Notes:

- Development and testing is typically focused on achieving the Target level
- Values not meeting at least the Minimum level mean the NFR has not been correctly implemented (verification has failed)
- At least one of these keywords should be specified for a NFR
- Collectively, these keywords can be referred to as a **Landing Zone**.

Landing Zones

A Landing Zone defines a “region of success” for a Non-Functional requirement.

Any time between 7seconds and 10 seconds **meets** the requirement.
Any time greater than 10 seconds means the requirement **has not been met.**

Landing Zones focus attention on what will create success

Example Landing Zones

Requirement	Minimum	Target	Outstanding
Release Date	1 Sep 11	15 Aug 11	13 Jul 11
Install time	5 seconds	4 seconds	3 seconds
Peak Project Headcount	40 SW developers	35 SW developers	25 SW developers
# of transactions per minute	375	450	500
Design Wins	20+	30+	40+
Total First Year Volume	95K	110K	125K

Using Planguage to Rewrite the NFR Examples to be Verifiable

Example 1

Order processing must be fast

Tag: Order Processing Time

Ambition: Don't make the users wait too long for order processing

Scale: Time

Meter: Measured from the user clicking on the "Submit Order" icon to the display of the "Order Complete" message on the order entry menu.

Minimum: 5 seconds

Target: 4 seconds

Outstanding: 3 seconds

Exercise: Rewrite Example 2

The software must support at least 25 users

Tag:

Ambition:

Scale:

Meter:

Minimum:

Target:

Outstanding:

Hint: 25 users at a time or one at a time?

Exercise: Rewrite Example 3

Make the web site software reliable

Tag:

Ambition:

Scale:

Meter:

Minimum:

Target:

Outstanding:

Hint: How will we measure this reliability? What is our scale?

Exercise: Rewrite Example 4

The configuration software should be intuitive to use

Tag:

Ambition:

Scale:

Meter:

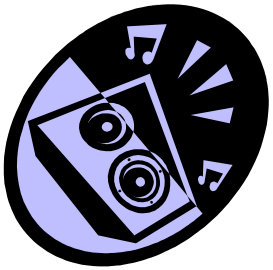
Minimum:

Target:

Outstanding:

Hint: Think of an example of configuration SW

Exercise: Rewrite Example 5



The audio software must reproduce music nearly perfectly

Tag:

Ambition:

Scale:

Meter:

Minimum:

Target:

Outstanding:

Hint: What type of Scale? Natural, Constructed or Proxy?

Wrap up

Session Summary

In this session we have:

- Provided an overview of functional and non-functional requirements
- Defined natural language and identified its issues (weak words, unbounded lists and ambiguity)
- Introduced Planguage, a technique for writing effective non-functional Requirements
- Examined critical Planguage keywords in detail
- Rewritten natural language non-functional requirements so that they are verifiable

Final Thoughts

- Effective NFRs are verifiable

  You must be able to verify a NFR to know it's been implemented correctly

- Removing weak words, unbounded lists and ambiguity is key to making NFRs verifiable

  Specify NFRs using objective, bounded terms

- Planguage provides the framework to make NFRS verifiable

  Use the critical Planguage keywords to assist in developing the proper test for a NFR

Writing effective NFRs is crucial for determining whether product performance and quality goals have been met

Contact Information

Thank You!

For more information, please contact:

John Terzakis

john.terzakis@intel.com

Back up

Possible Solution: Example 2

The software must support at least 25 users

Tag: Number of Concurrent Users

Ambition: Handle as many concurrent users as possible

Scale: Number of concurrent users

Meter: Concurrent users logged in, authenticated and registering for the same conference using the Beta software while maintaining a response time of 1 sec or less for any single user

Minimum: 25

Target: 50

Outstanding: 70

Possible Solution: Example 3

Make the web site software reliable

Tag: Web Site Software Reliability

Ambition: Make the web site software as reliable as possible

Scale: Number of “show stopper” defects

Meter: Measurement of all classes of defects reported by customers during Alpha testing

Minimum: 5

Target: 3

Outstanding: 0

Possible Solution: Example 4

The configuration software should be intuitive to use

Tag: Configuration SW Usability

Ambition: Make the configuration software easy to use

Scale: Average time required for a Novice to configure the wireless router for WPA using only the online help system for assistance

Meter: Measurements obtained on 100 Novices during user interface testing.

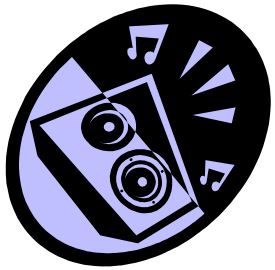
Minimum: Less than 30 seconds

Target: Less than 25 seconds

Outstanding: Less than 20 seconds

Defined: Novice: A user with no prior experience with the software

Possible Solution: Example 5



The audio software must reproduce music nearly perfectly

Tag: Perceived Audio Quality

Ambition: Produce nearly perfect music reproduction

Scale: Score on a five-point scale: 5=imperceptible; 4=perceptible, but not annoying; 3=slightly annoying; 2=annoying; 1=very annoying

Meter: The “double-blind triple-stimulus with hidden reference” method as found in Recommendation ITU-R BS.1116-1, “Methods For The Subjective Assessment Of Small Impairments In Audio Systems Including Multi-Channel Sound Systems”.

Minimum: 4.0

Target: 4.5

Outstanding: 4.8


intel®
Leap ahead™