Automated requirements

Tom Rochette <tom.rochette@coreteks.org>

December 21, 2025 — 77e1b28a

0.1 Context

An important aspect of the job of a programmer is to convert clients requirements into computer procedures. In order to do so, the programmer must be aware of the existing functionalities of the system and understand how the clients requests will integrate within the existing system. The goal of this study is to discover the various tasks related to requirements elicitation that must be done in order to move to the step of programming said requirements.

0.2 Learned in this study

0.3 Things to explore

- Is there a way to write requirements such that it is some kind of formal language?
  - There's already the "MUST/SHOULD/COULD/WON'T" (MoSCoW method) which indicates requirements priority

1 Overview

- Verify if this is similar to the ability of generating test cases for functions (automated testing)

2 Steps

- The client determines he has new needs that the software should be fulfilling
- The client and the software development representatives discuss about the requested features
- New features are evaluated against existing requirements in order to determine if existing functionality will be reused, require to be modified, or written from scratch
- The time to develop a feature is roughly estimated in order to assess the price tag of each feature (X hours = Y $, generally evaluated at Z $/hour)
- The features are ordered by priority and/or ROI
- Dependencies between features are established
- A development plan is prepared

3 Communication and requirements gathering

We cannot free ourselves from (programming) languages if our goal is to communicate needs between individuals. Even if we make an AGI that understands natural language, we still will have to communicate with it, and this communication will be imperfect and require filling in the blanks and reducing the ambiguity of the requirements. Communicating requirements and implementing them is thus a trade-off between the two. It requires the implementer to have prior domain experience/knowledge of what has been built so that new information conveyed by the user can relate to this prior work. This implies that the implementer needs

to have the ability to form a model of what it has built so it can reuse that information when gathering new requirements.

We can see the definition of a requirement as starting with a tensor of infinite dimension, and as more and more of the behavior of this requirement is defined, the dimension of the tensor is lowered until it becomes a concrete tensor.

This also implies that there are two types of activities: modifying existing functionalities to add to or modify their behavior and adding new functionality from scratch. In the latter case, this means that the implementer will have to be able to infer data structures given a problem specification.

2 types of flow:

- Requesting a new program
- Modifying an existing program

3.0.1 Requesting a new program

User: I want a program that list all my tasks. -> (program request)
Implementor: What does such a program do?
User: It lists my tasks. -> (program description + noun extraction)
Implementor: What are tasks?
User: They have a title, a description, a start and end date. -> (data structure definition)
Implementor: Does this structure seem appropriate to you?
Title: string
Description: string
Start Date: Date
End Date: Date
User: Start Date and End Date should be timestamps.
Implementor: Does this structure seem appropriate to you?
Title: string
Description: string
Start Date: Timestamp
End Date: Timestamp
User: Yes.

3.0.2 Modifying an existing program

User: I'd like to make a change to my program.
Implementor: Which one?
User: Program X. I want to make it more intelligent.
Implementor: How do you want to do that?
User: I don't know. You figure that out.
Implementor: Since you did not provide a definition for "more intelligent", I will assume that the program is now "more intelligent". Task completed.
User: That is not what I asked!

4 Questions

- How many users will use the system?
- When will users use the system?
- What languages should it support?
- Does the system need to work without Internet access?
- How often do the users use the system on a daily basis?
- How long do they use the system per day?
- What is the most frequent task?

- What is the easiest/most difficult task to execute?
- How much data will the system manage?

5 Prototype language

- MUST/SHOULD/COULD/WON'T
- Specify the affected actor

5.1 Examples

The system <PRIORITY> <ACTION> <ACTOR> <TARGET> <CONDITION> <WHEN>

PRIORITY := MUST|SHOULD|COULD|WON'T
ACTION := text (identifies the procedure that should be executed)
ACTOR := text (uniquely identify the actor it applies to)
CONDITION := IF <CONDITION_DESCRIPTION>
CONDITION_DESCRIPTION := text (specify the conditions that must be respected)
WHEN := text (specify when it occurs, either periodically, or due to some condition)

The system <MUST> <export>

5.2 Alternative

As <who> <when> <where>, I <what> because <why>

5.3 Gherkin

Desired Future state

What business requirements will this system address?
What information do you need from this system that you don't have now?
Where is any of this data currently captured in another corporate system?
How would you like to see this information?
What functionality do you need from the system?
What data and/or functionality is shared by other (many) business areas?
If the reports were dynamic, what would they do differently?
How much historical information is required?

Business Objectives

What are your goals in developing this system?
Who are the key stakeholders and users? Do their goals differ? If so, how?
How do the system goals map to business goals?
What is the most important business goal of the system?
How will the system change the way you are doing things now?
How will the system help you be more efficient?
What are the system deliverables?
What will the new system accomplish that is not currently accomplished manually or with other systems?
What will the new system do?

Current Problems

What are the current problems you are facing today without the system?
What problems should this system solve?
What do you have to do manually that you would like to automate?
What performance problems need to change?

What functional limitations would you like to change?
What packages are you using that force you to constrain your business functionality to the boundaries of the package?
Which reports do you currently use? What data on the report is important? How do you use the information?
Where are there specific bottlenecks to getting at information?
How do you analyze the information you currently receive? What type of data is used? How do you currently get the data? How often do you get new data?
What type of ad hoc analysis do you typically perform? Who requests ad hoc information? What do you do with the information?

Success Criteria (What does "winning" look like?)

What is most important for success of the application?
What do we need to accomplish to make this project successful?
What do we need to change to make this project successful?
What buy-in do we need?
What critical elements such as budget, resource allocation, or support are we lacking?
What are training considerations for developers and users?

Identify Users

Who will be using the system?
What are the titles and roles of the people who will use the system?
What are their levels of expertise?

Source: http://blog.learningtree.com/the-right-questions-to-ask-in-requirements-elicitation/

6 See also

- Automated programming
- A procedure for writing programs

7 References

- https://openai.com/requests-for-research/#description2code
- https://github.com/Avmb/code-docstring-corpus
- https://en.wikipedia.org/wiki/User_story
- https://dev.to/tra/will-programming-be-automated-a-slack-chat