GAME OF ROLES

Software Engineering, Course 2022-2023

Following with the previous business case of Spotify:

Owner: Spotify

Extending the touchpoints and avenues by which users can stream music

Key facts

- In the entertainment space, Spotify has become a pioneering example of an atomized service, achieving ubiquity by enabling access through multiple third-party touch-points (e.g. Sonos, Ford, iOS, Android and Samsung Smart TVs).
- The company has released Software Development Kits (SDKs) for iOS and Android developers and more recently launched the Spotify + Uber integration, allowing users to remotely control music in enabled Uber rides.

Uniqueness

- Spotify is extending the touchpoints and avenues by which users can stream music through their service by collaborating across sound system, home and auto entertainment providers.
- Spotify Connect sets up a connection between Hi-Fi and Wi-Fi – allowing streaming directly via Spotify and not via a user's phone (which instead can serve as a remote and remain free to use for other activities).

Value

- Spotify is valued at over $10 billion.
- Services 50 million users, with over 12.5 million paying subscribers.
- Approximately 30% of revenue is retained by the company, while approximately 70% is split among rights holders.

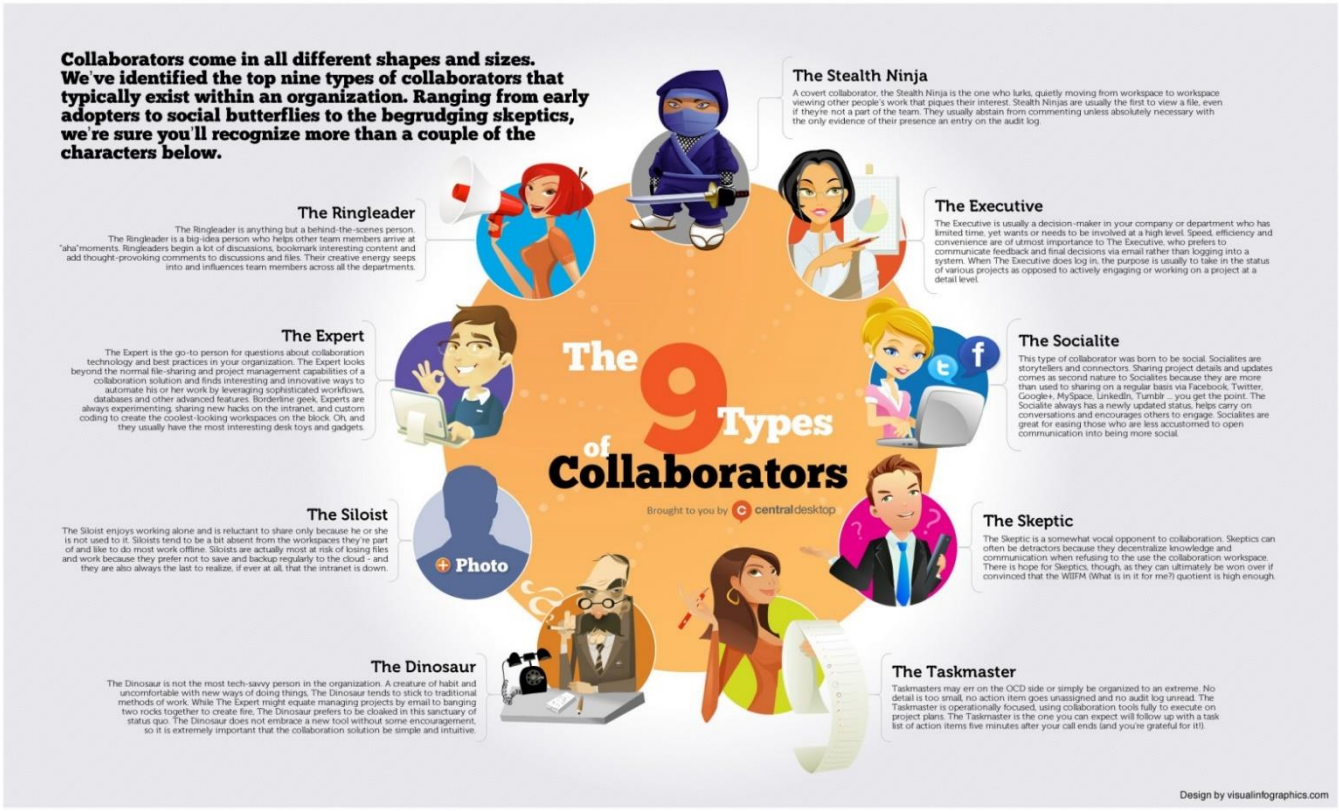
Approach

- Spotify has created connections with 30+ partners.
- Spotify utilizes a Freemium business model, with potential to drive conversions from new partnerships and platforms.
- In order to benefit from the interoperability that Spotify Connect offers, users must subscribe to Spotify Premium ($9.99/mo)

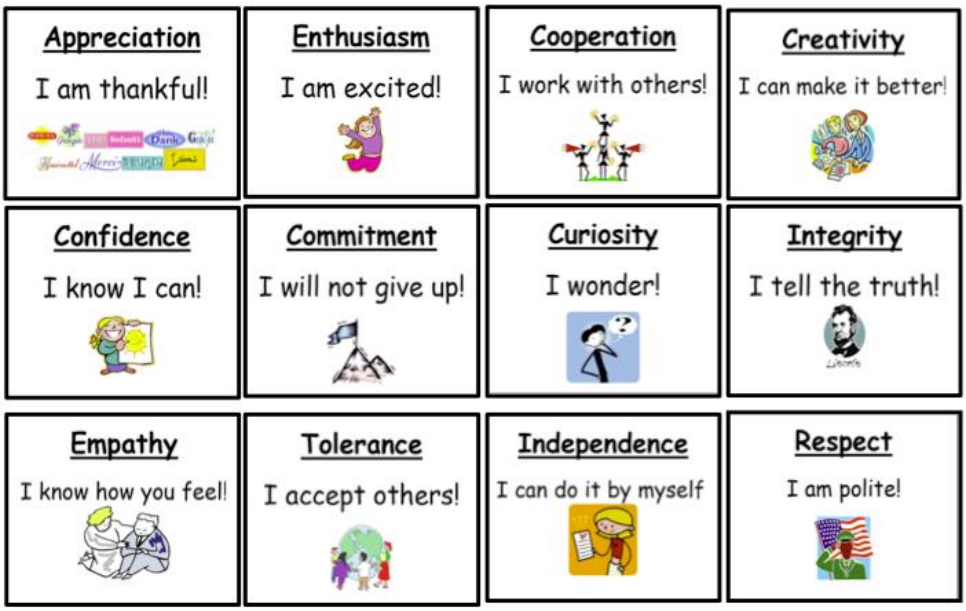
1. **You must play the different roles we can find within a software engineer project.**
    a. **Client**

b. **End-user**
c. **Business analyst/Requirements Engineer**

To make the game more realistic, consider adopting some of these roles and attitudes:



Source: http://infographics-images.idlelist.com/wp-content/uploads/2013/03/the-9-types-of-collaborators-1024x624.jpg



Source: http://www.sps186.org/images/basic/269894_1345927987.png

And also consider to include some of the "sins of the specifier":

Table 1.
The seven sins of the specifier.

Noise:	The presence in the text of an element that does not carry information relevant to any feature of the problem. Variants: *redundancy; remorse.*
Silence:	The existence of a feature of the problem that is not covered by any element of the text.
Overspecification:	The presence in the text of an element that corresponds not to a feature of the problem but to features of a possible solution.
Contradiction:	The presence in the text of two or more elements that define a feature of the system in an incompatible way.
Ambiguity:	The presence in the text of an element that makes it possible to interpret a feature of the problem in at least two different ways.
Forward reference:	The presence in the text of an element that uses features of the problem not defined until later in the text.
Wishful thinking:	The presence in the text of an element that defines a feature of the problem in such a way that a candidate solution cannot realistically be validated with respect to this feature.

Source: http://se.ethz.ch/~meyer/publications/ieee/formalism.pdf

For instance, if I play the role of a client (who is going to pay):

As a client:
- **"I only have XXXX €"**
- **"I need the system in X months"**
- **"What is the improvement?"**
- **"Which are the critical success factors?"**
- **"This product is very similar to the one we already have"**
- **…**

As an end-user:
- **"I want to do this, this and this…"**
- **"The system must be easy to use."**
- **"I have this problem…"**
- **"I want something like XXX"**
- **…**

As a requirements engineering:
- **"This is not possible.**
- **"This is not realistic."**

- **"If possible it will take a lot of time."**
- **…**

2. **Create a set of functional (10) and non-functional requirements (5).**

ID	Title	Type	Source	Derived from
1	The <ENTITY> shall <action>…			
2				

3. **Check if your requirements are "SMART". Show an evidence.**

	Concept	Description
S	**S**pecific	"A good requirement is specific and not generic. It should not be open to mis-interpretation when read by others. This is the most important attribute to get correct."
M	**M**easurable	"This answers whether you will be able to verify the completion of the project. You should avoid signing up for any requirement that cannot be verified as complete. These are especially risky when you use non-quantitative terms (best, optimal, fastest) for acceptance criteria.."
A	**A**chievable	"The requirement is physically able to be achieved given existing circumstances"
R	**R**ealistic	"Whether the requirement is realistic to deliver when considering other constraints of the project and requirements."
T	**T**ime-bound	"Each requirement should be time-bound or specify by *when* or *how fast* a requirement needs to be completed or executed. "

4. **Rewrite your requirements according to the following rules.**

Precision

R1 – Use definite article "the" rather than the indefinite article "a."

R2 – Use the active voice with the actor clearly identified.

R3 – Make the subject of the requirement appropriate to the layer in which the requirement exists.

R4 – Only use terms defined in the glossary.

R5 – Quantify requirements precisely – Avoid imprecise quantifiers that provide vague quantification, such as "some," "any," "several," "many," "a lot of," "a few," "approximately," "almost always," "nearly," "about," "close to," "almost," "approximate," "significant," "flexible," "expandable," "typical," "sufficient," "adequate," "appropriate," "efficient," "effective," "proficient," "reasonable."

R6 – Use appropriate units, with tolerances or limits, when stating quantities – Explicitly state units for all numbers.

R7 – Avoid the use of adverbs – words that end in -ly" Such as "usually," "approximately," "sufficiently," "typically"

R8 – Avoid the use of vague adjectives such as "ancillary," "relevant," "routine," "common," "generic" and "customary."

R9 – Avoid escape clauses such as "so far as is possible," "as little as possible," "as much as possible," "if it should prove necessary," "where possible" and "if practicable."

R10 – Avoid open-ended clauses such as "including but not limited to," "etc." and "and so on."

Concision

R11 – Avoid superfluous infinitives such as ".. be designed to...," "...be able to....," "...be capable of...."

R12 – Use a separate clause for each condition or qualification.

Non-ambiguity

R13 – Use correct grammar.

R14 – Use correct spelling.

R15 – Use correct punctuation.

R16 – Use the logical construct "X AND Y" instead of "both X AND Y."

R17 – Avoid the use of "X and/or Y."

R18 – Avoid the use of the oblique ("/") symbol except in units, i.e., Km/hr

Singularity

R19 – Write a single, simple, single-thought, affirmative, declarative sentence, conditioned and qualified by relevant sub-clauses.

R20 – Avoid combinators such as "and," "or," "then," "unless," "but," "/," "as well as," "but also," "however," "whether," "meanwhile," "whereas," "on the other hand" and "otherwise."

R21 – Use an agreed typographical device to indicate the use of propositional combinators for expressing a logical condition within a requirement.

R22 – Avoid phrases that indicate the purpose of the requirement.

R23 – Avoid parentheses and brackets containing subordinate text.
R24 – Enumerate sets of entities as explicit requirements instead of using generalizations.
R25 – When a requirement is related to complex behavior, refer to the supporting diagram or model rather than a complex textual description

Completeness

R26 – Avoid the use of pronouns.
R27 – Avoid using headings to support explanation of subordinate requirements.

Realism

R28 – Avoid using unachievable absolutes such as 100% reliability or 100% availability.

Conditions

R29 – State applicability conditions explicitly instead of leaving applicability to be inferred from the context.
R30 – Express the propositional nature of a condition explicitly instead of giving lists of conditions.

Uniqueness

R31 – Classify the requirement according to the aspects of the problem or system it addresses.

R32 – Express each requirement once and only once.

Abstraction

R33 – Avoid stating a solution – focus on the problem "what" rather than the solution "how."

Quantifiers

R34 – Use "each" instead of "all," "any" or "both" when universal quantification is intended

R35 – Define the range of acceptable values associated with quantities.

R36 – Provide specific measurable performance targets.

R37 – Define temporal dependencies explicitly instead of using indefinite temporal keywords.

Uniformity of Language

R38 – Define a consistent set of terms to be used in requirement statements in a glossary – avoid the use of synonyms.

R39 – If acronyms are used in requirement statements, use a consistent set of acronyms.

R40 – Avoid the use of abbreviations in requirement statements.

R41 – Use a project-wide style guide.

Modularity

R42 – Group mutually dependent requirements together.

R43 – Group related requirements together.

R44 – Conform to a defined structure or template.