MoSCoW method

The **MoSCoW method** is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement; it is also known as *MoSCoW prioritization* or *MoSCoW analysis*.

The term *MoSCoW* itself is an acronym derived from the first letter of each of four prioritization categories (*Must have*, *Should have*, *Could have*, and *Won't have*), with the interstitial *O*s added to make the word pronounceable. While the *O*s are usually in lower-case to indicate that they do not stand for anything, the all-capitals *MOSCOW* is also used.

Contents

Background

This prioritization method was developed by Dai Clegg[1] and first used extensively with the agile project delivery framework Dynamic Systems Development Method (DSDM).[2]

MoSCoW is often used with timeboxing, where a deadline is fixed so that the focus must be on the most important requirements, and as such is a technique commonly used in agile software development approaches such as Scrum, rapid application development (RAD), and DSDM.

Prioritization of requirements

All requirements are important, but they are prioritized to deliver the greatest and most immediate business benefits early. Developers will initially try to deliver all the *Must have*, *Should have* and *Could have* requirements but the *Should* and *Could* requirements will be the first to be removed if the delivery timescale looks threatened.

The plain English meaning of the prioritization categories has value in getting customers to better understand the impact of setting a priority, compared to alternatives like *High*, *Medium* and *Low*.

The categories are typically understood as:[3]

Must have
: Requirements labeled as *Must have* are critical to the current delivery timebox in order for it to be a success. If even one *Must have* requirement is not included, the project delivery should be considered a failure (note: requirements can be downgraded from *Must have*, by agreement with all relevant stakeholders; for example, when new requirements are deemed more important). *MUST* can also be considered an acronym for the Minimum Usable SubseT.

Should have
: Requirements labeled as *Should have* are important but not necessary for delivery in the current delivery timebox. While *Should have* requirements can be as important as *Must have*, they are often not as time-critical or there may be another way to satisfy the requirement, so that it can be held back until a future delivery timebox.

Could have
Requirements labeled as *Could have* are desirable but not necessary, and could improve user experience or customer satisfaction for little development cost. These will typically be included if time and resources permit.

Won't have (this time)
Requirements labeled as *Won't have* have been agreed by stakeholders as the least-critical, lowest-payback items, or not appropriate at that time. As a result, *Won't have* requirements are not planned into the schedule for the next delivery timebox. *Won't have* requirements are either dropped or reconsidered for inclusion in a later timebox. (Note: occasionally the term *Would like to have* is used; however, that usage is incorrect, as this last priority is clearly stating something is outside the scope of delivery).

Variants

Sometimes W is used to mean Wish (or Would), i.e. still possible but unlikely to be included (and less likely than Could). This is then distinguished from X for Excluded for items which are explicitly not included.

Use in new product development

In new product development, particularly those following agile software development approaches, there is always more to do than there is time or funding to permit (hence the need for prioritization).

For example, should a team have too many potential epics (i.e., high-level stories) for the next release of their product, they could use the *MoSCoW method* to select which epics are *Must have*, which *Should have*, and so on; the minimum viable product (or MVP) would be all those epics marked as *Must have*.[4] Oftentimes, a team will find that, even after identifying their MVP, they have too much work for their expected capacity. In such cases, the team could then use the *MoSCoW method* to select which features (or stories, if that is the subset of epics in their organisation) are *Must have*, *Should have*, and so on; the minimum marketable features (or MMF) would be all those marked as *Must have*.[5] If there is sufficient capacity after selecting the MVP or MMF, the team could then plan to include *Should have* and even *Could have* items too.[6]

Criticism

Criticism of the MoSCoW method includes:

- Lack of rationale around how to rank competing requirements: why something is *must* rather than *should*.[7][8]
- Ambiguity over timing, especially on the *Won't have* category: whether it is not in this release or not ever.[7]
- Potential for political focus on building new features over technical improvements (such as refactoring).[8]

References

1. Clegg, Dai; Barker, Richard (1994). *Case Method Fast-Track: A RAD Approach*. Addison-Wesley. ISBN 978-0-201-62432-8.
2. Bittner, Kurt; Spence, Ian (2002-08-30). *Use Case Modeling*. Addison-Wesley Professional. ISBN 978-0-201-70913-1.
3. "MoSCoW Analysis (6.1.5.2)". *A Guide to the Business Analysis Body of Knowledge* (2 ed.). International Institute of Business Analysis. 2009. ISBN 978-0-9811292-1-1.
4. Wernham, Brian (2012). *Agile Project Management for Government*. Maitland and Strong. ISBN 0957223404.
5. Davis, Barbee (2012). *Agile Practices for Waterfall Projects: Shifting Processes for Competitive Advantage*. Project Management Professional Series. J. Ross Publishing. ISBN 1604270837.
6. Cline, Alan (2015). *Agile Development in the Real World*. Apress. ISBN 1484216792.
7. Wiegers, Karl; Beatty, Joy (2013). *Software Requirements*. Washington, USA: Microsoft Press. pp. 320–321. ISBN 978-0-7356-7966-5.
8. McIntyre, John (October 20, 2016). "Moscow or Kano - how do you prioritize?" (http://www.hotpmo.com/blog/moscow-kano-prioritize). *HotPMO!*. Retrieved October 23, 2016.

External links

- RFC 2119 (Requirement Levels) (http://www.ietf.org/rfc/rfc2119.txt) This RFC defines requirement levels to be used in formal documentation. It is commonly used in contracts and other legal documentation. Noted here as the wording is similar but not necessarily the meaning.
- Buffered Moscow Rules (http://manage.techwell.com/articles/membersub/time-boxing-planning-buffered-moscow-rules) This essay proposes the use of a modified set of Moscow rules that accomplish the objectives of prioritizing deliverables and providing a degree of assurance as a function of the uncertainty of the underlying estimates.
- MoSCoW Prioritisation (https://www.dsdm.org/content/moscow-prioritisation) Steps and tips for prioritisation following the DSDM MoSCoW rules.
- The ToToTo Method (http://creativehero.es/tototo-method/) A method inspired by the MoSCoW Method of prioritization.
