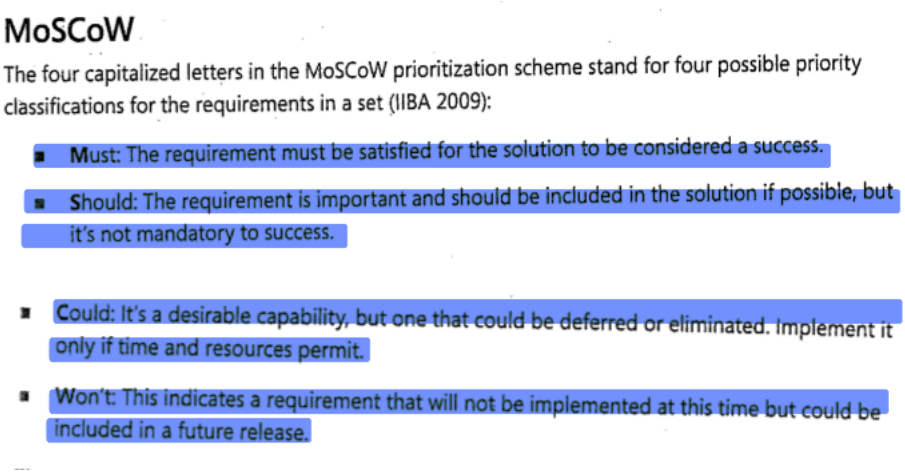
Faculdade UnB Gama - FGA
Professor: André Barros de Sales Disciplina: Requisitos de Software
Matrícula: 231027186 Nome: Samuel Nogueira Caetano
Tópico: Engenharia de Requisitos – Lista de verificação – Técnica de Priorização – MoSCoW

A. Definições Claras e Acordadas

1. As definições para cada uma das quatro classificações de prioridade (Must, Should, Could, Won't) foram claramente estabelecidas e compartilhadas com todos os stakeholders?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)



MoSCoW

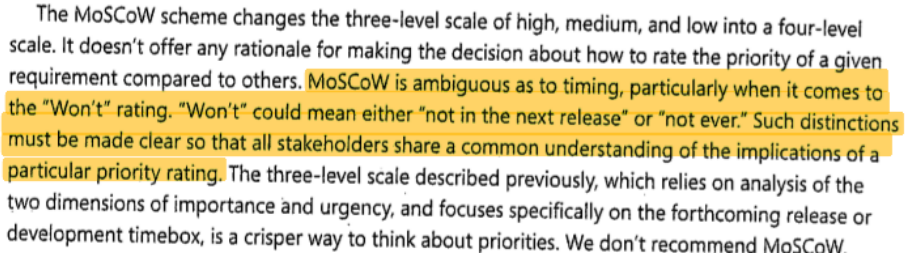
The four capitalized letters in the MoSCoW prioritization scheme stand for four possible priority classifications for the requirements in a set (IIBA 2009):

- Must: The requirement must be satisfied for the solution to be considered a success.
- Should: The requirement is important and should be included in the solution if possible, but it's not mandatory to success.
- Could: It's a desirable capability, but one that could be deferred or eliminated. Implement it only if time and resources permit.
- Won't: This indicates a requirement that will not be implemented at this time but could be included in a future release.

B. Ambiguidade do "Won't"

2. Foi esclarecido o significado da classificação "Won't"?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)



The MoSCoW scheme changes the three-level scale of high, medium, and low into a four-level scale. It doesn't offer any rationale for making the decision about how to rate the priority of a given requirement compared to others. MoSCoW is ambiguous as to timing, particularly when it comes to the "Won't" rating. "Won't" could mean either "not in the next release" or "not ever." Such distinctions must be made clear so that all stakeholders share a common understanding of the implications of a particular priority rating. The three-level scale described previously, which relies on analysis of the two dimensions of importance and urgency, and focuses specifically on the forthcoming release or development timebox, is a crisper way to think about priorities. We don't recommend MoSCoW

3. Todos os envolvidos entendem se significa "não será implementado nesta release" ou "não será implementado nunca"?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)

The MoSCoW scheme changes the three-level scale of high, medium, and low into a four-level scale. It doesn't offer any rationale for making the decision about how to rate the priority of a given requirement compared to others. MoSCoW is ambiguous as to timing, particularly when it comes to the "Won't" rating. "Won't" could mean either "not in the next release" or "not ever." Such distinctions must be made clear so that all stakeholders share a common understanding of the implications of a particular priority rating. The three-level scale described previously, which relies on analysis of the two dimensions of importance and urgency, and focuses specifically on the forthcoming release or development timebox, is a crisper way to think about priorities. We don't recommend MoSCoW

C. Concentração Excessiva em "Must"

4. Foi evitada uma concentração excessiva de requisitos na categoria "Must"?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)

MoSCoW in practice

One consultant described how a client company actually practiced the MoSCoW method on its projects. "All the action centers around getting an 'M' for almost every feature or requirement that is captured," he said. "If something is not an 'M' it will almost certainly not get built. Although the original intent may have been to prioritize, users have long since figured out to never submit something that does not have an 'M' associated with it. Do they understand the nuanced differences between S, C, and W? I have no idea. But they have figured out the implications of these rankings. They treat them all the same and understand their meaning to be 'not happening any time soon'."

5. A distribuição das prioridades parece razoável?

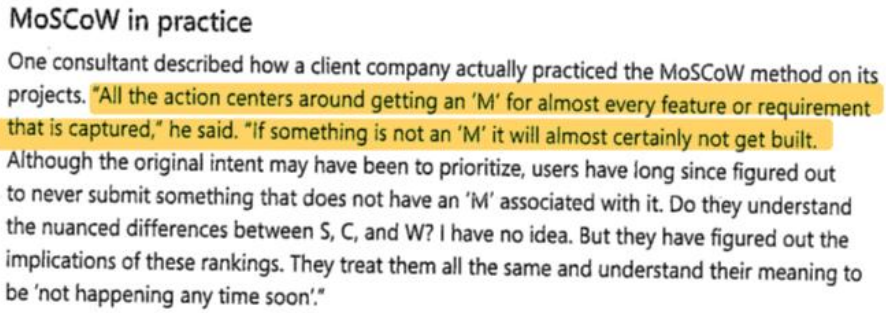
(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)

MoSCoW in practice

One consultant described how a client company actually practiced the MoSCoW method on its projects. "All the action centers around getting an 'M' for almost every feature or requirement that is captured," he said. "If something is not an 'M' it will almost certainly not get built. Although the original intent may have been to prioritize, users have long since figured out to never submit something that does not have an 'M' associated with it. Do they understand the nuanced differences between S, C, and W? I have no idea. But they have figured out the implications of these rankings. They treat them all the same and understand their meaning to be 'not happening any time soon'."

6. A maioria dos itens foi classificada como "Must" para garantir sua implementação?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)



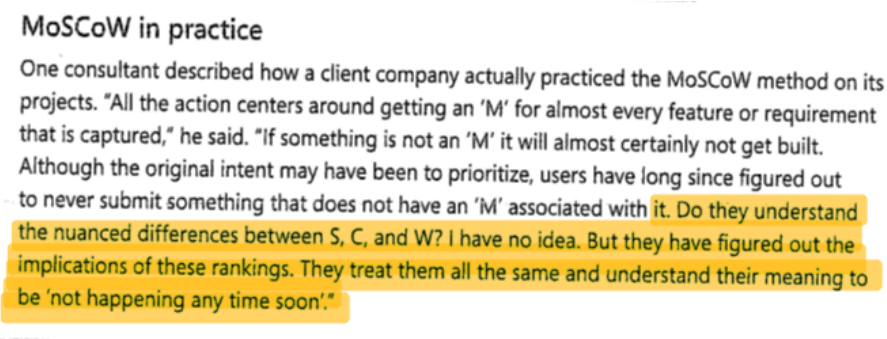
MoSCoW in practice

One consultant described how a client company actually practiced the MoSCoW method on its projects. "All the action centers around getting an 'M' for almost every feature or requirement that is captured," he said. "If something is not an 'M' it will almost certainly not get built. Although the original intent may have been to prioritize, users have long since figured out to never submit something that does not have an 'M' associated with it. Do they understand the nuanced differences between S, C, and W? I have no idea. But they have figured out the implications of these rankings. They treat them all the same and understand their meaning to be 'not happening any time soon'."

D. Compreensão das Nuances

7. Há evidências de que os stakeholders compreenderam as diferenças sutis entre "Should", "Could" e "Won't"?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)



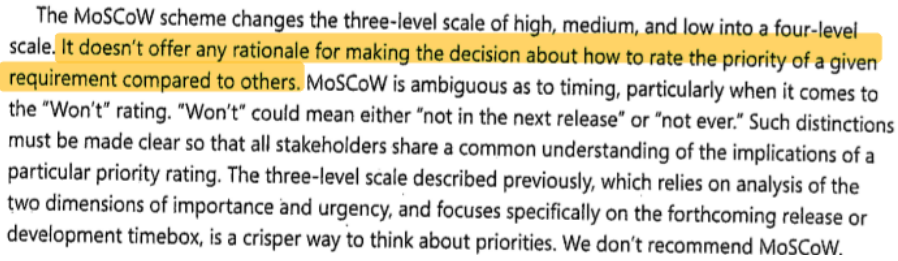
MoSCoW in practice

One consultant described how a client company actually practiced the MoSCoW method on its projects. "All the action centers around getting an 'M' for almost every feature or requirement that is captured," he said. "If something is not an 'M' it will almost certainly not get built. Although the original intent may have been to prioritize, users have long since figured out to never submit something that does not have an 'M' associated with it. Do they understand the nuanced differences between S, C, and W? I have no idea. But they have figured out the implications of these rankings. They treat them all the same and understand their meaning to be 'not happening any time soon'."

E. Racionalidade da Priorização

8. A técnica MoSCoW foi utilizada apenas como um esquema de classificação?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)



The MoSCoW scheme changes the three-level scale of high, medium, and low into a four-level scale. It doesn't offer any rationale for making the decision about how to rate the priority of a given requirement compared to others. MoSCoW is ambiguous as to timing, particularly when it comes to the "Won't" rating. "Won't" could mean either "not in the next release" or "not ever." Such distinctions must be made clear so that all stakeholders share a common understanding of the implications of a particular priority rating. The three-level scale described previously, which relies on analysis of the two dimensions of importance and urgency, and focuses specifically on the forthcoming release or development timebox, is a crisper way to think about priorities. We don't recommend MoSCoW.

9. Foi realizada uma análise racional que justifique por que um requisito é "Must" e outro é "Should"?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)

The MoSCoW scheme changes the three-level scale of high, medium, and low into a four-level scale. It doesn't offer any rationale for making the decision about how to rate the priority of a given requirement compared to others. MoSCoW is ambiguous as to timing, particularly when it comes to the "Won't" rating. "Won't" could mean either "not in the next release" or "not ever." Such distinctions must be made clear so that all stakeholders share a common understanding of the implications of a particular priority rating. The three-level scale described previously, which relies on analysis of the two dimensions of importance and urgency, and focuses specifically on the forthcoming release or development timebox, is a crisper way to think about priorities. We don't recommend MoSCoW.

F. Consciência sobre as Limitações da Técnica

10. A equipe do projeto foi informada sobre as limitações e críticas da técnica MoSCoW (como a falta de racionalidade e a tendência ao uso indevido) antes de sua adoção?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 6)

The MoSCoW scheme changes the three-level scale of high, medium, and low into a four-level scale. It doesn't offer any rationale for making the decision about how to rate the priority of a given requirement compared to others. MoSCoW is ambiguous as to timing, particularly when it comes to the "Won't" rating. "Won't" could mean either "not in the next release" or "not ever." Such distinctions must be made clear so that all stakeholders share a common understanding of the implications of a particular priority rating. The three-level scale described previously, which relies on analysis of the two dimensions of importance and urgency, and focuses specifically on the forthcoming release or development timebox, is a crisper way to think about priorities. We don't recommend MoSCoW.

G. Conexão com os Objetivos de Negócio

11. A classificação de um requisito como "Must" está claramente justificada por sua contribuição direta para os objetivos de negócio do projeto?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 4)

In reality, some system capabilities are more essential than others from the perspective of satisfying business objectives. This becomes apparent during the all-too-common "rapid descoping phase" late in the project, when nonessential features are jettisoned to ensure that the critical capabilities ship on schedule. At that point, people are clearly making priority decisions, but in a panicked state. Setting priorities early in the project and reassessing them in response to changing customer preferences, market conditions, and business events lets the team spend time wisely on high-value activities. Implementing most of a feature before you conclude that it isn't necessary is wasteful and frustrating.

H. Priorização como Processo Contínuo

12. O processo de priorização MoSCoW foi estabelecido como uma atividade contínua, a ser revisitada periodicamente, em vez de um evento único no início do projeto?

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 3)

On every project, a project manager must balance the desired project scope against the constraints of schedule, budget, staff, and quality goals (Wiegers 1996). One way to accomplish this is to drop—or to defer to a later release—low-priority requirements when new, more essential requirements are accepted or when other project conditions change. That is, prioritization is a dynamic and ongoing process. If customers don't distinguish their requirements by importance and urgency, project managers must make these decisions on their own. Not surprisingly, customers might not agree with a project manager's priorities; therefore, customers must indicate which requirements are needed initially and which can wait. Establish priorities early in the project, when you have more flexibility for achieving a successful project outcome, and revisit them periodically.

(WIEGERS, Karl; BEATTY, Joy. Software requirements. 3. Ed, Pág. 4)

In reality, some system capabilities are more essential than others from the perspective of satisfying business objectives. This becomes apparent during the all-too-common "rapid descoping phase" late in the project, when nonessential features are jettisoned to ensure that the critical capabilities ship on schedule. At that point, people are clearly making priority decisions, but in a panicked state. Setting priorities early in the project and reassessing them in response to changing customer preferences, market conditions, and business events lets the team spend time wisely on high-value activities. Implementing most of a feature before you conclude that it isn't necessary is wasteful and frustrating.