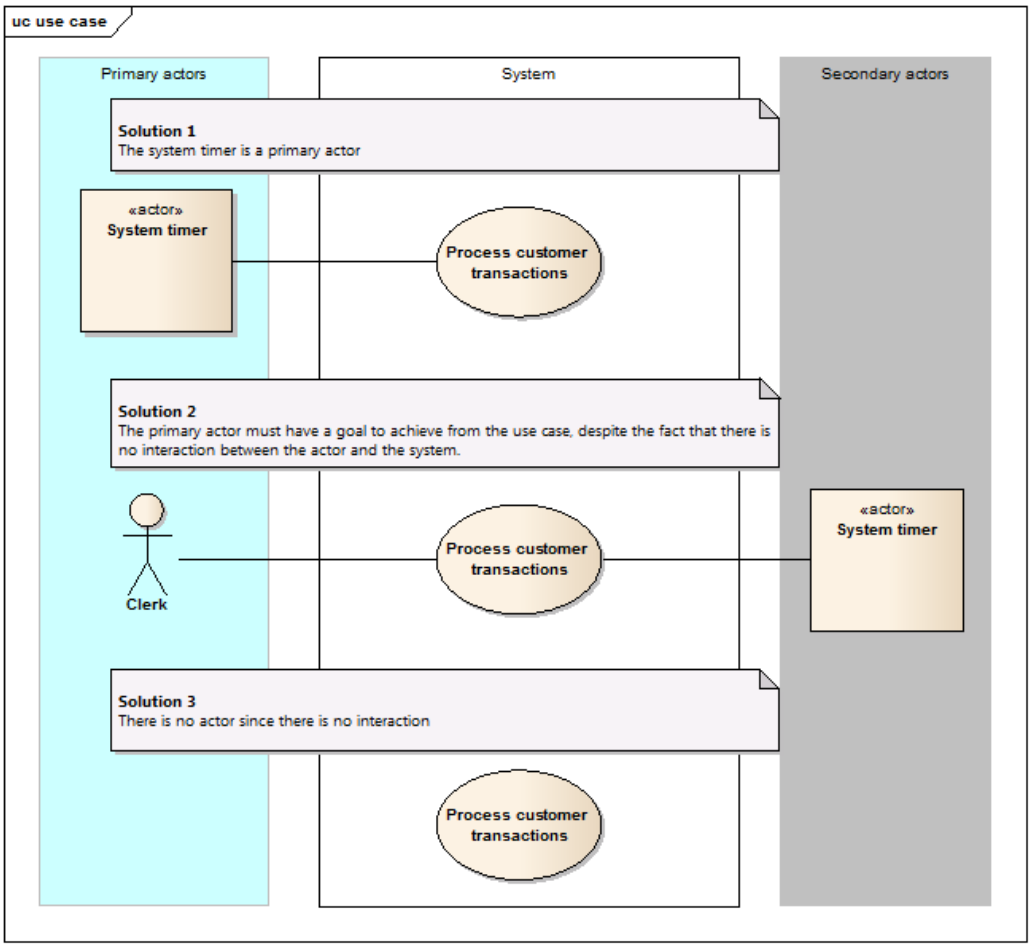
What actor should I use for scheduled use cases?

Author: Guillaume FINANCE (gfinance@objetdirect.com)
UML & SysML Consultant at Objet Direct, Sparx Systems authorised reseller, VAR, and training partner in France.

This article deals with a recurring question in use case modelling: given a use case that's automatically triggered by the system clock or timer, what actor should be used? To illustrate the options given in this article, I've chosen the example of a system that processes new card transactions on a daily basis, aim of the use case "Process customer transactions".

In this context, it is common use to define as the primary actor the system's clock or timer, since this actor triggers the use case. Is it the only approach or are there alternatives? The following use case diagrams shows 3 alternatives.



Solution 1: the "System timer" is a primary actor

Since a scheduled task is run internally by the system, it doesn't provide any visible outcome from a **black box perspective**. However since this task matches a functionality, it can be considered as being in the system scope.

With this option, the system timer (or scheduler) is an internal feature that's taken outside of the system (from a black box perspective) by representing it as a primary actor based on the following: **the system timer triggers this use case on a periodic basis** (e.g. daily, every minute, etc.). It may argued that the system timer doesn't completely fulfil the primary actor definition since it doesn't have any goal to achieve from the use case. However when there is no trigger defined, the first action from the primary actor can be considered as the triggering event.

This supports the choice of modelling the system timer as the primary actor, which is common practice in UML, providing the advantage of specifying unambiguously that all use cases associated with the System Timer primary actor are triggered on a scheduled basis.

A "system timer" is suitable in the use case analysis since it doesn't define how this functionality will be implemented. As the later design stage, it will be possible to choose a system clock or any other mean to achieve the role held by the system timer actor.

Solution 2: the primary actor must have a goal to achieve

An alternative involves identifying the primary actor that has a goal in executing the use case.

In the customers' transaction processing example, the Clerk is the actor who has a **goal** in executing the "Process customer transactions" use case, i.e. getting an up to date list of the processed transactions for all customer accounts.

What makes this use case diagram different from most use cases is the **lack of interaction between the primary actor and the studied system**. If applicable, it is also possible to model the use case's **triggering event** through a **secondary actor**, the system timer. The use of a secondary actor easily communicates via the use case diagram through the use case details that the associated use case runs on a periodic basis.

If the system timer isn't used, it is recommended to specify as use case trigger, e.g. "the system timer starts this use case".

Solution 3: the use case has no actor

This last option involves specifying a use case without any actor. This is UML compliant since we specify a use case that hasn't got any interaction with the external environment.

However so we don't miss the "scheduled nature" of our use case, a trigger can be defined with a reference to the internal system timer. If on the contrary no trigger is set, a use case on its own may be more difficult to understand for some, or even lead to some questions such as "is this diagram complete? I thought you hadn't finished your analysis".

Conclusion

Personally I've always used the first option i.e. I identify the system timer as the primary actor.

The exercise carried in this article was intended to discuss and compare the identified options, which should help weighing the impact from choosing either option or any other that may exist. An important element to be taken into account should be the impact on the communication with the stakeholders and the team through the use case specifications, and how well they will be understood. Personal views and each context will of course affect the final decision.