|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decision rule** | **Rule #** | **Improvement** | **Additional decision factors** | **Quotes** |
| IF the framework upgrade workload is feasible OR the framework upgrade process is automated, THEN a startup team should upgrade  the development framework  to the latest version | 10 | First, check the probability  of changing the framework.  If it is high, skip the  upgrade. If it is low, then check the  existing factors in the model  (upgrade process and  workload). Finally, before  going to skip the upgrade,  check # of new versions of  the framework. If it is more  than 2, then do the upgrade. | * # of new versions   of the framework (ahead from the current version)   * The probability of   changing the  framework | *“They might say no we have other important tasks. This will always happen, and developers will spend a lot of time without doing the upgrade. Then, they want to develop a new feature, and face the evolution restriction. P-2”*  *“We should do the frequent update considering the workload, but If we have two or three new versions, it means we should give higher priority to do the update. P-13”*  *“I think it really depends on the*  *scenario. Usually, we should frequently upgrade, there is no discussion there. In one scenario, I was about to upgrade the framework, but I thought about that we are going to stop using it fully. So why we go through the hassle of the upgrade. P-17”* |
| IF financial resources are  available, THEN a startup  team should gradually  restructure the software (15)  ELSE a startup team  should  develop the upcoming  features on a new dev  infrastructure (16) | 15  16 | Additional factors  influence the effort needed  to perform the software  restructuring, and hence, the  financial resources. | * Current code quality * The differences between the current and new infrastructure | *“Let say if you are using React and you want to move to Next.js, you can easily restructure the product without thinking of financial. But if you are working on React and you want to catch a different framework structure, then it will*  *be a completely new infrastructure and repository. P-17”* |
| IF the growth of the  development team is high  (or expected to be high  soon), THEN a startup team  should document the  requirements of some  important functions | 13 | This rule should only be considered for the high level documentation (e.g., modeling). Other low level documentation (e.g., code comments, development setup, etc.) should be improved gradually. | **--** | *“I would say like the basic*  *documentation should be at the beginning. At least the setup and local setup procedures. But the very detail and high-level documentation should be at the team growth level, yes. P-13”*  *“I would say even before going to*  *team growth. P-17”* |
| IF financial resources are  available, THEN a startup  team should gradually  replace some scripting with  strongly typed language | 11 | This rule should not be used if the team is familiar with a strongly typed language. | **--** | *“It does not make sense to intentionally do it all scripting then replaying it later. P-2”*  *“I would recommend if we do have experience working with strongly typed language, then we should prefer it from the beginning. P-13”* |