Decision List

This decision list belongs to a case study entitled: “Empirical Insights into the Evolving Role of Architects in Decision-making in an Agile Context”.

The decision types in italic are taken from Drury [1]. The letters after the numbers indicate the different perspectives of the decision types where M = management, D = design, and U = user.

Initiating phase
1. M Decide to elaborate an idea into an epic
2. M Decide to put an epic in the portfolio
3. D Decide on the solution direction of the project/product
4. D Decide on the content of the FSA
5. M Decide on the budget of a project
6. M Decide on the time scope of a project
7. M Decide if the initial scope of the project meets the budget
8. D Decide if the scope of the project during the project still meets the budget + time scale
9. D Decide on the content of the PSA
10. D Decide on the application architecture
11. D Decide the application programming interface design
12. D Decide if the PSA is final
13. D Decide if the final PSA aligns with the FSA
14. M Decide the t-shirt size of a story container
15. U Decide the minimal viable product, with all its perspectives
16. M Decide who is suitable for the team
17. M Decide for someone (extern or intern) to join the team during the project
18. U Decide on functional requirements
19. U Decide on non-functional requirements
20. D Decide which technologies, systems and platforms are used in the project
21. M Decide the roadmap of the project
22. D Decide the vision of the project

Operational phase
23. M Decide iteration goals and scope (user stories and tasks)
24. U Decide if the backlog meets the business requirements
25. M Decide priorities within the iteration
26. M Decide which people will be available
27. U Decide if new backlog items are still in line with the functional requirements
28. M Decide capacity for team members
29. D Decide how a functional requirement can be implemented
30. D Decide how a technical requirement can be implemented
31. D Decide how systems and platforms need to align in theory
32. M Decide if a (new) requirement meets the scope (time, budget, minimal viable product)
33. M Decide if a user story meets the scope (time, budget, minimal viable product)
34. M Decide on the balance of velocity and story points taken into a sprint
35. M Decide who will work on what
36. M Decide if one requirement is more beneficial to the outcome than another
37. M Outweigh one user story for another
38. M Outweigh one requirement for another
39. M Outweigh one feature for another
40. M Outweigh one task for another
41. M Outweigh one epic for another
42. M Decide if a task is relevant for a user story
43. M Decide what goes in the backlog
44. M Decide the priority of the features
45. M Decide the priority of the bugs
46. M Decide the approach to delivering a story
47. M Decide how many story points a story needs to have
48. U Decide if a user story is valuable for the business outcome
49. M Decide the input and subjects in a pre-refinement session
50. M Decide task estimates
51. M Decide what user stories are involved in a feature
52. M Decide if user stories require more discovery work
53. M Decide definition of when a story is "done" (i.e. completed, when to accept/reject story)
54. U Decide if a ‘done’ user story has met the technical requirements
55. U Decide if a ‘done’ user story has met the non-functional requirements
56. M Decide if a task is suitable for the user story
57. M Decide if a task fits in the scope of the project
58. D Decide if a task meets the architecture criteria
59. M Decide to split or combine user stories

Execution phase
60. M Decide whether iteration scope should be changed (i.e. reprioritize tasks, accept new tasks)
61. D Decide if a (part of a) solution of a previous project is applicable for this project
62. D Decide if the project is in line with the architecture criteria
63. M Decide definition of when a feature is “done” (i.e. completed, when to accept/reject feature)
64. M Decide who will pair together for paired programming
65. D Decide that a part of a user story will be technical debt
66. D Decide if a technical direction or solution is too comprehensive
67. D Decide if a functional direction or solution is too comprehensive
68. M Decide if a backlog item/task is done
69. D Decide on a technical solution
70. D Decide the interface design
71. D Decide how to implement technical functionality
72. D Decide how to implement functional functionality
73. M Decide if an issue need any architectural input before making a decision
74. M Decide if a decision needs to go to the board
75. D Decide if decisions are in line with the architecture criteria and guidelines
76. M Decide if the project is still funded
77. D Decide when to commit code
78. D Decide what tests to create
79. M Decide whether to add/remove/change acceptance criteria
80. D Decide on the architecture/design for technical functionality

Reflection
81. U Decide if delivered product meets customer expectation
82. U Decide on the acceptance and test criteria of a user story
83. U Decide if a user story is done
84. M Decide whether story estimates need to be modified
85. M Decide whether to continue with the project
86. U Decide whether to accept the iteration content
87. M Decide what stories and defects be scheduled for next iteration, particularly if not completed
88. U Decide that the minimal viable product is ready for release
89. M Decide if the scope of the project needs to be changed/adjusted
90. D Decide if the new scope fits in the FSA
91. U Decide if the project is done
92. U Decide that a project can be operated
93. U Decide that a project has met his requirements
94. D Decide that a project has met the (architecture) criteria

Retrospective
95. M Decide what to improve during the next iteration
96. M Decide what went well to continue during next iteration
97. M Decide what new things the team will try in next iteration
98. M Decide root cause if team did not meet its iteration goal
99. M Decide priorities for things to address in future iterations
100. M Decide issues that will most influence team success
101. M Decide whether and how to measure team metrics