1.

A. For small-scale software-intensive project.

- UseCase Diagram is necessary because this defines the use case for user group which is important regardless of the scale of the project.

- Class Diagram is necessary because even the small project could be divided into classes and class diagram could make the project to be more organized and clearer.

B. For small to medium scale software-intensive project.

- UseCase Diagram and class diagram are necessary for the same reason.

- Sequence Diagram should be important in this case because medium scale project may involve in some complex operation logics and it might be hard design this without a clear sequence clarification.

- Deployment Diagram is also necessary when the user group become larger and larger. For small project, this is not necessary because small projects usually do not have too much users or not require internet connections. However, for medium project, performance should be considered more.

- Requirement Diagram: The requirements of a medium project are usually not that simple and it could be hard to express them orally.

C. For large scales software-intensive project.

-UseCase Diagram, Class Diagram, Sequence Diagram, Deployment Diagram, Requirement Diagram are necessary for the same reason.

- Package Diagram is necessary as well because larger projects are usually involves many packages.

- Composite Structure Diagram: Since the scale become larger, and class diagram could be complex. this diagram could help to demonstrate the structure of the system from a higher perspective.

D. For small scales software embedded project

- Similarly, UseCase Diagram and Class Diagram should be necessary in this case.

- State Machine Diagram is very important for embedded project because the state of hardware is crucial for embedded software and will decide the entire system.

E. For medium software embedded project.

- Use case diagram, Class Diagram, Sequence Diagram, Deployment Diagram, Requirement Diagram are necessary for the same reason

- State Machine Diagram is important as well.

F. For large scales software embedded project.

- UseCase Diagram, Class Diagram, Sequence Diagram, Deployment Diagram, Requirement Diagram, Package Diagram, Composite Structure Diagram are necessary for the same reason.

- State Machine Diagram is important similarly.

2.

A. Model Integration.

I think making an overview model from a higher perspective could be a solution. For example, the type of information of several sub-models should be included in this overview model. And clearly define how different models connect to each. What is more, the data should be marked by unified and clear identifier.

B. Authoritative Data.

Define the priority of information. For example, the information of overall model should have the highest priority. After the complement of model designing, the developer should develop the project following the guidance of the model so that the conflict could be minimized.

C. End-to-end Solution

I think there should also be an individual or a team who will always focus on end-to-end testing. Testing plan could be done at the very beginning of the project which will cover all the user scenario. In addition, tester should own the permission or the ability to coordinate the team if there are conflicts between the works.

D. Intellectual Property and Security Protection

During the modeling of the project, designer should add a security layer to the entire project to protect the system from being attacked. Some basic rules and principles should be pre clarified. For example, all the form should be checked in the browser before submitting to the server. Some input with illegal characters should be forbidden. After the development, team could also ask for help from professional team and apply penetration testing to the project.

E. Workforce skills/Training

Leader or the coordinator of the team could hold training to the entire team. Coordination work should be emphasized. For example, feedback of the team should be collected regularly during the training and correct any mistakes as soon as possible to avoid conflict in the future.

3.

1. Project planning
   1. Kepner-Tregoe Model could be used in project planning [1]. This model could help designer to manage the definition and prioritization of the requirements and alternatives and help designer to make a better and reasonable project plan
2. Analysis of alternatives
   1. Deployment diagram could help developer to select the best alternatives. Since the alternative might conflict with any component of the project, research on the compatibility of the candidate and all the other components in the deployment diagram could be helpful.
3. Business projection
   1. Deployment model will support business analysis activity because in this model, hardware cost is clearly declared which is important in cost estimation
4. Identification of use case or user scenarios
   1. Interaction Diagram could be helpful in identifying the use cases. After the gathering of requirement, prototyper could identify the user scenario and all possible interaction between the user and the system. After this, the user case could be identified by splitting the interacting operations.
5. Requirements negotiation
   1. Win win negotiation is a good model for requirements negotiation. In this process, the requirement of the client and the limitation of the developer could be fully considered at the same time.
6. Identification of logical architecture
   1. Use case diagram could be referred when designing the logical architecture of the system. All the logical components could be design by the guidance of use cases because all the use cases should be realized.
7. System performance analysis
   1. Activity diagram could support the system performance analysis. Because tester could apply the performance testing in server user scenarios which is included in activity diagram.
8. Risk analysis
   1. Risk management report. Like what we did in 577b course, we could maintain a table to track all risks. For each risk, the possibility and potential damage should be evaluated separately. As a result, it will be easy for members to identify the most important risk and mitigate it as soon as possible.
9. Technical debt management
   1. Technical debt report we are using currently is a good model. In the report, we classified the technical debt and make it easy to manage.
10. Testing
    1. Use case diagram could be used to support testing works. Most test cases could be generated from the use case. For each use case, we could design several test cases to check this use case like what we did this semester.
11. System deployment
    1. DevOps Model could help developer to deployment. By using DevOps Model, continuous deployment will allow the system to be updated and deployment quickly. The efficiency of the team could be increased as well.
12. System maintenance
    1. DevOps Model will support system maintenance too.
13. System retirement
    1. Deployment diagram could support retirement activity because all hardware is included in this diagram. We should make sure all the hardware components are retired.

Reference:

[1]<https://www.brighthubpm.com/project-planning/97808-examples-of-project-planning-models/>