# Process Grid

This grid is provided to help you define ten key processes for your team. Less formal processes are flexible and have low overhead, but increase the probability that problems will affect project success. More formal processes have more overhead and can be irksome, but reduce the impact of problems. What is a good choice for you depends on the project characteristics as well as your team’s work style and preferences?

| **Process Area** | **🡨------ Less formal --------------------------------------------------------------------- More formal------------------🡪** | | | |
| --- | --- | --- | --- | --- |
| Team meetings | Held whenever there is a need, objectives not always specified | Meetings scheduled in advance on demand, specified objectives | Regularly scheduled meetings, initial (flexible) agenda, decisions noted. | Meetings with fixed agenda, strict time limits, meeting minutes. |
| Meeting format | Free-form discussions | Focused orderly discussions, no moderator | Meeting facilitator to keep discussions on track | Time limits on discussions, defined decision-making procedures |
| Version Control | No version control. Everyone works off a single copy of the source available to all team members. | Use of version control tool. Individual workspaces. Checkout / update / commit from repository with change conflict resolution. | Version control tool as before, but with tagging of significant versions / milestones for fallback and recovery. | Upfront agreement on who is allowed to modify what files & when. Key product configuration items identified upfront. |
| Quality management | Team members individually try to produce quality outputs | Some coding standards, informal reviews (“look over others’ code when asked”) | Defined coding standards, some reviews/inspections, checklists, tracking of problems to closure. | Inspections/reviews of all items. Formal release checklists with signoffs. Consistent data gathering and analysis. |
| Activity planning | Team members identify tasks as they go along and get them done. | Weekly meetings where tasks are identified and allocated to people. | Upfront task list created, updated when requirements change. Effort estimation and effort tracking. | Upfront planning and task allocation using effort estimates and task dependencies. Periodic revision of estimates based on experience. |
| Activity tracking | Progress discussed informally. | Weekly meetings discuss progress, plan for next week. | Progress tracked using tracking charts. | Progress tracked against plan. Re-planning when progress and plan diverge. |
| Integration and build consistency (coordinate changes to get consistent versions for builds) | Team members get together informally, integrate code and debug it. Informal coordination to ensure consistent builds. | Integration responsibility allocated. Integration approach chosen at implementation time. Coordination of changes during team meetings. | Defined strategy for integration. Activity planning done with integration in mind. | Predefined integration sequence and approach, taken into account during planning phase itself. |
| Testing | Team members test their own code in their own way. | Test cases defined and run before each release. | Test cases chosen carefully to ensure coverage. Unit testing by developer, separate integration and system testing. | Formal test plan, coverage analysis. Formal test tracking and bug fix tracking. Regression testing whenever changes are made. |
| Risk management | Reactive risk management (“handle it when it occurs”) | Informal proactive risk management (“take some preventive steps”) | Risk identified upfront, mitigated where possible. | Formal identification and prioritization of risks. Periodic review and tracking. Identified owner for each risk. |
| Requirements Management | Make changes to requirements as needed | Change notification: Review reqts. changes with customer, notify team | Change control: Analyze reqts. changes for technical and schedule impacts. Decide whether to do now or defer. | Maintain traceability information and versioning information for requirements. Ensure propagation of requirements changes through project documents |

Highlight above the processes that you plan to use by selecting the appropriate cell from each row and changing the cell background color to yellow.

Then, for each of the ten areas, give a brief two to four sentence rationale as for why you selected the process you did. Couch your answer in terms of why you *didn’t* select one of the adjacent processes.

|  |  |
| --- | --- |
| Team meetings | Weekly meeting will be a major communication and catch up time. It is essential for a smooth project |
| Meeting format | Having a plan for the meeting and a list of things to discuss keeps up on track and makes sure we make proper use of our important time |
| Version Control | This is the easier way for us to use the version control without over managing |
| Quality management | Don’t want to waste our short time together on scheduled code reviews. But a standard will be set for easy inter-team code use |
| Activity planning | Meetings are the best time to discuss and plan our upcoming activities |
| Activity tracking | Same as planning, in that meetings it will be an essential to keep up to date with each other |
| Integration / build consistency | The structure of the app will be decided before it is coded. We want to implement the features in a scalable way |
| Testing | Testing will be more important as the app is closer to completion but we don’t want to fill all our time with hardcore testing |
| Risk management | We will be identifying risks when we plan the structure of the project |
| Requirements Management | We will be reviewing what is required at every meeting and make changes as needed. |