Model Selection

We have selected the Agile Scrum model for our Gadget Shop Management System because it is an adaptive, quick, and self-organizing process that allows us to handle the unpredictable nature of software development. As shown in our project visualization, the work is divided into Sprints—short iterative cycles lasting one to four weeks—where we convert items from the Product Backlog into a working Product Increment. This model is highly relevant to our project because it treats the development phase as a black box where changes are expected. Instead of trying to plan the entire gadget shop system perfectly at the start, we can use Sprint Planning Meetings to prioritize the most important features—like the sales module or inventory tracking—and deliver them in small, manageable pieces. If we discover new requirements or need to fix a bug, the Sprint Backlog allows us to adapt quickly without disrupting the whole project, ensuring we always have a potentially shippable product at the end of every cycle.

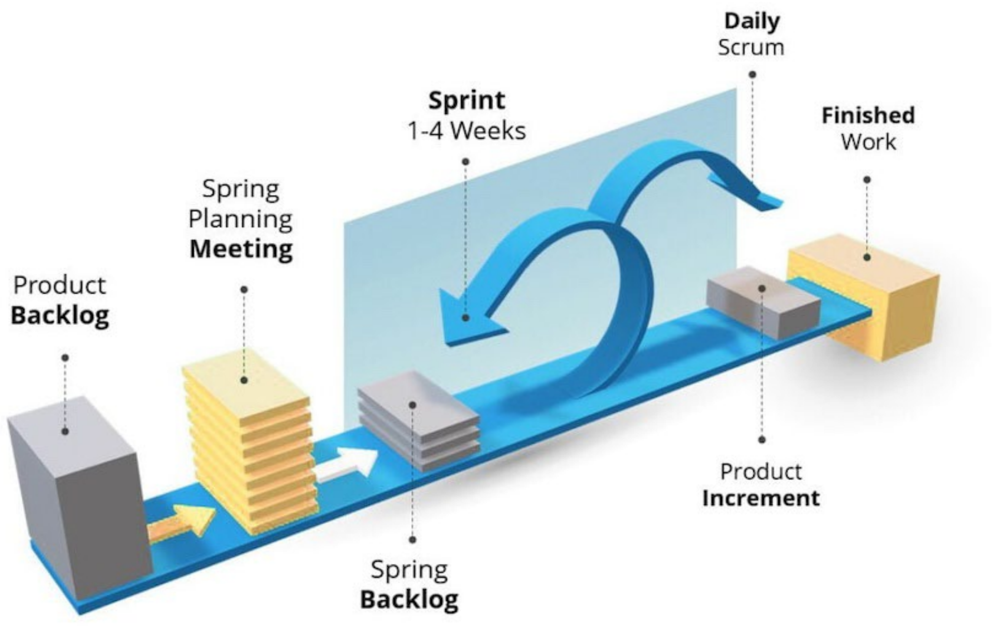

Fig: Flow of Scrum process Model

In contrast, other process models were not suitable for our specific needs. We rejected the Waterfall and V-Model because they suffer from inflexible partitioning, where a phase must be fully completed before the next begins, making it difficult to respond to changing requirements since backtracking is not allowed. These models are only appropriate when requirements are well-understood from the start, which is rarely the case in student projects. Extreme Programming (XP) was also unsuitable because it requires an on-site customer to be available full-time and relies heavily on pair programming, which is difficult for us to organize logistically. RAD was not chosen because it requires frozen requirements within each increment and is designed to deliver a system in a very tight 60–90 day window, which is too rigid for our learning curve. Finally, we avoided FDD and DSDM; FDD is designed to scale to much larger projects with specific roles like Chief Architects and Class Owners that we do not have, while DSDM enforces strict fixed time and resources which limits the flexibility we need to explore and learn new technologies.