To develop our system (the Scientific Instruments App), with our requirements and constraints, our team has selected the Spiral development model.

Generally, the software development model is chosen in response to the circumstances of the software’s development. Our client is a real-world client – Professor Jason Donev, who is an individual and understandably has a busy schedule. He can meet up with us around once a week, but it would be difficult and unreasonable to remain in constant communication. There are difficulties for us as well as students going to classes on a set schedule, thus any method which requires constant client input and interaction on every aspect of development appears unsuitable. This difficulty eliminates the Scrum model, due to it being an agile model which requires such constant communication.

Next, we must consider our requirements, or lack there of. As we currently understand it, our client has a general trajectory as to what sort of software product he wants, but not the full specification of what exactly is desired down to the last detail. Furthermore, future improvement ideas have been discussed and thought of as early as the very beginning of the project, such as integrating a similar but not identical device – the Geiger counter. These facts strongly point away from incremental models, towards iterative models.

Furthermore, we are all inexperienced in this project, not knowing the full details of what we might need to learn to complete this project, thus we cannot easily and evenly divide up the project – ruling out the concurrent model directly. The waterfall model requires a more intensive discussion with the client period before dropping off the radar and just building the software, where the first half might demand more attention than can be reasonably provided and the latter half squanders our ability to contact him for more questions, clarifications, and demonstrations of prototypes. As such, the Waterfall model is no fit either.

Phased release might fit, but it suffers from the fact that it is an incremental model when we do not actually have all the requirements detailed and specified down to the last. It also suffers from the same intense requirements gathering phase needed in the beginning, and the rolling out of the releases do not actually factor in input from the client, again squandering a major advantage of being able to meet with and discuss the project with our client. As such, phased release is also not a reasonable model.

We are now left with the Spiral and Opportunistic models. Of these, the Opportunistic model doesn’t factor in client communication, and is not a clear and reliable software development plan for any external client or larger project. The scope of our project is not so small that the opportunistic model can work, since an app with multiple parts is much larger than projects suited for opportunistic, such as brief Computer Science class assignments or small segments of code. This leaves the Spiral model.

Rather than being the last one standing, the Spiral model is good fit for its own reasons. Consider our criteria: Brief meetings approximately once a week, undefined project requirements, possibilities for future upgrades. The spiral model’s quick cycles complete with risk assessment, planning and development stages appear to fit perfectly for this setup. In these brief meetings, we can gather requirements from our client and demonstrate the project as it currently stands; as it is an iterative model we can accommodate for changes in the project requirements and for future upgrades. As such, Spiral is a near perfect fit for our situation, and as such will be used.