1. The V Model is a redesigned version of the waterfall model where each development phase goes through a verification process before moving onto the next. It is a Software Development Life Cycle Model (SDLC) that follows a sequential order in a manner that can be drawn in a V-like shape, hence the name. This type of model is also called the Verification and Validation model.
2. The V-Model is most acceptably used in smaller projects that have specific, rigid requirements that must be met. If all resources and necessary materials have been met, then this model should most likely be used.
3. System design typically consists of a full comprehensive understanding of the hardware being used for a software in development, whereas architectural design usually involves more technical aspects and encompasses the interaction/communication of all inner and outer systems. Further more, system design is comprised of different modules that make up for different functions within the software, and is often considered to be an example of high-level design.
4. The Software Testing Life Cycle (STLC):
   1. Requirement Analysis: in the first phase of the STLC, the types of tests and the methods in which they are to be conducted are identified. An automation feasibility analysis, which helps determine whether any test cases should be automated, would be conducted in this phase if applicable.
   2. Test Planning: A development team must write up a planning document as well gathering all necessary testing resources and materials needed before any other test processes can be carried out. Not only that, but assigned roles have to be determined so each developer knows what specific tests they are to perform. Cost estimations for said tests would also be calculated in this phase.
   3. Test Case Development: this phase entails the creation of test cases and scripts that detail the processes and tasks a software will be tested on. They are designed around the individual unit requirements that were specified in the first phase of this life cycle. This phase provides the testing team with necessary test data.
   4. Test Environment Setup: the environment setup phase refers to the determination of the hardware and software conditions in which the tests will be executed in, and can be often conducted in parallel with the previous phase.
   5. Test Execution: at this point in the cycle, the testing team is finally prepared to carry out the previously planned testing procedures. Test scripts are executed and maintenance is performed when needed. If any bug reports were to be generated, then the scripts would be sent back to the development team for patching, and testing would be executed again.
   6. Test Cycle Closure: in the final phase of the STLC, a closure report and other important testing metrics would be created. These metrics typically involve overall time spent on testing, the overall cost, the specific units that were tested as well as the units in which bug fixing was conducted. All important testing information must be documented in this final step.
5. In regards to the V-Model, it is important to stress the difference between validation and verification. Validation refers to assuring the customer’s specific software/system requirements, and would be considered a more external task. In contrast, Verification is more of an internal task that consists of assuring a developed software has met the needs and requirements that the customer has previously specified; this verification task might be completed during the testing life cycle.
6. The incremental model is a method of the software development life cycle in which each task is performed in a repetitive sequence. To further elaborate, each step in the incremental model is added one by one, and is only added once the cycle has begun to repeat. Once the first increment is completed, for example, it is analyzed by the customer. Only if the customer approves can a plan for the next increment start development. This process is extremely helpful if a development team is short on staff.
7. When compared to the V-Model, the incremental model can prove to be much more efficient. Instead of having to work backwards as you would with the V-Model, the incremental model assures the proper functionality of each highest-priority component of the developing software. The V-Model can slow down the testing process, as bugs can appear in any individual component, as they are all being examined at once.
8. The waterfall model would be more applicable than the incremental model if there was a large development team on hand to provide their primary focus on the product. Furthermore, the waterfall model would be more useful if the purpose of development is not heavily specified by the consumer.
9. The quote, “Each stage of the incremental model is completed following the waterfall model,” is something I agree with to a certain degree. While each development phase of the incremental model can be conducted falling the steps of the traditional waterfall model, it is typical for changes to be made during the incremental process. For example, documentation and early planning is not as critically important as it would be during the standard waterfall method. The incremental model is also not entirely linear during each module development phase. Tasks can be performed and changes can be more easily made at a later date. The waterfall model does not leave a lot of room for error.
10. The evolutionary model is considered to be a combination of both the incremental and iterative models. When a consumer wants to have access to the main features of a software, a development team will release such. However, the core features are only the skeleton of the software being developed, and future updates will be developed as time moves on, hence the incremental aspect. The importance of this model lies in time it takes for the product to slowly evolve.
11. The prototype model is a method of development in which early, tested, yet unpolished versions of a product are released to the customer for the purpose of receiving feedback and refining aspects upon request, and is something that can often be seen used in projects involving video game development. With the incremental model, however, the time before customer access is much longer, as the software is incrementally tested and only released when it can be considered truly finished.
12. One of the notable differences between the incremental model and the spiral model is the cost; the spiral model winds up being much more expensive in comparison to the incremental model, as the spiral model focuses on risk-monitoring, which can require a lot of pricey equipment.
13. In contrast to the prototype model, the spiral model puts a much higher emphasis on risk analysis, and avoiding risky solutions by shifting focus on finding safer alternative solutions. On the other hand, the prototype model leaves a much higher margin for error, as fixes can be constantly released to the customer after the initial release.
14. If the client were unsure as to the specific requirements of the of the software they wanted developed, I would probably work along the evolutionary model. This model allows for the basic bare-bone features of the software to be released to the client, allowing them to request any features they need at a later date.