Software development

Software Development If it is possible for a 15 to 20-gram product, whose main constituent costs less than ten cents, to generate annual global revenues of 220 billion dollars for its manufacturers, would one not be interested about how such product is developed? This is exactly how the software industry fares in the international market, but software development is a non-nonsense process. The following exposition shall attempt to familiarize a generally non techno-savvy audience as to the protocol involved in developing a software. The term software development is defined as a component of the software engineering methodology which involves the structuring, planning, writing, testing, delivery, and maintenance of a software product (Pfleeger & Atlee, 2010). The software development process encompasses procedures that address how the software product will be designed, created, tested, implemented and maintained (Jawadekar, 2004). Furthermore, the software development process involves the following components: (1) activities; (2) resources; and (3) constraints. All major process activities are planned out, sub-processes and their relationships between each other are identified, schedules of activities are plotted with definite start and exit points, objectives of each process is listed, and resources required for each step is allocated and adjusted according to identified constraints (Pfleeger & Atlee, 2010). The first step in the software development process involves the identification and analysis of the various system requirements. In this stage, data is gathered regarding client requirements, types of hardware and software needed during development, and manpower requirements (composition of software development team, skills needed, etc.). After data gathering, it is analyzed to determine the following: (1) functional and non-functional requirements; (2)

the relationship between external factors and the specific functions of the software being developed; and (3) important system functions (Saleh, 2009). The second step in software development involves system design which basically maps out the entire system at different levels based on the results of system analysis. Design documents produced in this stage includes: (1) high-level architectural design (software modules and interfaces); (2) detail design (data structures and algorithms); (3) database design (database schema); and (4) interface design (graphical user interface) (Saleh, 2009). The third step is implementation. This step involves transforming the information processed during system design into executable code, or simply put, creating the program based on the system design. Software developers have the option of creating an entirely new program from the ground up, or using ready-made modules and integrating these into a single software package. Code writing normally follows a set of coding standards and is based on the system specifications (Saleh, 2009). After writing the program code, it is reviewed for errors. Program code is then incorporated into software documentation either within the source code itself or through supporting documentation such as a user's manual (Sundar, 2010). The fourth step involves testing the program with a set of test inputs or cases designed to simulate a particular real-world scenario. This is done to ensure that the program works and behaves as expected. The aim of software testing is to identify errors in the program code for later correction. Software is tested based on the following parameters: requirements, structure, data, and coding guidelines (Sundar, 2010). After testing is completed, the software proceeds to the fifth step – implementation. The software is delivered to the client wherein the software is installed and tested by the

client. Once the client is satisfied with the results, the software is accepted by the client (Saleh, 2009). The last step in software development is maintenance. This step covers corrective, preventive, perfective, and adaptive maintenance activities. Corrective maintenance occurs when faults on the software are encountered. This may range from simple cosmetic changes to a user interface to complex changes in how data is processed. Preventive maintenance addresses weakness or vulnerabilities identified after implementation. This may include weakness and vulnerabilities in security and data integrity. Perfective maintenance focuses on implementation of improvements on the software created after implementation. This includes improvements on how the software handles data and the incorporation of additional functionalities. Lastly, adaptive maintenance occurs when clients need to implement the software on another platform or operating system. An example would be adapting of Windows-based software to run in a Macintosh operating system. References

Jawadekar, W. S. (2004). Software engineering: principles and practice. New Delhi: Tata McGraw-Hill. Pfleeger, S. L., & Atlee, J. M. (2010). Software engineering: theory and practice. Upper Saddle River, NJ: Pearson Higher Education. Saleh, K. A. (2009). Software engineering. Fort Lauderdale, FL: J. Ross Publishing. Sundar, D. (2010). Software engineering. New Delhi, University Science Press. Verberne, B1 (2010). Global software top 100 edition 2010: the highlights. Retrieved from http://www. softwaretop100. org/global-software-top-100-edition-2010-the-highlights