Software Aesthetics and Human Flourishing in the Making of Technology

Software applications in general and software integrated systems in particular are rapidly becoming ubiquitous parts of our personal lives (*e.g.*, smart phones, cars, appliances) as well as essential components to critical infrastructure (*e.g.*, power grids, air traffic control, national security). The demand for software systems is increasing in number, complexity, and size as the culture becomes more accustomed to the pervasiveness and benefits. This increase in demand requires additional computer scientists, information technologists, and software developers.

Often the perception of software development is seen as analogous to assembly line work (*i.e.*, menial unskilled labor). The development process is stereotyped as regimented, straightforward, and purely logical. Software requirements are captured, systematically mapped to a design, and transformed to implementation artifacts in a straightforward detached manner disassociated from aesthetics. This perception is present in the culture which is influenced by the modernist mindset and is present in industry as well as at academic institutions.

Christian thinking presents a different perspective on work in general. When God created human beings He included work as a part of His good creation. Humans were to tend the Garden of Eden. Only after the cosmic event of mankind's Fall did work become perverted and a struggle. Work is not inherently a curse but rather a good gift from God affected by the Fall. As Dorothy Sayers states, "… work is not, primarily, a thing one does to live, but the thing one lives to do. It is, or it should be, the full expression of the worker's faculties, the thing in which he finds spiritual, mental and bodily satisfaction, and the medium in which he offers himself to God."

The academic discipline of computing science can be broadly split into theoretical and applied categories. Theoretical computing science investigates the limitations of computation (*e.g.*, discovering uncomputable problems, analysing general theoretical timeliness of software). Applied computing science investigates how to create computational artifacts (*e.g.*, software, hardware) to solve a wide range of challenges in various domains (*e.g.*, biology, music, information management). Humans are creative creatures reflecting God's creativity. Applied computing science in general and software engineering in particular inherently reflect this creativity which is the focus of this paper.

The challenge arises of integrating the work of developing software systems with human flourishing and the redemption of work. Several questions arise when addressing human flourishing and the technology of software engineering. What is a Christian perspective in developing software? How can humans flourish in the development of software? Can other disciplines inform software development concerning human flourishing?

In 1994 the seminal book Design Patterns: Elements of Reusable Object-Oriented Software was published to disseminate good software design practices. The book describes several design patterns that can be used to resolve requirements within a particular context. In some sense, the book describes tradecraft for software development and documents the wisdom of experts to be leveraged by the broader software development community. The software design pattern concept was inspired by the writings of Christopher Alexander and his book A Timeless Way of Building where he

describes patterns used in architecture and buildings. When used with discernment design patterns can create elegant software designs and implementations that inherently have a quality of beauty. As one of the authors of the Design Patterns book stated, “Erich Gamma shared his joy in the order and beauty of software design as coauthor of the classic Design Patterns.”

An important area to consider regarding beauty and software development is the interaction of discipline and creativity. Some computing science students initially believe discipline and consistency in software development stifles creativity. However, the arts (*e.g.*, dancing, painting) show that consistency and discipline actually stimulate and provide channels for creativity. A script for an actor does not limit the actor’s creativity but provides a framework within which the actor can exercise creativity. This same argument can be made for consistent formatting and documentation of software systems. Creativity is needed in addressing requirements and solving problems. Consistent formatting and discipline in development provide the creative framework for software development.

Beauty in software development is evident when complex functionality is encapsulated behind a simple and elegant interface. Beauty can be found in software systems that are comprehensibility at the appropriate level of scope and context. The order and symmetry of a well-designed software system is beautiful. Computing science students need to be trained to detect and create this beauty which argues for a liberal arts education that includes instruction in the fine arts and humanities.