Technology & Engineering Core Concepts

Core Concepts are the overarching ideas that define the field of study and are the most important concepts that students must grasp to make connections and be able to transfer their knowledge to understand new situations. CCR has identified and defined nine engineering Core Concepts.

1. **Technology Is an Amplifier:** We must be mindful of the fact that technology can amplify existing dynamics and patterns, thus having huge impacts on society. These impacts can be positive or negative.

2. **Problem Solving Process**

    a. **Explore problem space:** To define the problem in the most useful way, it is important to be open to different possible ways, considering the situation from different perspectives. The ultimate users will probably not be able to articulate their problems and so there is an important step before starting to solve the problem: identifying it.

    b. **Generalizing the Problem:** The scale at which a problem is defined will affect the kinds of solutions that will be arrived at. Sometimes it is useful to solve a more general form of a problem, which then leads to solutions to other related specific problems. (e.g., TRIZ, TQM)

    c. **Building on the Shoulders of Giants:** Not every piece of a system needs to be created from scratch. There is a lot of combining and recombining that can result in novel solutions.

        i. **Hacking:** You can make any formally specified system do all sorts of interesting and surprising things it wasn't specifically intended to do. Doing so can lead to novel solutions.

    d. **Idea Generation:** During the idea generation phase it is important to suspend judgment and be open to lots of possible ideas, even those which seem untenable at first glance. Those that seem untenable may end up elucidating assumptions or other things to consider.

    e. **Defined Goals/Metrics for Success:** To make decisions about ideas, goals need to be unambiguously articulated as they define what we are doing and what it means to get there (e.g., cost, features, etc.). For instance, there may be contradictory goals that suit different needs (e.g., time to market vs robustness). Often, there will be different impacts that suggest different, even contradictory (as in the example above), approaches to the problem.

        i. **Customers and Users**: Customers fund projects, while users interact with products. Customers can be users, but they are often distinct groups that can have goals and perspectives that are at odds with each other.

f. **Refining Solutions/ Tinkering:** Progress is often made by playing around and making "mistakes" with existing resources, ideas, and solutions. Rather than deducing the entire plan a priori, it is possible to slightly change the current plan and see the result.

g. **Design Review:** To improve from iteration to iteration, it is necessary to critique each solution and to point out flaws to develop improvements.

   i. **Focus on the Human Experience:** The success or failure of a product depends on how users interact with it and how the customer perceives the success. If something doesn't affect the user *or* the customer's success metrics, it should be very low in importance. Something that "almost works" from the developer's perspective is equivalent to not existing from the user's perspective.

3. **Iteration:** Progress is ultimately an iterative process with every solution creating a new component on which newer solutions can be built. Innovation is the process of finding unique combinations of or improvements to existing solutions.

   a. **Just start:** By doing *something* you are likely to give yourself more information which can lead to new solutions. Therefore, it is not necessary to completely define the problem before starting.

   b. **Trial and Error:** Technology & Engineering design is essentially a trial-and-error process that seeks an optimal solution defined by parameters such as time, cost, etc.

   c. **Iterate Early:** It is much easier and often much cheaper to perform a big iteration during early phases of design and development than in later phases (when smaller iterations are still being performed).

   d. **Reverse Engineering:** Sometimes the best way to build something is to look at an existing product and work backwards, breaking it down into its components to understand what's happening, and then rebuild or remix the item or similar items again.

4. **Tradeoffs:** There are decisions that need to be made and relationships between the decisions making some of them work at odds with each other. (e.g., software development: fast and cheap vs robust/high quality and expensive or hardware: speed vs size vs power-efficiency).

   - **Feedforward vs Feedback:** In many situations, "an ounce of prevention is worth a pound of cure." Feedback systems imply data to be corrected for and can induce ringing before stabilizing. Feedforward – correcting before the issue goes out of bounds – is frequently the better approach.

5. **Diminishing Returns:** The initial work that is put into a project creates a lot of value, but the more time passes, the less impactful the work becomes, relative to what has already been done. There is an appropriate time to stop making improvements, as they are no longer worth the time and money that they cost. A heuristic for determining this point is the 80/20 rule: the first 20% of work that is put in results in 80% of the total value of the final product, but this needs to be tempered by considerations about how unique and valuable the product becomes.

6. **Continuous Improvement:** Many products (though usually not vehicles or static structures) can be released before they are finished and can then incorporate iterative feedback from users or from the system.

7. **Fragility:** When designing and building a product, it is important to consider the ways that the product may behave under stress, how it can be misused or how it might simply fail.

    a. **Failure modes:** Understanding failure modes leads to deliberate design performance

        i. **Catastrophic vs. Graceful Failure:** When there is a clear plan of what to do when there are failures, it is possible to control the degree and type of failure.

        ii. **Early vs Late Failure:** The chances a given product breaks decreases sharply with time**.** In other words, if a product does not break within the first few weeks or months, it is likely not to break for a long time.

        iii. **Accidental vs Planned Failure (aka planned obsolescence):** Planned failure is a deliberate failure mode induced by the product designer. It is often in the best interest of a company to create a product that does not last forever, but rather requires the customer to return and buy more products with a relatively predictable cadence.

    b. **Robustness:** To create products which do not break under stress is an important goal of design. This can be built in different ways.

        i. **Safety Margin:** Some products and processes are designed with the goal of avoiding failure at all costs. Because products are used in the real world, they are built so that their calculated failure point is well above the level at which normal operation occurs.

        ii. **Redundancy:** Complex systems and structures tend to have a lot of redundant paths, so that the failure of any one component doesn't result in a failure of the entire system or structure.

    c. **Anti-Fragility:** Some products, often those modeled after organic systems, respond to stress by getting stronger rather than getting weaker (fragile) or being functionally unaffected (robust).

8. **Interfaces:** An interface defines a contract to which software or hardware developers agree, so that the underlying implementation doesn't need to match up for the pieces to work together and create a larger whole (e.g., application programming interfaces, electrical outlets).

9. **Standards & Protocols:** If enough engineers across enough organizations agree on a set of technical criteria, methods, processes, and practices, it is usually called a standard. Many standards are international, and often the race to create the standard determines which country or company will dominate a particular technology.