Abstract
A metaphor of mountain climbing is proposed to demonstrate how, as climbing footholds; artifacts capture, communicate, change and transform ideas. This metaphor is proposed to dispel common beliefs found regarding artifact use and is informed by academic research, a reflection and industry experiences.

Keywords
Artifacts, model-based design, natural programming

ACM Classification Keywords
D.2.10 [Software Engineering]: Design - Methodologies

General Terms
Design, documentation, languages

Introduction
The potential of artifacts seems hampered by the common belief that systems are created by clear-cut step-wise progressions from analysis to design to development to testing to release. In this model, each phase is supported with its own team member’s artifacts and at the end of a phase, artifacts are thrown over a wall to the next phase and everything flows smoothly. This does not happen.

My position regarding artifacts is that:
Artifacts are **footholds**, determined by the terrain, in the continuous climb to a finished system and operate by **capturing and communicating** ideas for the team as ideas **change** and artifacts **transform** into the final acceptable system.

The climbing metaphor implies a lack of distinction between phases and the need for all team members to use the same artifacts as they climb. In this metaphor, climbers are team members, footholds are artifacts, paths are ideas, and the mountain is the system. This is supported by my academic research [4], a reflection [3] on 3DUI design and development [5] and two years industrial experience on projects ranging from tens of thousands to tens of millions of dollars.

**Artifact Classification**

- **The path, not elevation, dictates the foothold type.**
Artifacts are classifiable into three general dimensions: informal/formal artifacts, design/development artifacts and artifacts of high/low fidelity. The following are some examples:

  - **informal design artifacts** – rough hand-drawn UI layouts, meeting notes, relationship sketches between actors and objects in systems
  - **formal design artifacts** – storyboards, wireframes, scenarios
  - **informal development artifacts** – scratch paper to side, informal timing, informal bug lists, notes to self about debugging or problem/hypothesis testing, To Do lists, notes to self, boxes and arrows
  - **formal development artifacts** – use cases, UML, requirements documents, bug trackers, change tracker

These dimensions are useful for selecting an artifact for a problem. Informal artifacts are often personal, incomplete, filled with domain specific terms, focused on a specific context and are used to create the formal artifacts. Formal artifacts are polished, complete, and made for others to see. Design artifacts are used for idea elicitation and consensus seeking. Development artifacts are used for comprehensive documentation, maintaining consensus, teaching, enabling an expert language of communication about the system and tracking for maintenance and feature reasons. Observations regarding these classifications include:

- People like formal artifacts more as they are more understandable and represent conclusions.
- Informal artifacts are valuable for problem-hypothesis testing common in reflective practices [3].
- People are oddly embarrassed and see little value in informal artifacts, despite the innate design rational.

Artifact fidelity dictates their use. High fidelity appears closer to the finish product, such as a mockup in the system’s final look and feel. They generally imply higher finalization of ideas and result in closer attention to low-level details [2]. Low fidelity focuses on only the details which practitioners are currently concerned, such using as wireframes for layout and early functionality testing. Low fidelity artifacts are commonly used early to “fail often and fail early”. In my experience:

- Higher-fidelity artifacts are more valued, especially by non-practitioners, even when they should not be.
- The purpose of low-fidelity artifacts is often confusing to non-practitioners and team members.
Uses and Purposes of Artifacts

Artifacts capture ideas and communicate them to others. As the ideas are refined, the ideas change. As artifacts are completed, their ideas are transformed into new artifacts, exploring the idea in new ways.

Capturing Ideas

- THE PATH DICTATES THE SPEED TRAVELED.
- FOOTHOLDS MUST MATCH THE TERRAIN TRAVELED.

Early on, important ideas assault the team in a massive rush but then slow to a trickle. Artifacts that capture the massive rush of information are about speed and abstraction of currently unimportant or cluttering ideas. These artifacts tend to be low-fidelity, informal and design-oriented such as meeting notes, sketches and wireframes. Later on, artifacts can focus on capturing refinements. Abstraction is still used but these artifacts can go into minute detail as these are within the practitioners’ knowledge limits. These artifacts tend to be higher fidelity and formality, such as prototyped systems, use cases, glossaries and flowcharts. It has been my experience that:

- Early ideas are often streamed, full of conflicting ideas, contain large information gaps and use domain specific terms loaded with meaning and unfamiliar to the practitioner.

Communicating Ideas

- FOOTHOLDS ARE USEFUL TO ALL CLIMBERS ON THE TEAM.
- GOOD FOOTHOLDS ARE WORTH THE EFFORT.

Artifacts are important for rapidly communicating ideas for team agreement. Good artifacts: avoids late-stage surprises, succinctly verify ideas, quickly kill bad ideas, find unknowns, elicit tacit knowledge, allow existing practice improvements, ramp-up new team members and propagate changes to team members. In my experience:

- Artifacts can be misinterpreted and cause confusion if the team is not properly instructing in their use.
- Not all artifacts are understood by team members.

Change

- EACH FOOTHOLD REQUIRES THE CLIMBER TO REAJUST.
- THE WRONG FOOTHOLDS LEAD CLIMBERS ASTRAY.

Maintaining artifacts in an up to date form is a challenge as a change can ripple through multiple artifacts and impact the team’s understanding in unforeseen ways. As team size grows, communication overhead skyrocket [1] and ideas can change rapidly.

- You can be assured that someone will find an out-of-date artifact and assume it is correct.
- You can be assured that not everyone will be up to date on all recent ideas (e.g. clients, analysts, users).
- New members need low-fidelity artifacts to catch up. These tend to be created early and rapidly go out dated. New members are the least likely to know this.
- Changed artifacts need to be reevaluated with all team members. This process clarifies the change and possibly changes it again.
- Explore ideas in small, informed groups and seek consensus with larger groups with the smaller group

---

1 I was once asked to design an interface without wireframes to save time because the wireframes were seen as a deliverable and not communication. It was disastrous.
attending. The smaller group will defend changes with their clout, experience and domain-specific rational.

- Many fear change and might block change to make project deadlines. Change requires more time in later stages and impacts many more artifacts.
- Common wisdom: “fail often and fail early”, avoid “too many cooks in the kitchen”, and use low-fidelity artifacts to represent “single points of failure.”

Transformation

- Footholds allow climbers to reach the next foothold at the furthest reaches of their ability.
- Sometimes good footholds lead to dead end paths and the climber must descend to find another path.

Artifact creation transforms the ideas of previous artifacts into new forms to explicitly address new issues. The transformation process is important in filling in information gaps. For example, static wireframes tease out controls and hinting at usage. Wireframes transformed into a storyboard explore work and users’ ability to complete tasks. Additionally, these transform into code, use cases and testing plans for other team members and team needs.

- Transformation is also an often overlooked part of the practitioner’s own growth in understanding.
- Transformation is very difficult and lacks tools.

Supporting Design and Development

The system design and development view I propose is: everything is an artifact, from meeting notes, right down to the final executable and the user manuals. Unfortunately, there is a lack of tools for managing ideas as they change and thread through artifacts. For this reason, my research has turned to multi-tiered model-based representations [6], called Concept-Oriented Design [4]. In this, a concept represents a single idea and transforms from common language, to causal representations, to automata, to code. As such, concepts manage idea capture, communication, change and transformation.

References