Motivation in Component-Based Software Development

G. Chroust
J. Kepler University Linz, Austria

INTRODUCTION

Information systems are designed for the people, by the people. The design of software systems with the help of software systems is another aspect of human-computer interfaces. New methods and their (non-)acceptance play an important role. Motivational factors of systems developers considerably influence the type and quality of the systems they develop (Arbaoui, Lonchamp & Montangero, 1999; Kumar & Bjoern-Andersen, 1990). To some extent, the quality of systems is a result of their developers' willingness to accept new and (supposedly) better technology (Jones, 1995). A typical example is component-based development methodology (Bachmann et al., 2000; Cheesman & Daniels, 2001). Despite considerable publication effort and public lip service, component-based software development (CBD) appears to be getting a slower start than anticipated and hoped for. One key reason stems from the psychological and motivational attitudes of software developers (Campell, 2001; Lynex & Layzell, 1997). We therefore analyze the attitudes that potentially hamper the adoption of the component-based software development approach. Maslow's Hierarchy of Need (Boeree, 1998; Maslow, 1943) is used for structuring the motives.

BACKGROUND

The Human Side of Software Engineering

Kunda and Brooks (1999) state that "software systems do not exist in isolation ... human, social and organizational considerations affect software processes and the introduction of software technology. The key to successful software development is still the individual software engineer" (Eason et al., 1974; Kraft, 1977; Weinberg, 1988). Different software engineers may account for a variance of productivity of up to 300% (Glass, 2001). On the other hand, any other single factor is not able to provide an improvement of more than 30%. The influence of an individual's motivation, ability, productivity, and creativity has the biggest influence by far on the quality of software development, irrespective of the level of technological or methodological support. Therefore, it is worthwhile investigating for what reasons many software engineers do not fullheartedly accept component-based methods (Lynex & Layzell, 1997).

Software development in general introduced a new type of engineers who show marked differences when compared to (classical) engineers (Badoo & Hall, 2001; Campell, 2001; Eason et al., 1974; Kraft, 1977; Kunda & Brooks, 1999; Lynex & Layzell, 1997). The phenomenon is not fully understood yet but seems to have to do with the peculiarities of software (Brooks, 1986), the type of processes and environments needed to develop software (Kraft, 1977), and especially to the proximity of software development to other mental processes (Balzert, 1996).

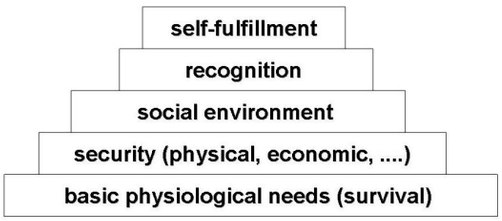
Maslow's Hierarchy of Needs

Maslow's theory (Boeree, 1998; Huitt, 2002; Maslow, 1943; McConnell, 2000) provides a practical classification of human needs by defining a five-level Hierarchy of Needs (Figure 1).

The five levels are as follows:

- **Basic Physiological Needs (Survival):** At this level, the individual is fighting for survival against an adverse environment, trying to avert hunger, thirst, cold, and inconvenient and detracting physical work environments.
- **Security (Physical, Economic ...):** On this level, the individual is concerned with the sta-

Figure 1. Maslow's hierachy of needs



bility of his or her future and the safety of the environment. Worries include job security, loss of knowledge, loss of income, health, and so forth.

- **Social Environment:** This category includes the need to have friends, belong to a group, and to give and receive love.
- **Recognition:** Individuals strive to receive appropriate recognition and appreciation at work and to be recognized as having a valuable opinion.
- **Self-Fulfillment:** This level is considered the highest stage attainable in the development of a person, drawing satisfaction from the realization of one's own contribution to a goal and one's fullfillment of their full potential as a human being.

Reuse and Component-Based Software Development (CBD)

An old dream in software development is to avoid unnecessary duplication of work by consistently and systematically reusing existing artifacts. Reuse promises higher productivity, shorter time-to-market, and higher quality (Allen, 2001; Cheesman & Daniels, 2001). Initially, ready-made pieces of software were made available; these delivered a defined functionality in the form of a black box (i.e., without divulging the internal structure to the buyer/user). They were called COTS (commercials off the shelf) (Voas, 1998). Later, an improved and more restricted concept was employed: software components (Bachmann et al., 2000; Cheesman & Daniels, 2001; Woodman et al., 2001). Software components have to fulfill additional requirements, restrictions, and conventions beyond the properties of COTS. To a user of a software component, only its interfaces and functionality are known, together with the assurance that the component obeys a specific component model. This component model defines how the component can be integrated with other components, the conventions about the calling procedure, and so forth. The internal structure, code, procedures, and so forth are not divulged—it is a black box.

Systematic, institutionalized CBD needs a change in the attitude of software engineers, different work organization, and a different organization of the whole enterprise (Allen, 2001).

Component-Based Development and Software Engineers' Needs

The acceptance of a new technology often meets with strong opposition caused by psychological motives, which can be traced to Maslow's Hierarchy of Needs.

Basic Physiological Needs

This level does not have any strong relevance; software engineering is a desk-bound, safe, non-endangering activity. We have to recognize, however, that very often software engineers have to struggle with adverse infrastructure (floor space, noise, etc.) (deMarco, 1985).

Security

The desire for security is threatened by numerous factors. The fears can be categorized into four groups:

Losing the Job or Position

- **Job Redundancy:** CBD promises considerably higher productivity and less total effort as a result of removing the redundancy of reimplementing already existing functions. This carries the thread of making an individual redundant, especially since the development of components very often is outsourced to some distant organization (e.g., India).
- **Implementing vs. Composing:** deRemer (1976) stressed the difference between implementing a module/component (programming in the small) and building (composing) a system out of components (programming in the large).