SEMAT, March 2010

Software, Engineering, Artefacts, Language

Observations

Software

Coding can be viewed as having to deal with someone else's representation (program notation or otherwise).

Software

Modelling can be viewed as working with a direct representation of the purpose that humans associate with a system.

Engineering?

Stuff

➡ **Terminology**

➡ Data structures

➡ **Information**

➡ **Knowledge**

➡ Solutions / products

➡ Monetised value

Engineering?

If software is information (models and code), then subject matter experts in various disciplines produce the vast majority of software, and software developers only produce a small fraction of software.

- Distinguishing between “us” (software developers) and “them” (software users) is counter-productive. We are all “computer users”, and all of us consume and produce information (software).

Language

The artefacts that subject matter experts produce when not shackled to a software engineering “methodology” tend to be neither specifications in a general purpose programming language nor do they tend to be long-winded stories expressed in natural language.

Artefacts

To date software producers have neglected the role of artefacts as natural units of work, and as a mechanism for defining the boundaries of areas of knowledge.

Proposed Definitions

Software

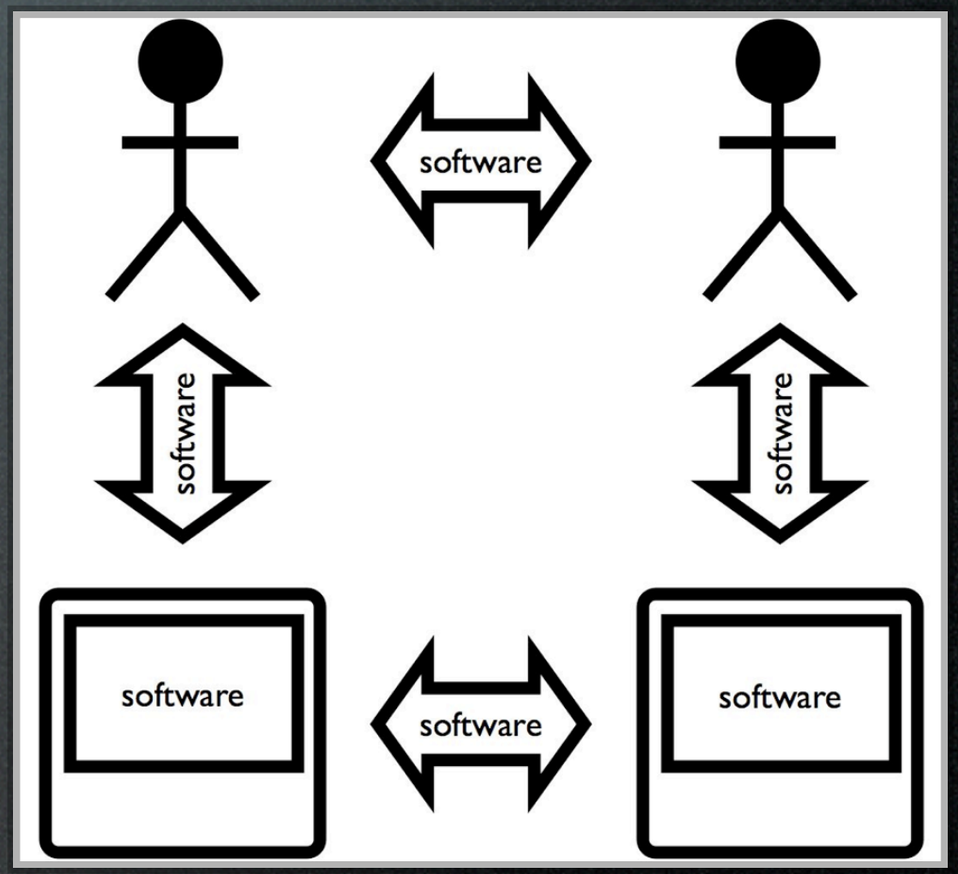
Coding happens when we work with third party implementation technologies (hardware or software) and when mapping to such technologies.

Software

Modelling happens when we capture knowledge in a domain specific notation that is grounded in established domain terminology.

Software

Software consists of all the models and code used within a computer and in the interactions between computers and humans



Engineering?

Engineering happens when we combine theories and empirically tested techniques to automate coding.

???

Artefacts

An **artefact** is a container of information that

- is created by a specific actor (human or a system)
- is consumed by at least one actor (human or system)
- represents a natural unit of work (for the creating and consuming actors)
- may contain links to other artefacts
- has a state and a lifecycle

Artefacts

A **software artefact** is an artefact that meets the following requirements:

- It is created with the help of a software program that enforces specific instantiation semantics
- The information contained in a software artefact can be easily processed by software programs
- Referential integrity between software artefacts is preserved at all times with the help of a software program
- No circular links between software artefacts are allowed at any time
- The lifecycle of a software artefact is described in a state machine
- The events consumed and produced by the artefact state machine are available for processing in software programs

Language

Software artefact design is the emerging discipline of recording useful domain specific jargon and nudging the jargon into a shape where ambiguities are resolved, and where the artefacts articulated in the jargon meet the proposed definition of a software artefact.

Proposed Goals

Software

Software production techniques should promote modelling, and should aim to minimise the amount of code that humans are directly exposed to.

Engineering?

All software changes should be as low-risk as a database transaction.

- Program specification changes are simply very-long-running transactions, and their duration is artificially inflated by the arcane mechanisms that we currently use to perform such transactions.

Artefacts

Fundamental practices and patterns for software production must directly relate to the production of software artefacts.

Language

Software producers need to agree on

- a practical notation for decorating any software artefact with instantiation semantics, such that the result is a template for software artefacts produced by down-stream roles in the value chain.

- a software program that acts as the reference implementation for software artefact instantiation semantics.

Thank you

Jorn Bettin
jbe @ sofismo . ch