Extreme Programming (XP): Die Metapher

Seminar "Ausgewählte Beiträge zum Software Engineering"
WS 2004/2005

Vorbemerkung zu XP (1)

- XP definiert (nach Kent Beck) 12 Praktiken:
  - Das Planungsspiel
    - Iterative Planung, Storycards, Beteiligte: Entwicklung und Kunde
  - Kurze Releasezyklen
    - Kleine Releases erstellen
  - Einfaches Design
    - Keine Redundanzen, geringste Anzahl von Klassen und Methoden,...
  - Testen
    - Komponententests, Funktionstests, Automatisierung,...
  - Refactoring
    - Vereinfachung
  - Programmierung in Paaren
    - Aller Code wird von zwei Personen zusammen geschrieben

Vorbemerkung zu XP (2)

- Gemeinsame Verantwortlichkeit
  - Jeder darf an jeder Stelle des Codes arbeiten
- Fortlaufende Integration
  - Code wird nach wenigen Stunden integriert (Integrationsrechner) und getestet
- 40-Stunden-Woche
  - Jeden Tag frisch und tatkräftig beginnen
- Kunde vor Ort
  - Zusammenarbeit mit dem Team/Verfügbarkeit, Fragen beantworten…
- Programmierstandards
  - Keine unterschiedlichen Programmierstile, Code von allen sieht gleich aus
- Metapher

Vorbemerkung zu XP (3)

- XP stützt sich (nach Kent Beck) auf folgende vier Werte:
  - Kommunikation
  - Einfachheit
  - Feedback
  - Eigenverantwortung

Was will dieser Vortrag? (1)

- Klärung/Verdeutlichung des Begriffs (System-)Metapher aus dem Extreme Programming (XP)
  - Verstehen was eine Metapher ausmacht und wie sie "arbeiten" kann.
  - Schön wäre auch: Wie verbessert die Metapher die Kommunikation?
- "Ideen" für den Einsatz in Softwareprojekten liefern
  - Dazu Vorstellung eines Artikels aus dem Jahr 2004:

    Rilla Khaled, Pippin Barr, James Noble, (Victoria University of Wellington/New Zealand) and Robert Biddle (Carleton University/Canada)

    System Metaphor in "Extreme Programming": A Semiotic Approach
    - Zentrales Thema: Beleuchtung der Struktur von Systemmetaphern durch das Aufstellen eines Modells.
    - Dazu wird viel Arbeit in die Bereitstellung von Begrifflichkeiten gesteckt!

Was will dieser Vortrag? (2)

- Der Weg ist das Ziel!
  - Struktur der Systemmetapher begreifen!
- ...aber nicht das einzige!
  - Konkrete Beschreibung zur Auswahl und Bewertung von Systemmetaphern.
- Die Evaluierung des eigentlichen Nutzen bleibt aber (leider) erst einmal weitgehend außen vor!

Aufbau

- Einführung in das betrachtete Metapher-Umfeld
- Allgemeine Begriffsklärung Metapher/Systemmetapher
- Erkennen der damit zusammenhängenden Probleme
- Vorstellung Modells für XP Systemmetaphern
  - Semiotik und Zeichen nach Peirce
  - Metaphern nach Lakoff und Johnson
  - Metaphern als Zeichen
  - Objektorientierte Programmierung und Zeichen
  - Ein auf Zeichen beruhendes Modell für Systemmetaphern
    - Kunden, Programmierer und Manager benutzen eine gemeinsame Sprache und können somit besser zusammen arbeiten (Kommunikation zw. Levels)
    - Kunden erhalten eine "Sprache", um ihre Domain zu modellieren (Kommunikation auf einem höheren Level)
    - Entwickler erhalten eine "Sprache", um ihre Zusammenarbeit beim Bau der Software zu verbesseren und mit der Software selber zu arbeiten (Kommunikation auf einem niedrigeren, technologienahen Level)

Die Materie oder "Worum geht‘s?" (1)

- Allgemeine Definition des Begriffs **Metapher** (nach Wikipedia):
  - *Metapher (v. griech.: meta pherein = anderswo hintragen) ist eine rhetorische Figur, eine Verdichtung, die der Verdeutlichung und Veranschaulichung dient. In dieser Art des Tropus erfolgt der Ersatz der Bedeutung eines Ausdrucks durch einen versinnbildlichten Ersatzausdruck.*
  - *Bei der Metapher werden zwei getrennte Sinnbereiche in einen ungewohnten, oft kreativen Zusammenhang gerückt.*
  - *Metaphern sind zweideutig. Wenn man sie "wörtlich" (beziehungsweise die Wörter in ihrem ursprünglichen, gewohnten Sprachgebrauch) nimmt, sind sie sozusagen falsch.*
- Verwendung eines oder mehrerer Wörter nicht in eigentlicher, sondern übertragener Bedeutung, das eigentliche Wort wird durch ein anderes aus einem anderen Vorstellungsbereich ersetzt, wobei eine sachliche oder gedankliche Ähnlichkeit bzw. Bildstruktur gemeinsam ist.

Die Materie oder "Worum geht‘s?" (2)

- Lakoff/Johnson (*Metaphor we live by, 1980*):
  - "Die Metapher ist für die meisten Menschen ein Mittel der poetischen Imagination."
  - "Überdies ist es typisch, dass die Metapher für ein rein sprachliches Phänomen gehalten wird."
  - "Wir haben dagegen festgestellt, dass die Metapher unser Alltagsleben durchdringt, und zwar nicht nur unsere Sprache, sondern auch unser Denken und Handeln."
  - "Unser alltägliches Konzeptsystem, nach dem wir sowohl denken als auch handeln, ist im Kern und grundsätzlich metaphorisch."
  - "Unsere Konzepte strukturieren das, was wir wahrnehmen, wie wir uns in der Welt bewegen und wie wir uns auf andere Menschen beziehen. Folglich spielt unser Konzeptsystem bei der Definition unserer Alltagsrealitäten eine zentrale Rolle.

Die Materie oder "Worum geht‘s?" (3)

- Z.B.: Die Metapher "Argumentieren ist Krieg"
  - Sie findet sich in der Alltagssprache in vielen Ausdrücken wieder:
    - "Ihre Behauptungen sind *unhaltbar.*"
    - "Er *griff jeden Schwachpunkt* in meiner Argumentation *an*."
    - Seine Kritik *traf ins Schwarze*.
    - "Sie sind anderer Meinung? Nun, *schießen Sie los*"
  - Allerdings ist es so, das wir über das Argumentieren nicht nur in Kriegsbegriffen sprechen:
    - Wir können beim Argumentieren auch *gewinnen* und *verlieren*.
    - Die Argumentationsstruktur spiegelt ein Kampfgeschehen wieder: Angriff, Verteidigung, Gegenangriff…
  - Die Metapher strukturiert Handlungen, die wir beim Argumentieren ausführen.
  - Achtung: Metaphern sind kulturabhängig!

Die Materie oder "Worum geht‘s?" (4)

- Beispiele für Metaphern (www.literaturnetz.com): Übertragung…
  - …von einem Lebewesen auf ein anderes
    - "Rabenvater"
    - "Du bist ein Esel"
  - …vom Leblosen auf Lebloses
    - "Flussbett"
  - …vom Belebten auf Lebloses und umgekehrt
    - "Holzkopf"
  - …vom Konkreten auf Geistiges und umgekehrt
    - "glühende Liebe"
- Lakoff/Johnson:
  - **Strukturmetaphern**: „[…] Fälle, in denen ein Konzept von einem anderen her metaphorisch strukturiert wird."
  - Es gibt noch andere Klassen von Metaphern: z.B. **Orientierungsmetaphern**
    - Tugend ist oben/Laster ist unten: "Sie setzt *hohe* Standards", "Sie hat einen *aufrechten* Charakter", …

Die Materie oder "Worum geht‘s?" (5)

- Definition **Systemmetapher** nach Kent Beck:
  - *Eine Geschichte, mit der jeder – Kunde, Programmierer, Manager – die Funktionsweise des Systems veranschaulichen kann.*
  - *Die Metapher soll es allen Beteiligten lediglich erleichtern, die grundlegenden Bestandteile und deren Beziehungen zu verstehen.*
  - *Die nähere Beschäftigung mit der Metapher kann das gesamte Team zu neuen Ideen anregen.*
- Definition **System Metaphor** im Wiki auf http://c2.com:
  - *What Extreme Programming (XP) uses instead of a formal architecture. A simple shared story of how the system works, a metaphor. This story typically involves a handful of classes and patterns that shape the core flow of the system being built.*

Die Materie oder "Worum geht‘s?" (6)

- Der Begriff Systemmetapher bleibt im folgenden Sinn abstrakt:
  - Wie "findet" man eine geeignete (System-)Metapher?
  - Wie geht man mit ihr um?
  - Was ist der genaue "Wert" einer solchen Metapher?
    - *Ist sie wirklich hilfreich?*
    - *Wie verhindert man, dass sich eine gewählte Metapher später als falsch herausstellt?*
      - *Metapher kann zu schwach sein: Sie liefert keine Einsichten in potentielle Architekturen.*
      - *Metapher kann zu stark sein: Systemkomponenten werden in eine Form gepresst, in die sie logisch nicht passen.*
  - ⇨ Die Entwicklung einer Struktur soll helfen, die Fragen beantworten zu können.

Die Materie oder "Worum geht‘s?" (7)

- Die Unklarheiten bzgl. dieser Fragen hat auch Kent Beck bemerkt:
  - Auf der OOPSLA 2002 hält er einen Vortrag zum Thema *The Metaphor Metaphor.* Aus der Ankündigung:
    - […] the suggestion that customers and developers share a common metaphor or methaphors for the system is the most problematic.
    - […] there are two possibilities: I have done a poor job of explaining methaphors and their importance, or I‘m flat wrong and metaphors just aren‘t that important.
    - […] I will try just once more to explain the positive role of consciously metaphorical thinking […] if it works, if people get it, great. If not, I will swear off trying to explain metaphors forever.
  - Leider ist der Inhalt des Vortrags nicht zugänglich.

Die Materie oder "Worum geht‘s?" (8)

- Martin Fowler (*Extreme Programming Examined*):

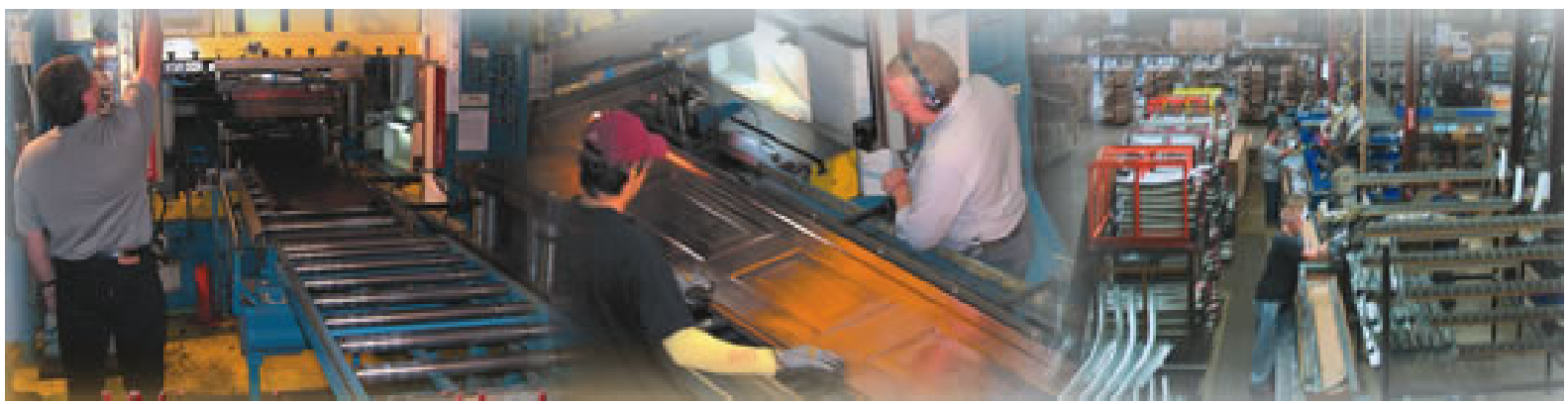
  […] I still haven‘t got the hang of this metaphor thing. I saw it work, and work well, on the C3 project, but it doesn‘t mean I have an idea how to do it, let alone how to explain how to do it.

- Ken Auer/Roy Miller (*Extreme Programming Applied*):

  - *A lot of people doing XP say they haven‘t really found a good metaphor or that they use metaphor only for certain parts of the system. All of the people we‘ve talked to who don‘t use metaphor haven‘t seen it as a significant problem.*

Beispiel: Metapher im Chrysler C3 Payroll System

- 1996 C3 Projekt (Chrysler Comprehensive Compensation)
  - Kent Beck und Ron Jeffries arbeiten als Coaches (ab 1996)
  - System zur Gehaltsabrechnung
  - Launch: 1997
  - Metapher: *Payroll System is an assembly line*
    - Extensiver Gebrauch von Konzepten aus der Fertigung
      - Leitungen/Bänder (lines)
      - Stationen (stations)
      - Behälter (bins)
      - Teile, Werkstücke (parts)



[KhaBarBob04]: System Metaphor in "Extreme Programming": A Semiotic Approach

- Motivation:
  - Die Systemmetapher wird wenig verstanden und oft ignoriert.
  - Es gibt nur sehr wenig Literatur, die sich mit den folgenden Fragen beschäftigt:
    - Wie wählt man eine Metapher?
    - Wie verwendet man eine Metapher?
  - Es ex. kaum Forschungen in diesem Bereich.
- Ziel:
  - Bereitstellung eines Modells für Systemmetaphern, welches eine Beschreibung liefert, wie diese etwas zu einem Softwaresystem beitragen können.
    - Verstehen wie eine Systemmetapher arbeiten kann.
    - Sehen wie die Systemmetapher die Kommunikation verbessern kann.
    - Anleitung zur Auswahl einer brauchbaren Metapher liefern.
      - Finden potentieller Metaphern
      - Kriterien zur Beurteilung von gefundenen Metaphern
- Methodik:
  - Verwendung/Beschreibung von existierenden Techniken von C.S.Peirce bzw. Lakoff und Johnson zur Entwicklung eines formalen, semiotischen Modells einer Metapher in XP.
    - Analyse der Struktur von Systemmetaphern unter Verwendung von *Peircean Semiotics*

[KhaBarBob04]: Zusammenfassung

- Bereitstellung eines Modells für die XP-Praktik Systemmetapher.
- Das Modell basiert auf Semiotik.
- Das Modell stellt Richtlinien für XP-Entwickler zur Entwicklung und Evaluierung von Metaphern zur Verfügung.
- Die Entwickler müssen hierzu nicht direkt Semiotik verwenden.

[KhaBarBob04]: XP und Metapher

- Noch einmal Definition:
  - Die Praktik der **Systemmetapher** ist ein Weg, die logische Architektur eines Systems in einer Sprache zu beschreiben, die sowohl den Entwicklern als auch den Kunden bereits vertraut ist.
  - Sie stellt ein zwischen Kunden und Entwicklern "geteiltes" Vokabular für die Diskussion von Problemen und Lösungen zur Verfügung.
  - Im Sinne von XP (Verhinderung von Investitionen in das "Unbekannte") ist die Systemmetapher ein preiswertes Systemdesign.
    - Hauptsystemkomponenten und ihre Interaktionen
  - Für Entwickler:
    - Die Systemmetapher unterstützt darüber hinaus Konsistenz in der Benamung von Programmelementen:
      - Subsysteme
      - Pakete
      - Klassen
      - Methoden

[KhaBarBob04]: XP und Metapher

- Kent Beck (*Extreme Programming Explained*):
  - *The metaphor just helps everyone on the project understand the basic elements and their relationships.*
  - *Words chosen to identify technical entities should be consistently taken from the chosen metaphor.*
  - *As development proceeds and the metaphor matures, the whole team will find new inspiration from examining the metaphor.*

Semiotik (1)

- Wikipedia:
  - *Semiotik* ist die allgemeine Lehre von den Zeichen, Zeichensystemen und Zeichenprozessen.
  - Ein *semiotischer Vorgang* liegt vor, wenn eine codierte Nachricht von einem Sender zu einem Empfänger gesendet wird, diese Nachricht vom Empfänger decodiert – also verstanden – werden kann und zu einer Information wird.
  - *Charles Sanders Peirce (*Mathematiker, Philosoph und Logiker, 1839–1914) gilt als einer der Begründer der "modernen" Semiotik:
    - Peirce geht von einem dreiteiligen System aus, welches er *Semiosis* nennt.

Semiotik (2)

- Hadumod Bußmann (*Lexikon der Sprachwissenschaft, 1990*)
  - *Semiotik* (gr. *semeion* 'Zeichen'. – Auch: Sematologie, Semiologie, Zeichentheorie). Theorie und Lehre von sprachlichen und nichtsprachlichen Zeichen und Zeichenprozessen, in deren Zentrum die Erforschung natürlicher Sprache als umfassendstem Zeichensystem steht.
  - Außer Sprach- und Kommunikationstheorie aber beschäftigen sich vielfältige geisteswissenschaftliche Disziplinen mit Theorien nichtsprachlicher Zeichen (Ästhetik, Graphik, Kunstwissenschaft, Mythenforschung, Psychoanalyse, Kulturanthropologie, Religionswissenschaft u.a.).
  - Mit Ch. W. Morris sind folgende Untersuchungsaspekte zu unterscheiden:
    - Syntaktischer Aspekt: Relation zwischen verschiedenen Zeichen (Syntax);
    - Semantischer Aspekt: Relation zwischen Zeichen und Bedeutung (Semantik);
    - Pragmatischer Aspekt: Relation zwischen Zeichen und Zeichenbenutzer sowie
    - Sigmatischer Aspekt: Relation zwischen Zeichen und Realität

Semiotik (3)

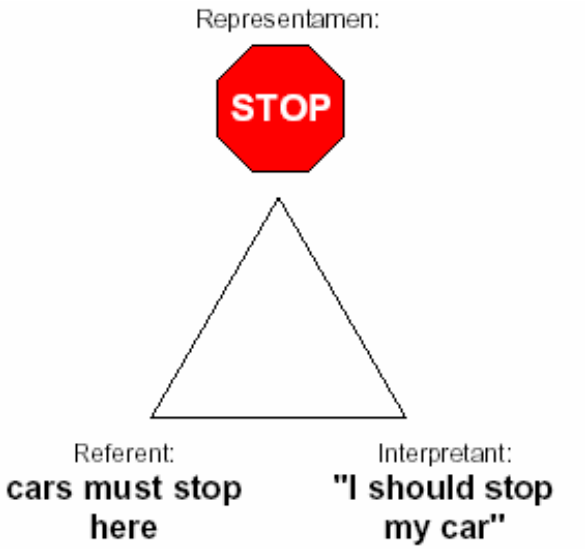
Semiotik ist die formale Untersuchung von *Zeichen*

[KhaBarBob04]: Definition Zeichen

- Modell von Zeichen nach Peirce:
  - *something which stands to somebody for something in some respect or capacity*
  - Zeichen als Tripel (*A Syllabus of Certain Topics of Logic*, 1903):
    - *A Sign, or **Representamen**, is a First which stands in such a genuine triadic relation to a Second, called its **Object**, as to be capable of determining a Third, called its **Interpretant**, to assume the same triadic relation to its Object in which it stands itself to the same Object.*
    - ***Representamen***: Aktuelle Verkörperung des Zeichens (das Bezeichnende), z.B. ein geschriebenes Wort.
    - ***Object***: Das Bezeichnete.
      - Im Rahmen von [KhaBarBob04] wird Object ***Referent*** genannt.
      - "[…] the referent represents that concept"
    - ***Interpretant***: Das Interpretierte: Ein Zeichen erzeugt etwas "im Kopf" des Interpretierenden.
      - Mit *Interpretant* ist somit der Effekt gemeint, den das Zeichen im Bewußtsein des Benutzers ausübt, nicht der Benutzer des Zeichens. Der Effekt hängt dabei sowohl vom Zeichen als auch von der Erfahrung des Benutzers mit dem Zeichen einerseits und mit dem Objekt, für das das Zeichen steht andererseits ab (John Fiske, *Introduction to Communication Studies*, 1982).

[KhaBarBob04]: Beispiel für Zeichen

- Anwendung von Peirceans Tripel auf ein Stop-Zeichen:



  - *Represantamen*: Rotes Achteck mit dem Wort "STOP" in weißen Buchstaben
  - *Referent*: Fahrzeuge müssen hier halten
  - *Interpretant*: "Ich sollte mein Auto anhalten"
    - Allerdings gibt es keine Garantie, dass diese richtige Interpretierung hervorgerufen wird

[KhaBarBob04]: Semiotik und Metapher (1): Noch eine Definition

- Nach Lakoff und Johnson (*Metaphor we live by*, 1980):
  - Das Wesen der Metapher besteht darin, dass wir durch sie eine Sache oder einen Vorgang in Begriffen einer anderen Sache bzw. eines anderen Vorgangs verstehen und erfahren.

[KhaBarBob04]: Semiotik und Metapher (2): Strukturierung der Metapher

- Lakoff und Johnson unterscheiden zw. *Tenor* und *Vehicle* einer Metapher:
  - Der ***Tenor*** (Zielbereich) ist das Subjekt, dem Attribute zugeschrieben werden.
  - Das ***Vehicle*** (Quellbereich) ist das Subjekt, von welchem die Attribute abgeleitet wurden.
  - Z.B.

    "All the world's a stage,
    And all the men and women merely players
    They have their exits and their entrances"

    (William Shakespeare, *As you like it*, 1599)

    - Die Welt soll beschrieben werden, indem bekannte Theatereigenschaften verwendet werden.
    - Die Welt ist hier der *Tenor* und die Bühne das *Vehicle*.
    - "Männer und Frauen" bilden einen ungeordneten *Tenor*, wobei "Schauspieler" das zugehörige *Vehicle* ist.
    - In der dritten Zeile werden dann Eigenschaften des *Vehicles* gewählt, um sie auf den Tenor zu übertragen
  - Z.B. *Juliet is the sun (*William Shakespeare, *Romeo and Juliet*, 1595)
    - *Juliet* ist der *Tenor*
    - *Sun* ist das *Vehicle*

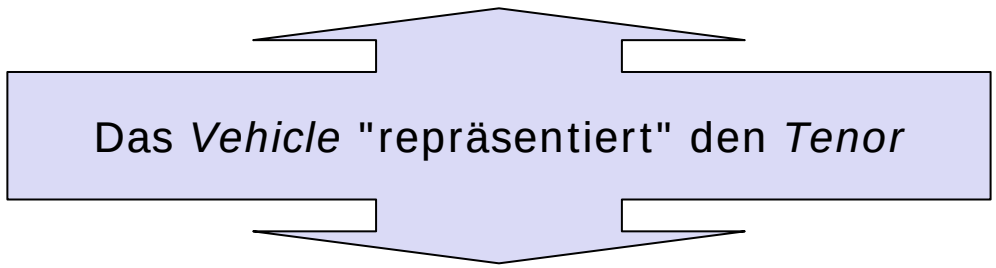
[KhaBarBob04]: Semiotik und Metapher (3): Übertragung von Eigenschaften bei Metaphern

- Lakoff und Johnson definieren auch den Begriff ***Metaphorical Entailment*** (semantische Implikation/metaphorische Ableitung):
  - Anwendung einer Eigenschaft/Tatsache des *Vehicle* einer Metapher auf den *Tenor* einer Metapher
  - Z.B. für *Juliet is the sun:*
    - Wenn man davon ausgeht, dass Romeo diese Aussage macht, ist ein mögliches *Metaphorical Entailment*:

      Romeo's world revolves around Juliet
    - Grund: Die Tatsache, dass die Elemente des Sonnensystems um die Sonne kreisen wird auf Juliet übertragen.
  - Während die Metapher einen direkten Vergleich zwischen *Tenor* und *Vehicle* macht, bestehen *Metaphorical Entailments* aus allen indirekten resultierenden Eigenschaften, die auf Basis des *Vehicles* über den *Tenor* abgeleitet werden können.

[KhaBarBob04]: Semiotik und Metapher (4): *Metaphor Introduction* modelliert als Zeichen

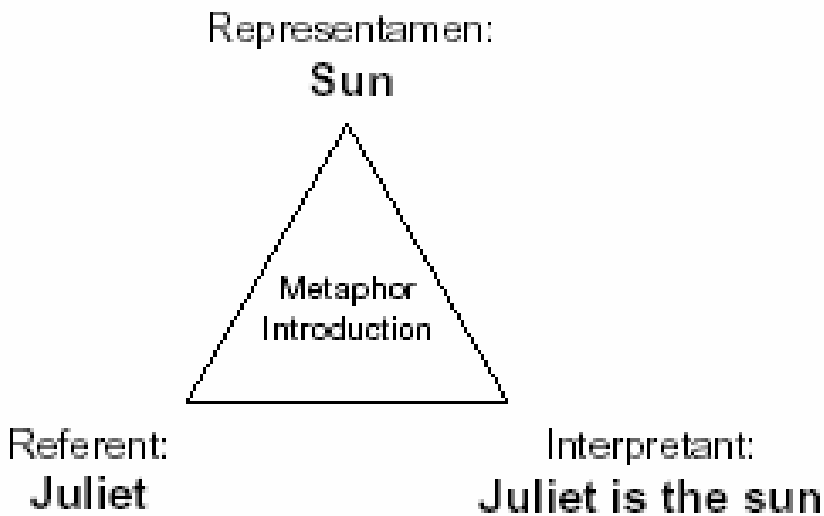
- Verwendung von Zeichen nach Peircean zur Modellierung (der Teile) einer Metapher
  - ***Metaphor Introduction Sign***: Einführung/Definition einer Metapher und Setzen ihres *Tenors* und *Vehicles* in den Kontext von Zeichen
    - Der *Represantamen* des Zeichens besteht aus dem *Vehicle* der Metapher.



    - Der *Referent* des Zeichens ist der *Tenor* der Metapher.
    - Der *Interpretant* des Zeichens ist die vollständige Metapher.
      - Da es eine "Interpretation" ist, zu welcher der Betrachter kommen kann, wenn er auf der *Represantamen* im Kontext des *Referent* trifft.

[KhaBarBob04]: Semiotik und Metapher (5): Beispiel für semiotisches Modell einer Metapher

- Beispiel eines semiotischen Modells der *Metaphor Introduction* (Metapher: *Juliet is the sun*):

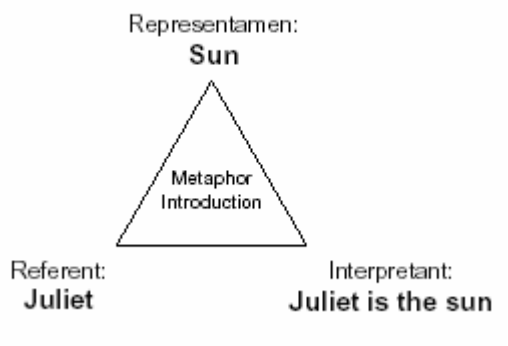


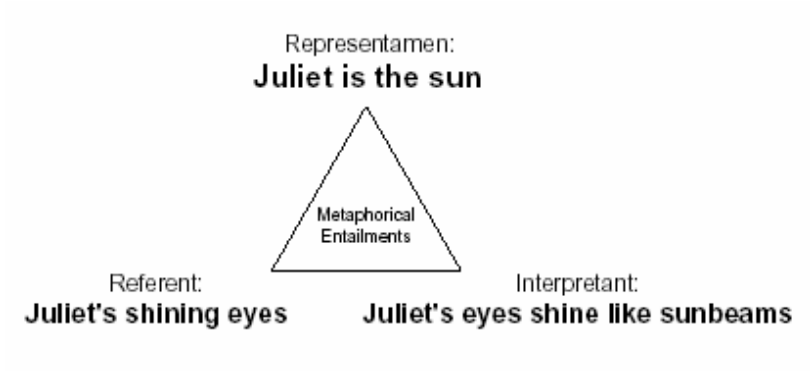
[KhaBarBob04]: Semiotik und Metapher (6): *Metaphorical Entailments* modelliert als Zeichen

- Die *Metaphorical Entailments* können als Exemplar eines weiteren Zeichens modelliert werden, dem ***Metaphorical Entailment Sign***
  - Der *Represantamen* (des *Metaphorical Entailment Signs*) ist der *Interpretan*t des *Metaphorical Intoduction Signs* (also die vollständige Metapher).
  - Der *Referent* (des *Metaphorical Entailment Signs*) ist das, wofür der *Represantamen* steht, typischerweise ein charakteristisches, unterscheidendes Merkmal des *Tenors* der Metapher.
  - Der *Interpretant* (des *Metaphorical Entailment Sign*) ist dann ein Ergebnis des Nachdenkens über die Metapher.

[KhaBarBob04]: Semiotik und Metapher (7)

- Beispiel eines semiotischen Modells des *Metaphorical Entailment* (Metapher: *Juliet is the sun*):





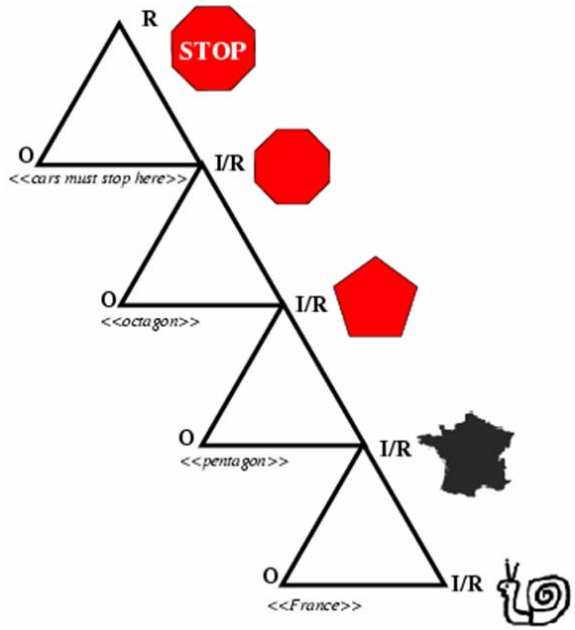
  - Hier ist *Interpretant* die Bedeutung, die Romeo der Metapher zuschreibt.
    - Diese ist typischerweise eng an die Eigenschaft gebunden, die durch *Referent* verkörpert wird.

[KhaBarBob04]: Semiotik und Metapher (8)

- Unterschied zwischen einer Eigenschaft und einem *Entailment*:
  - Die Eigenschaft beschreibt den *Tenor* unabhängig vom *Vehicle*
  - Das *Entailment* bezieht explizit Merkmale des *Vehicles* auf den *Tenor*.
- Unterschied zwischen den *Interpretants* der beiden Zeichen:
  - Der *Interpretant* des *Metaphorical Entailment Signs* ist konnotativ (*connotative*), dies heißt, dass seine Bedeutung von persönlichen und kulturellen Assoziationen abhängig ist.

[KhaBarBob04]: Semiotik und Metapher (9): Semiose

- Semiose (semiosis)
  - Bei Peirce impliziert das triadische Verhältnis der Zeichenglieder einen Prozess, den sog. Zeichenprozess, die Semiose.
    - Zeichen aus Zeichen erzeugen: Der *Interpretant* eines Zeichens kann zum *Represantamen* eines anderen Zeichens werden.
    - Der semiotische Interpretationsprozeß ist im Prinzip *unendlich* (Peirce: *unlimided semiosis*). Z.B.:



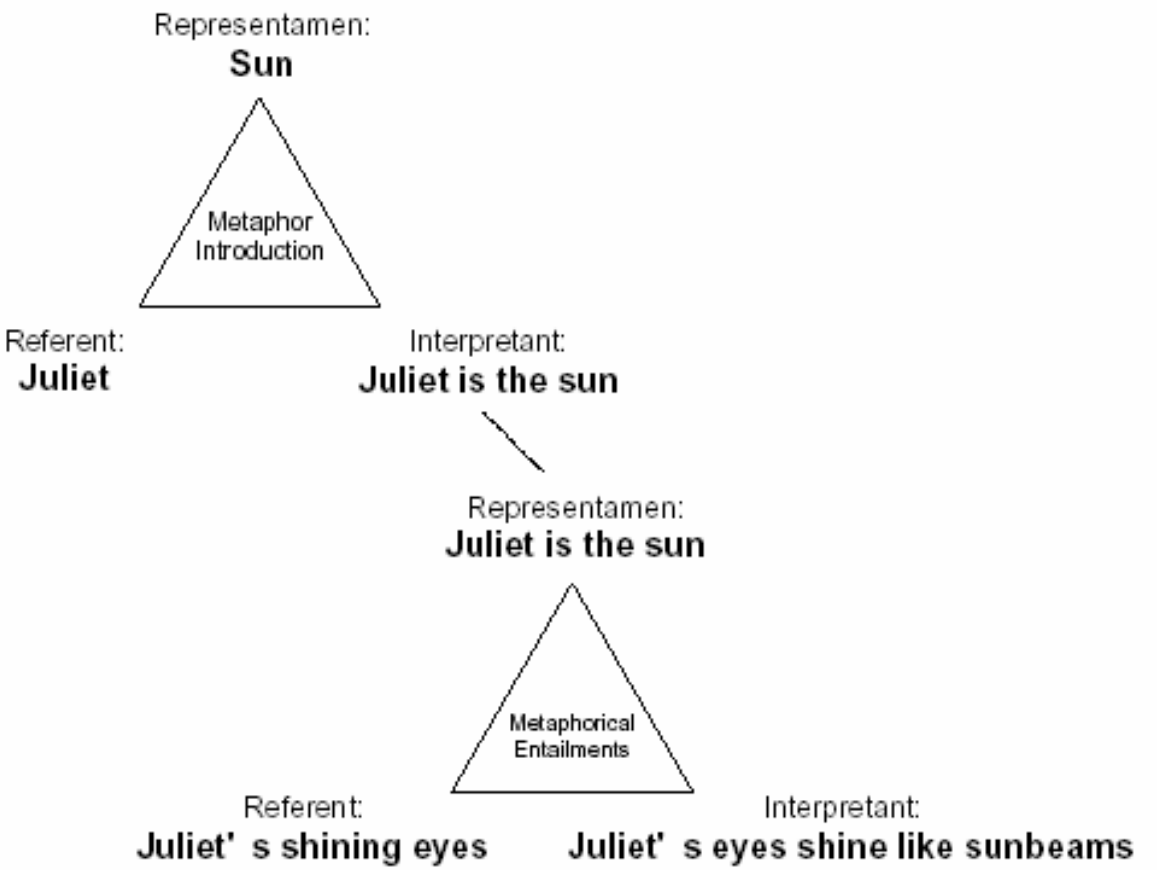
[KhaBarBob04]: Semiotik und Metapher (9): Semiose

- Mit der Betrachtung von Peirce hat ein Zeichen eine *generative Fähigkeit,* die im Fall der Metaphern auf besonders ausgeprägte Weise funktioniert:
  - Ein *Metaphor Introduction Sign* kann unmittelbar zu vielen Instanzen des *Metaphorical Entailment Signs* führen
    - » Z.B. existieren in Romeos Welt viele *Referents* bzgl. Juliet: ihre Haut, ihre Augen, ihr Haar, etc.
    - » Diese können alle zu Referents für *Metaphorical Entailment Signs* werden

⇨ Ein "gutes" *Metaphor Introduction Sign* wird eine schnell arbeitende Fabrik zur Erzeugung von *Metaphorical Entailment Signs*

[KhaBarBob04]: Semiotik und Metapher (10): Beispiel für Semiose

- Ein semiotisches Modell einer Metapher mit einem einzelnen *Entailment:*



[KhaBarBob04]: Semiotik und Metapher (10): Zusammenfassung

- Eine Metapher "funktioniert" durch die Beschreibung ihres *Tenors* (Zielbereiches) in Begrifflichkeiten ihres *Vehicles* (Quellbereichs).
- Die Metapher sowie ihre *Entailments* (semantische Implikationen) können als Zeichen beschrieben werden.
  - *Metaphor Introduction Sign*
  - *Metaphorical Entailment Sign*
- Diese beiden Zeichen sind durch eine Kette verbunden (Semiose)
  - Der *Interpretant* des *Metaphor Introduction Signs* wird zum *Represantamen* des *Metaphorical Entailment Signs.*
- Dieser Bindungsprozess tritt für eine Menge von *Referents* wiederholt auf. Dies erlaubt es, dass Charakteristiken des ursprünglichen *Referent* auf eine neue Weise interpretiert/gesehen werden.

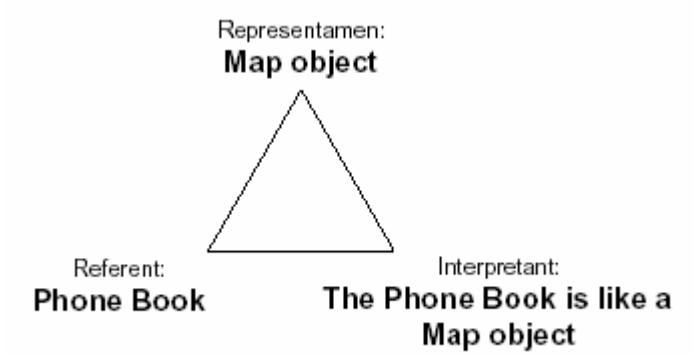
[KhaBarBob04]: Objektorientierte Programmierung und Zeichen (1)

- Semiotik kann dazu verwendet werden, objektorientierte Programmierung zu beschreiben.
- Hintergrund (1993, Madsen, Møller-Petersen, Nygaard: Object-Oriented Programming in the BETA Programming Language):

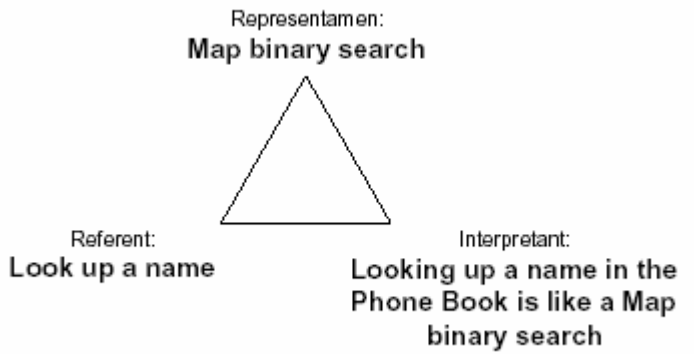
  A program execution is regarded as a physical model, simulating the behaviour of either a real or imaginary part of the world.
- Beispiel: Betrachten wir ein Telefonbuchobjekt, bei dem eine Map zur Implementierung des Telefonverzeichnisses verwendet wird:
  - Die Map repräsentiert das Telefonbuch, während
  - eine binäre Suche auf der Map das Nachsehen im Telefonbuch repräsentiert.

[KhaBarBob04]: Objektorientierte Programmierung und Zeichen (2)

- Dies führt zu "*Object-Oriented Programming" Signs (OOP Sign)*:
  - Der *Represantamen* ist ein Programmelement, welches eine Repräsentation bildet
    - Z.B.: Die Map oder die Suchmethode
  - Der *Referent* ist das zu repräsentierende Konzept aus der Domain
    - Z.B.: Das Telefonbuch oder das Nachsehen
  - Der *Interpretant* stellt die begriffliche Beziehung zw. dem Programmobjekt und dem externen Objekt her
    - Z.B.: Eine spezielle Map repräsentiert ein spezielles Telefonbuch



OOP Sign für Telefonbuchobjekt



OOP Sign für Suchmethode

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (1)

1. Die XP Systemmetapher ist zuerst einmal eine Metapher.
   - Beschreibung von etwas in der Terminologie von etwas anderem.
2. Die XP Systemmetapher ist aber auch ein Spezialfall der Metapher, da sie ein objektorientiertes Softwaresystem beschreibt.

⇨ Modellierung der Systemmetapher durch Kombination des semiotischen Modells einer allgemeinen Metapher mit dem semiotischen Modell von objektorientierten Systemen.

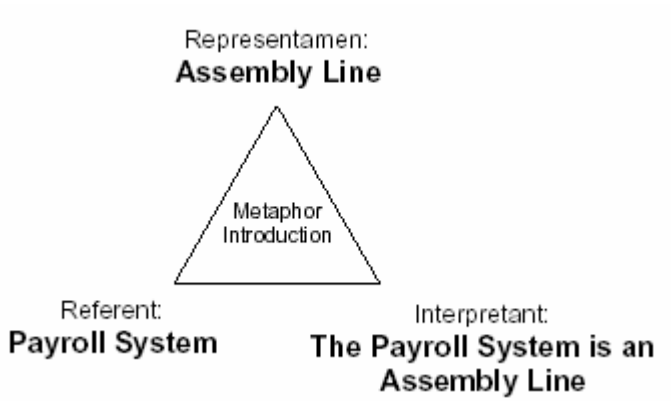
Wie?

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (2): Die Bestandteile im Überblick

- Das Modell der XP Systemmetapher besteht aus drei zusammenhängenden Zeichen:
  1. Das erste Zeichen führt die Metapher bzw. deren Teile ein (*Metaphor Introduction Sign*).
  2. Der zweite Zeichentyp "arbeitet" mit den *Entailments* der Metapher (*Metaphorical Entailments Sign*).
  3. Das dritte Zeichen repräsentiert Programmkonstrukte, die sich aus den *Metaphorical Entailments* ergeben.

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (3): *Metaphor Introduction Sign*

- *Metaphor Introduction Sign*
  - Der *Representamen* ist das Vehicle der Systemmetapher als Ganzes
    - Im C3-Projekt: *Assembly Line*
  - Der *Referent* bezieht sich auf die Domain für die das System gebaut werden soll
    - Im C3-Projekt: *Payroll System*
  - Der *Interpretant* ist die vollständige Metapher
    - Im C3-Projekt: *Payroll System is an assembly line*



[KhaBarBob04]: Strukturmodell der XP Systemmetapher (4): *Metaphor Entailments Sign*

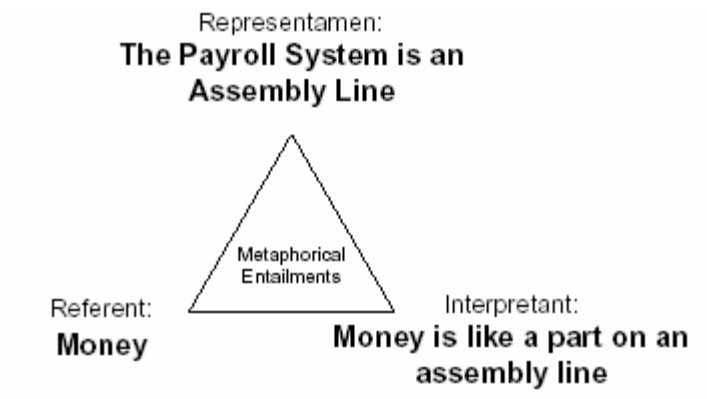
- Das *Metaphor Entailments Sign* modelliert die *Entailments* (metaphorischen Ableitungen) des *Metaphor Introduction Signs*
  - Der *Representamen* dieses Zeichens besteht aus der vollständigen Systemmetapher (der Interpretant des *Metaphor Introduction Signs*)
    - Im C3-Projekt: *Payroll System is an assembly line*
  - Der *Referent* besteht aus einem Merkmal/Kennzeichen des Tenors der Metapher (einer wichtigen Komponente oder Operation).
    - Im C3-Projekt z.B.: *Money, paycheque, pension deduction* (Operation)
    - Es ex. viele potentiell Merkmale. Zu einem Zeitpunkt wird immer eines betrachtet.

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (5): *Metaphor Entailments Sign*

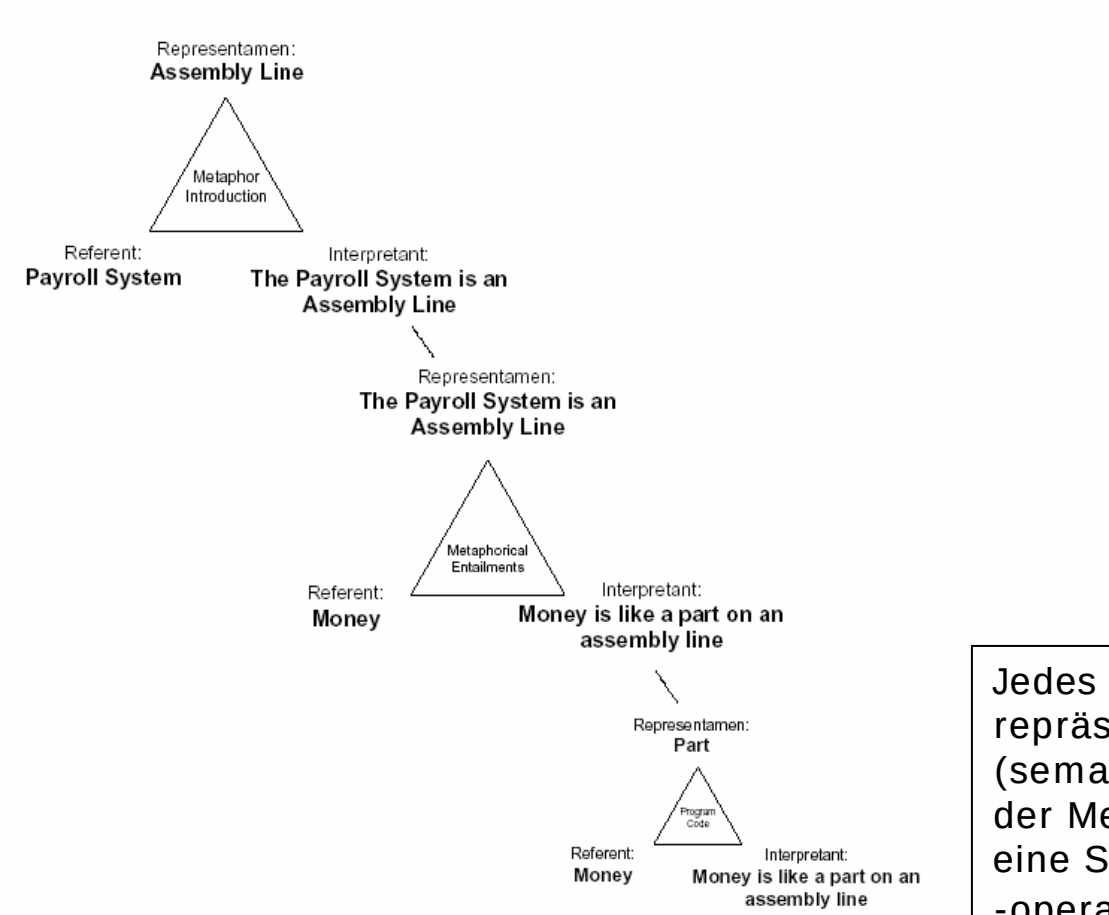
- Der *Interpretant* besteht aus dem Vergleich eines Konzeptes (oder einer Aussage) welches dem *Vehicle* zugeschrieben wird mit dem *Tenor*. Das Ergebnis ist eine Aussage, welche einen Aspekt des *Vehicles* auf den *Tenor* bezieht.
  - Im C3-Projekt:
    - *Money is like a part on an assembly line.*
      - » Der Gedanke, der hier dahinter steht ist: Gehaltsschecks sind eine Kombination von Zeit, Geld und zusätzlichen regulierenden Faktoren.
      - » Anderer mögliche *Interpretants* sind z.B.:

        A paycheque can be assembled from time and money.

        A pension deducation is like a station task on an assembly line.



[KhaBarBob04]: Strukturmodell der XP Systemmetapher (6)



A structural model of XP metaphor showing one particular entailment

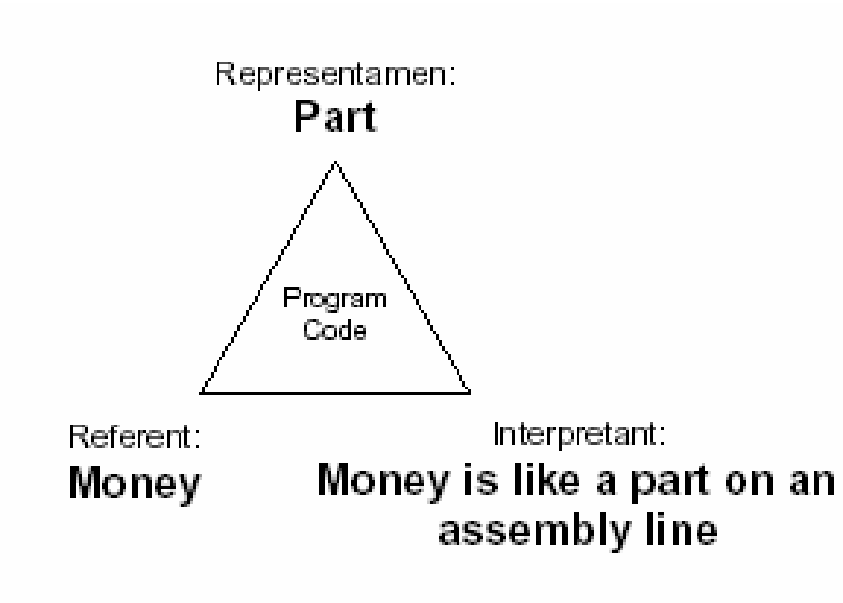
Jedes *Program Code Sign* repräsentiert ein *Entailment* (semantische Implikation) der Metapher auf eine Systemkomponente oder -operation.

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (7): *Program Code Sign*

- Jeder *Interpretant* (eines *Metaphor Entailments Signs*) generiert ein *Program Code Sign.*
  - "[...] is the result of combining concepts from the system metaphor with object-oriented programming."
  - Ein spezieller Fall des *OOP Signs*.
- Jedes Program Code Sign repräsentiert ein Entailment (semantische Implikation) der Metapher auf eine Systemkomponente oder -operation.
  - *Program Code Signs* werden letztendlich durch Code verkörpert.
    - Der *Representamen* ist ein Interpretant eines *Metaphorical Entailment Signs*.
    - Der *Referent* bezieht sich auf einen Teil der Domain.
    - Der *Interpretant* ist das Ergebnis der Anwendung des durch die Metapher inspirierten Konzeptes (~*Referent*) auf die Domain.
  - "The Program Code signs represent the level at which the metaphor supplies ideas that are directly usable for the system, i.e. the coding level."

[KhaBarBob04]: Strukturmodell der XP Systemmetapher (8): *Program Code Sign*

- Beispiel *Programming Concept Model*:
  - Der *Representamen* ist "Part" und der der *Referent* ist das Domain-Konzept "Money"
  - Der *Interpretant* wird "Money is like a part on an assembly line"



[KhaBarBob04]: Strukturmodell der XP Systemmetapher (9): Zusammenfassung

- Das hergeleitete semiotische Modell von Systemmetaphern macht klar, wie eine Systemmetapher arbeitet, um ein System zu beschreiben.
- Die Metapher kann in der Art funktionieren, dass das System auf unterschiedlichen Ebene verstanden werden kann
  - Auf einer *höheren Ebene* können Kunden mit Domainwissen verstehen, wie das System die Domain modelliert (bzw. an der Modellierung teilnehmen).
  - Auf einer *niedrigeren Ebene* können Entwickler mit Technologiekenntnissen das Modell verstehen, welches sie implementieren sollen.
  - Das verwendete Vokabular wird zw. den Ebene geteilt.
  - ⇨ Bessere und schneller Kommunikation, d.h. verbesserte Agilität.
  - ⇨ Unterstützung des vorgestellten Grundprinzips von XP!
- Was wir noch gerne hätten: Das Modell soll uns helfen eine Metapher
  - zu finden und
  - zu bewerten.

[KhaBarBob04]: Finden einer Systemmetapher (1): Brainstorming potentieller Metaphern

- **Aufgabe**: Finde zu einem gegebenen zu entwickelnden System (*Tenor*) ein passendes *Vehicle,* um es zu beschreiben. Hierbei soll dieses zu brauchbaren *Metaphorical Entailments* führen können.
- **Methode (auf Basis des Modells in drei Schritten)**:
  - **Schritt 1 (Brainstorm 1):** Sammlung einer Menge potentieller Metaphern.
    - Beteiligt ist das gesamte XP Team (insbesondere auch der Kunde bzw. dessen Vertreter)
    - "Inspirations-Ausgangspunkt": Nachdenken über Wege, das System und seine Funktionalität einem Kreis von Nichtfachleuten zu erklären.
    - Beispiel (*Bank Account*) für einen Ausgangspunkt: "Ein Bankkonto ist ein Behälter, in den unterschiedliche Parteien Inhalt hineinfüllen oder entnehmen können. Die Menge des Inhaltes im Container ist wichtig."
      - Vorgeschlagene "umliegende" Metaphern: "Stausee", "Parkplatz", "Stadt", "Krankenhaus"
    - Im Konsens die n (z.B. 3) am meisten versprechensten wählen.

[KhaBarBob04]: Finden einer Systemmetapher (2): Brainstorming Entailments

- **Schritt 2 (Brainstorm 2):** Aufdecken, ob eine Metapher für das Zielsystem anwendbar ist.
  - Erster Schritt bei der Bestimmung der Tauglichkeit einer Metapher.
  - *Entailments* bestimmen: Suchen nach *Interpretants* und *Referents* des *Metaphorical Entailments Signs* (auf Basis der festgelegten Kandidaten für die Systemmetapher)
  - Beteiligt ist das gesamte XP Team (zuerst jeder alleine, dann zusammen)
  - Möglicher "Ausgangspunkt": Zuerst Fokussierung auf eine Eigenschaft des Systems und Betrachtung dieser im Licht der Metapher.
    - Beispiel von *Entailments* für die Metapher "Ein Bankkonto ist ein Stausee":
      Der Kontostand verhält sich wie der Pegelstand.
      Einzahlen ist wie Wasserzufluss.
      (Maximaler) Dispositionskredit ist wie minimaler Wasserstand.
      Kontoführung ist wie Wasseraufbereitung.
  - Es kann auch umgekehrt (zuerst über mögliche *Entailments* nachdenken) vorgegangen werden.
  - Nach dem Brainstorm über die *Entailments* sollte das Team in der Lage sein, die Kandidaten zu beurteilen und den besten für das weitere Vorgehen auszuwählen (siehe Schritt 3).

[KhaBarBob04]: Finden einer Systemmetapher (3): Auswahl aus den Kandidaten

- **Schritt 3:**
  - Die gefundenen *Entailments* der Kandidaten im Team bewerten.
    - In Hinsicht auf das System nicht korrekte, untereinander nicht konsistente oder einen falschen Fokus setzende *Entailments* streichen.
      - » Sechs Kriterien für solche Bewertungen folgen auf den nächsten Folien.
    - Es sollten nur diejenigen Entailments behalten werden, welche mit der bekannten Arbeitsweisen des Systems konsistent sind und diese unterstützen.
  - Als Ergebnis sollte schließlich klar werden, wie gut die Kandidaten zu einem dienlichen Vokabular beitragen und die bekannten Systemkomponenten und –funktionen beschreiben.
  - Die Systemmetapher kann gewählt werden.
    - Evtl. muss/kann auch eine Kombination von Kandidaten gewählt werden.

[KhaBarBob04]: Bewerten von Systemmetaphern (1):

- Wie gut funktioniert die Metapher? – Drei Kriterien
  1. Enthalten die *Entailments* der Metapher "programmierbare" Ideen?
     - Die durch die Metapher "inspirierten" *Program Code Signs* sollten konsistent über die gesamte Metapher sein!
     - Nur dann stellt die Metapher einen umfassenden Zusammenhang her, welcher die Wahrscheinlichkeit eines "geteilten" Verständnisses verbessert!
     - Nur ein "geteiltes" Verständnis kann zu konsistentem und einfachem Code führen (Man denke an die XP-Werte!)
  2. Behandelt die Metapher die Hauptsystemkomponenten und ihre (bekannten) Funktionalitäten?
     - Existieren *Program Code Signs* für die Hauptkomponenten und –funktionen?
     - Beispiele für Hauptsystemkomponenten beim *Bank Account*: *bank accounts*, *deposite*, *overdraft*, *account management facilities* (korrespondieren zu *reservoir*, *inflow*, *outflow*, *minimum water level*)

  ⇨ Kriterium eins und zwei sind dahingehend ähnlich, dass sie die Beziehung zwischen Metapher und System überprüfen.

[KhaBarBob04]: Bewerten von Systemmetaphern (2):

3. Stellen die *Entailments* ein Vokabular zur Verfügung, mit welchem sich das System beschreiben lässt?
   - Die *Entailments* selber haben nicht immer notwendigerweise direkt mit Aspekten der Programmierung zu tun.
   - Sie können aber eine für die *Kommunikation* im Team (Entwickler und Kunden) nützliche Beschreibung der Funktionsweise des Systems liefern.
   - Z.B. liefert die Stausee-Metapher Vokabeln und Konzepte wie "Sammelbecken", "Zustrom", "Abfluss", "Aufbereitung"…

[KhaBarBob04]: Bewerten von Systemmetaphern (2):

- Ist die Metapher zu schwach? – Ein Kriterium
  1. Welche Systemkomponenten oder Aktionen werden durch die Metapher nicht beschrieben?
     - Systemmetaphern, die alle zentralen Systemkomponenten und -funktionen berücksichtigen, sind eher selten.
     - Diesem Umstand kann manchmal nur durch die Kombination mehrerer Metaphern begegnet werden.
     - Hierzu muss klar sein, was nicht von den einzelnen Metaphern abgedeckt wird.
       - Ein Beispiel: Die Stauseemetapher liefert den Zufluss als Metapher für das Einzahlen.
         - » Der Zufluss ist allerdings stetig, währen Einzahlungen diskret verlaufen.
         - » Der Aspekt des Diskreten bleibt also durch die Metapher unbeschrieben.

[KhaBarBob04]: Bewerten von Systemmetaphern (3):

- Ist die Metapher irreführend? – Zwei Kriterien
  1. Folgen aus den *Entailments* nichtexistente Systemkomponenten oder Verhaltensweisen?
     - Dies lässt sich natürlich nicht völlig verhindern, so dass es vorkommt, dass Entailments gestrichen werden müssen.
     - Allerdings: Jedes verworfenen Entailment einer Metapher schwächt die generischen Fähigkeiten einer Metapher und macht diese somit weniger hilfreich.
  2. Machen die *Entailments* das System komplizierter als nötig?
     - Im Auge behalten: Metaphern sollen das Systemverständnis verbessern, nicht verdunkeln.
     - In diesem Sinne sollte die Metapher dem System nicht überflüssige Komponenten aufzwingen!
     - Z.B. bei der Stausee-Metapher: Erdbebenschutz, Filter, Wasserdruck, Salzgehalt…

Das Herz von XP ist eine einfache und flexible Systemarchitektur

[KhaBarBob04]: Nachbemerkung

- Es ist nicht klar, ob ausgerechnet die Staussee-Metapher wirklich das Verständnis des Systems verbessert.
  - Schließlich wurde ja nur eine Menge von Synonymen für Systemkomponenten und Systemverhalten eingeführt.
    - Und das evtl. noch nicht einmal vollständig.
  - Außerdem: Höchstwahrscheinlich hat das Team mehr Wissen bzgl. Bankkonten als bzgl. Stauseen.

Quellen

- Rilla Khaled, Pippin Barr, James Noble, and Robert Biddle: **System Metaphor in "Extreme Programming": A Semiotic Approach**
- George Lakoff, and Mark Johnson: **Metaphors We Live By**

Danke