CPM: A Core Product Model for PLM support

Steven J. Fenves, Guest Researcher
Manufacturing Systems Integration Division, NIST
sfenves@cme.nist.gov

in collaboration with Sebti Foufou, Conrad Bock, Rachuri Sudarsan, Ram D. Sriram and others

CURRENT STATUS

1) Many (most?) PDM systems built on top of legacy CAD systems
2) Many (most?) PDM systems are blind to the files they manage
3) Many (most?) commercial PLM support systems built on top of PDM systems

What is wrong with this picture?

CONSEQUENCES

1) Can only represent the product’s form (more precisely, its geometry)
2) Can retrieve information only by file name (how many people in marketing etc., know the engineers’ file names?)
3) Can support only that segment of design process that deals with the product’s form (embodiment design and later)

WHAT IS NEEDED

A representation that gives equal status to three aspects of the product: its function, its form and its behavior and can therefore support:

- functional reasoning in the conceptual stages
- “traditional” engineering design stages
- behavior modeling in post-design stages

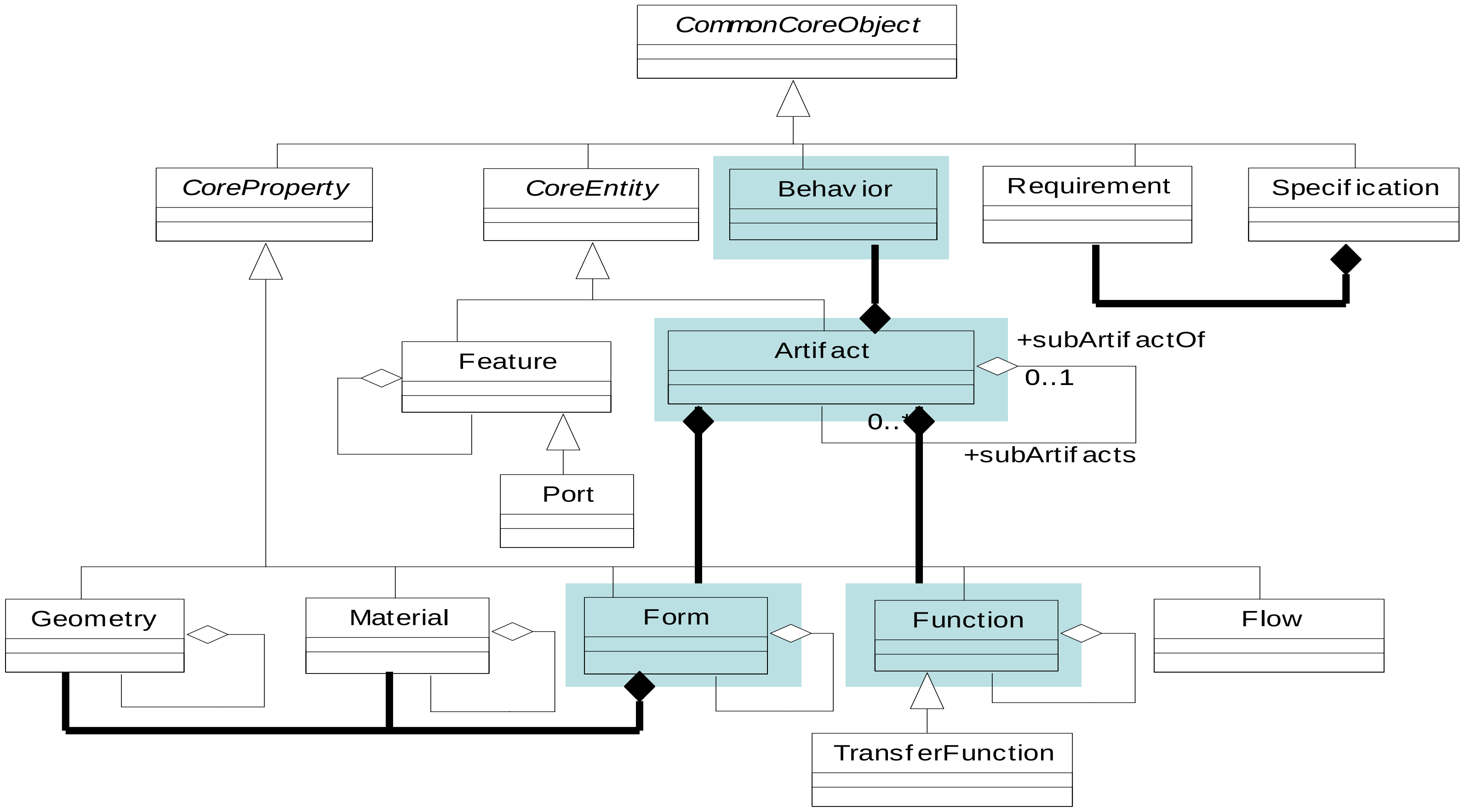
CORE PRODUCT MODEL

- Started as an in-house coordination project
- Evolved into conceptual data model for PLM support
- An abstract model with generic semantics

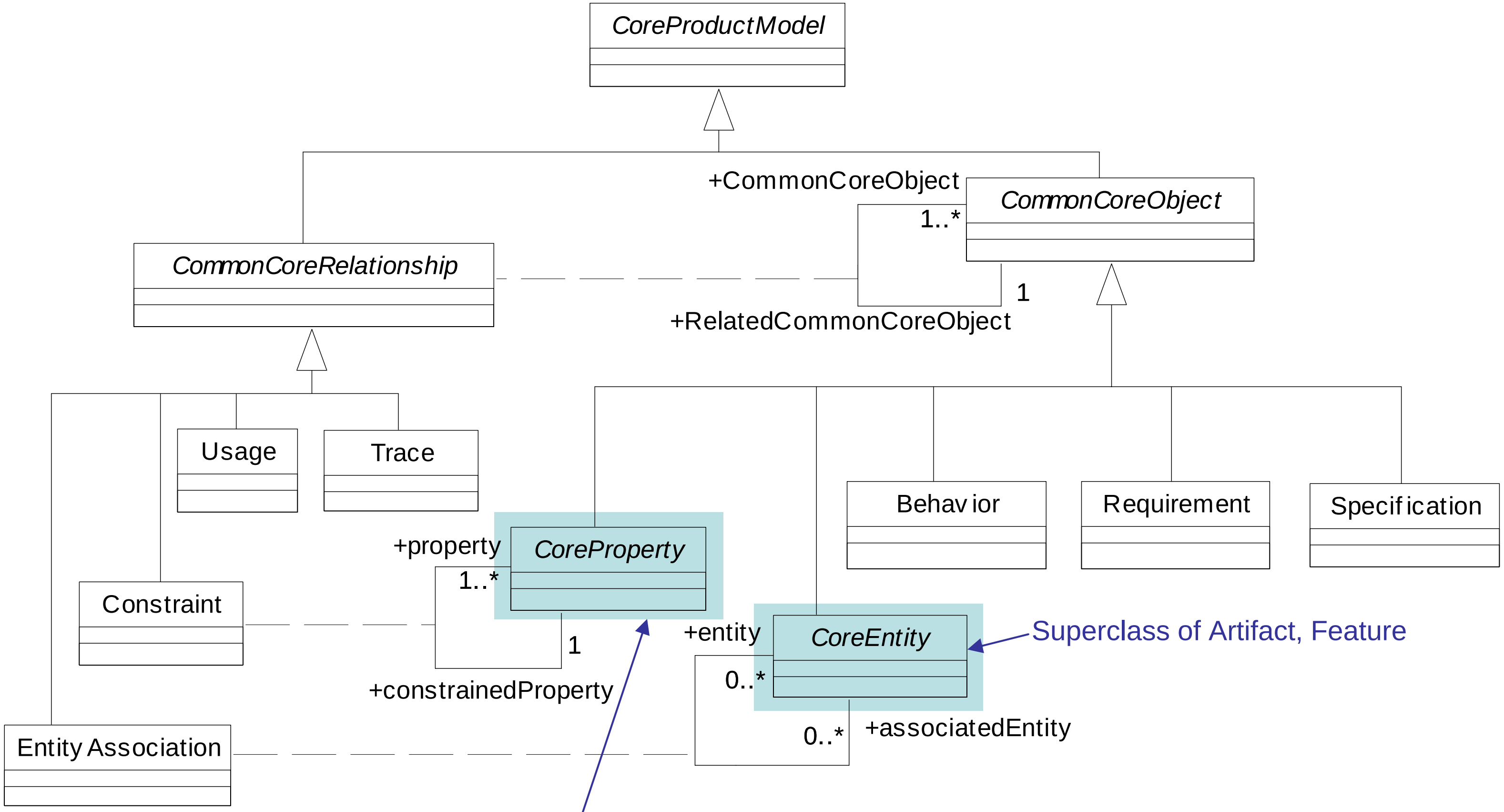
KEY CONCEPTS

- CPM is based on the concepts of **Artifact** and **Feature**:
  - **Artifact** = a distinct entity (component, part, subassembly, assembly)
  - **Feature** = a portion of the artifact's form with some specific function (design feature, analysis feature, …)
- **Artifact** is the aggregation of a triad:
  - **Function** = what the artifact is supposed to do; synonymous with the term *intended behavior*.
  - **Form** = the proposed design solution; modeled in terms of **Geometry** and **Material**.
  - **Behavior** = how the artifact's form implements its function; application of a behavioral model simulates the *observed behavior* of the artifact's form.

CPM OBJECT CLASSES



RELATIONSHIP CLASSES



Superclass of Artifact, Feature

Superclass of Form, Function

LEVELS OF CPM

CPM eventually to exist at three levels:

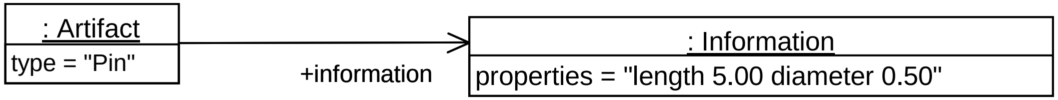
- conceptual
- intermediate
- implementation

CONCEPTUAL LEVEL

- As presented: abstract model without domain-specific semantics
- Suitable for developing extensions:
  - Assembly model
  - Product family evolution model
  - Design-analysis integration model
  - etc

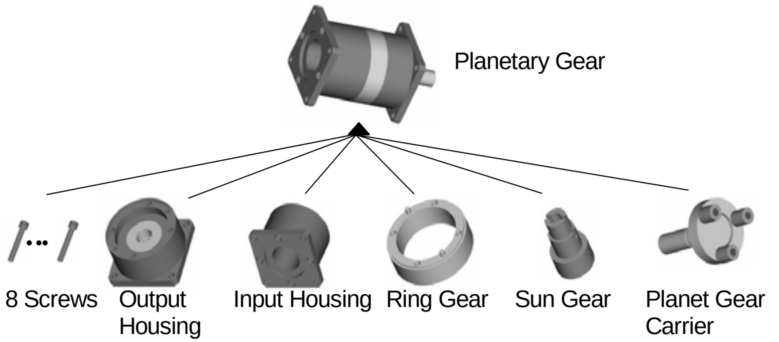
INTERMEDIATE MODEL

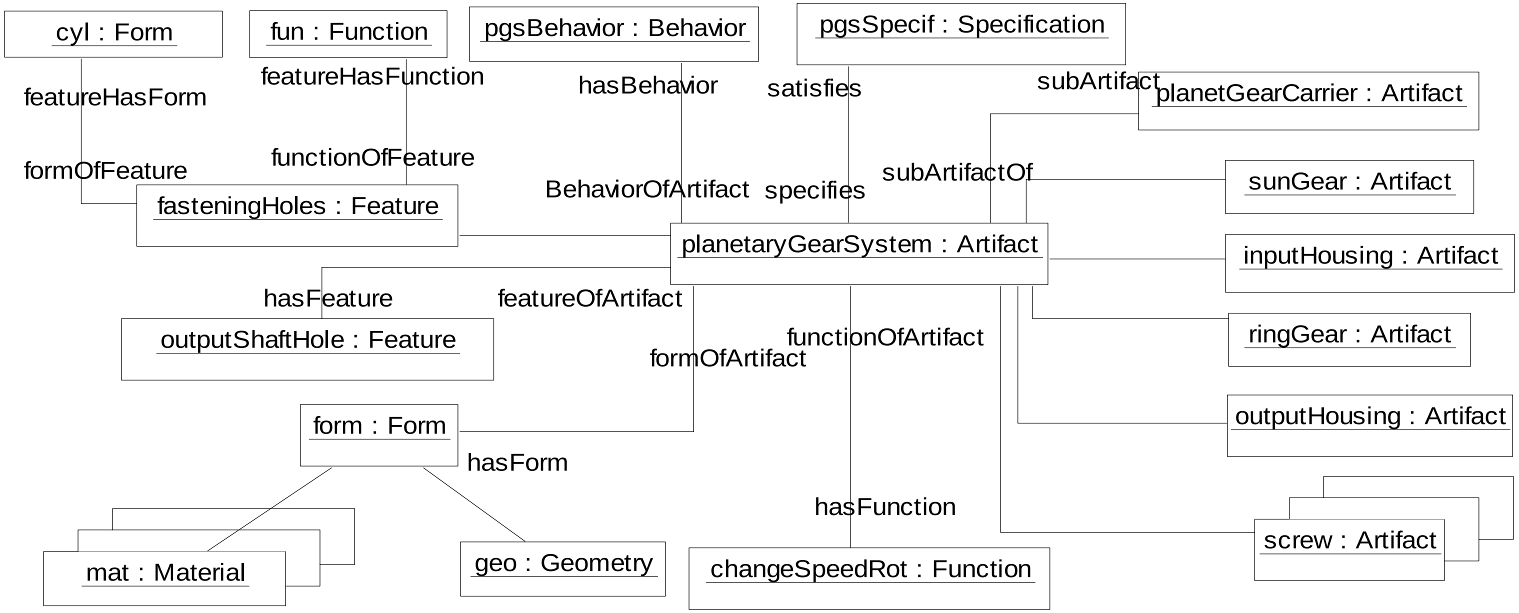
- Suitable for low-volume proof-of-concept applications, e. g., the NIST design repository – semantics still buried
- Two enabling mechanisms:
  - each object has an attribute **`type`**; can be used to build taxonomies
  - each object has an associated object **`Information`** with an attribute **`properties`**, consisting of a list of attribute-value pairs



- UML-to-XML converter available

AN EXAMPLE





cyl : Form

fun : Function

pgsBehavior : Behavior

pgsSpecif : Specification

featureHasForm

featureHasFunction

hasBehavior

satisfies

subArtifact

planetGearCarrier : Artifact

formOfFeature

functionOfFeature

BehaviorOfArtifact

specifies

subArtifactOf

sunGear : Artifact

fasteningHoles : Feature

planetaryGearSystem : Artifact

inputHousing : Artifact

hasFeature

featureOfArtifact

ringGear : Artifact

outputShaftHole : Feature

functionOfArtifact

formOfArtifact

form : Form

outputHousing : Artifact

hasForm

hasFunction

mat : Material

geo : Geometry

changeSpeedRot : Function

screw : Artifact

IMPLEMENTATION MODEL

- None exists yet
- Facilities provided to aid a model compiler:
  - create subclasses of **`Artifact`** from the classification hierarchy in the **`type`** slot; and
  - define attributes on the subclasses from the attribute names in **`properties`**
- Full application domain semantics can be supported/enforced

CONCLUSIONS

- A bit of serendipity: in-house effort with outside potential
- Conceptual level well explored, but questions remain:
  - do features have (independent) behaviors?
  - do we need to introduce **`Structure`** and/or **`Technology`** as top aspects of **`Form`**?
  - what other generic concepts are needed?
- The important questions are:
  - is the model robust enough for implementation?
  - is anyone interested in implementing it (NIST can't)?
- Your answers to these questions are welcome

sfenves@cme.nist.gov