



The Fun of Connection

Abstraction for Data Systems

Chet Mancini

Github & Twitter: @chetmancini

LEGO BASIC

545



6833

LEGOLAND

LEGO





Duplo®

Godfred K. Christiansen
US Pat. 3597875 / Aug. 10, 1971

Fischertechnik® K'Nex®

Artur Fischer
US Pat. 3464147 / Sep. 19, 1966

Joel Glickman & M. Doepner
US Pat. 5061219 / Dec. 11, 1990

Krinkles®

Walter Heubl
US Pat. 3603025 / Sep. 30, 1968

Lego®

Godfred K. Christiansen
US Pat. 3005282 / Jul. 28, 1958

Lincoln Logs® Tinkertoy®

John Lloyd Wright
US Pat. 1351086 / Jan. 8, 1920

Tinkertoy®

Charles H. Pajou
US Pat. 1113371 / Jul. 8, 1914

ZomeTool®

Steven Rogers & P. Hildebrandt
US Pat. 6840699 / Nov. 1, 2002*

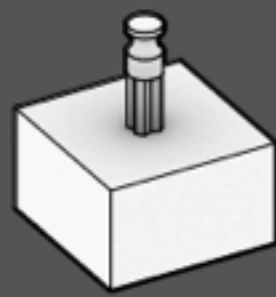
Zoob®

Michael J. Grey
US Pat. 5897417 / Dec. 11, 1996*

(Duplo to Duplo)



uck-0001m

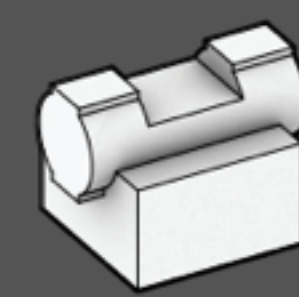


uck-0003m

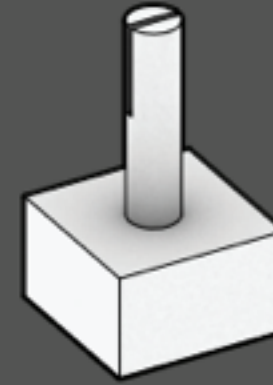


uck-0004m

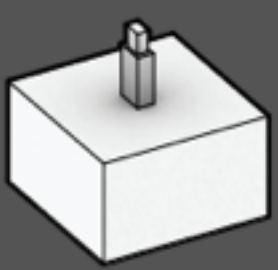
(Lego to Duplo
already supported
by manufacturer)



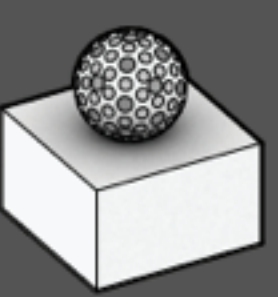
uck-0006m



uck-0007m

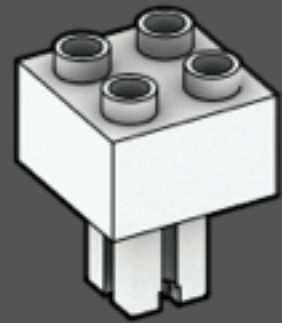


uck-0008m



uck-0009m

Duplo®



uck-0100m

(Fischertechnik to
Fischertechnik)



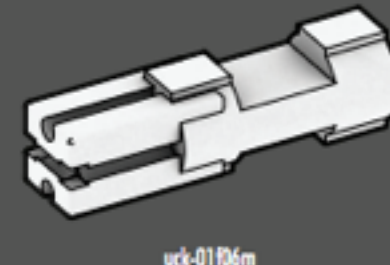
uck-0103m



uck-0104m



uck-0105m



uck-0106m



uck-0107m



uck-0108m



uck-0109m

Fischertechnik®



uck-0200m



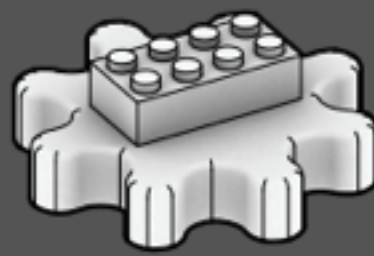
uck-0201m



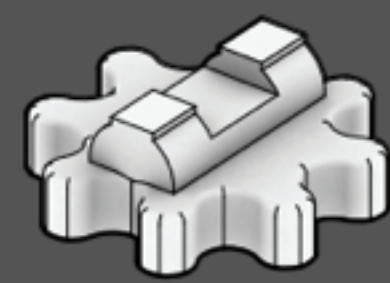
uck-0203m



uck-0204m



uck-0205m



uck-0206m



uck-0207m



uck-0208m



uck-0209m

Gears!
Gears!
Gears!®



uck-0300m



uck-0301m



(K'Nex to K'Nex)



uck-0304m



uck-0305m



uck-0306m



uck-0307m



uck-0308m



uck-0309m

K'Nex®



(Krinkles to Krinkles)



Krinkles®

(Krinkles to Krinkles)

Krinkles®



uck-0400m



uck-0401m



uck-0403m



uck-0405m



uck-0406m



uck-0407m



uck-0408m



uck-0409m

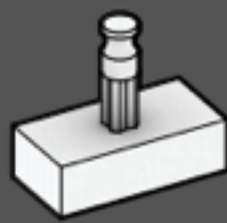
(Lego to Duplo
already supported
by manufacturer)

(Lego to Lego)

Lego®



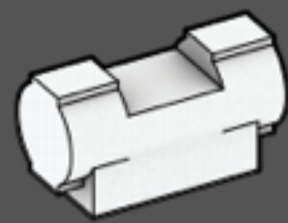
uck-0501m



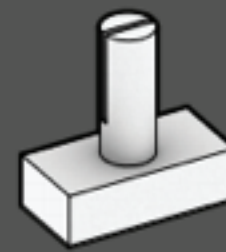
uck-0503m



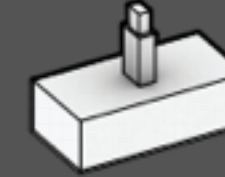
uck-0504m



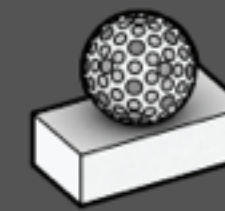
uck-0506m



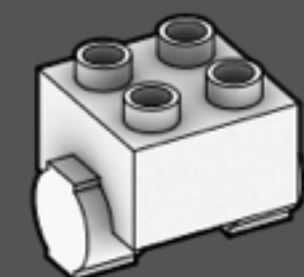
uck-0507m



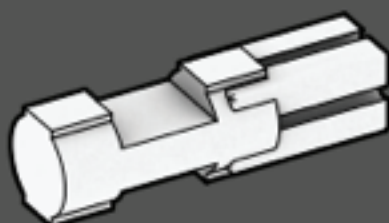
uck-0508m



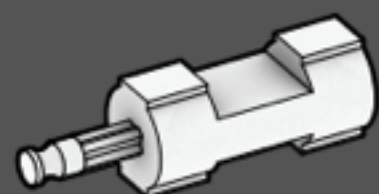
uck-0509m



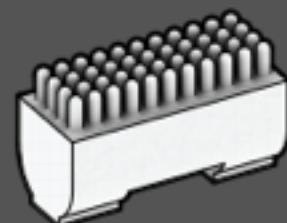
uck-0600m



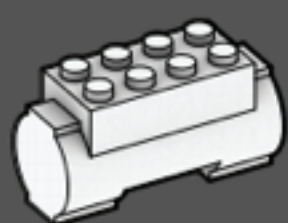
uck-0601m



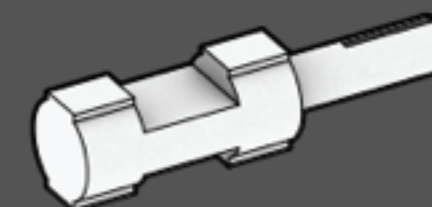
uck-0603m



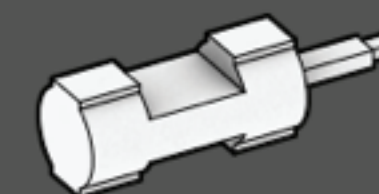
uck-0604m



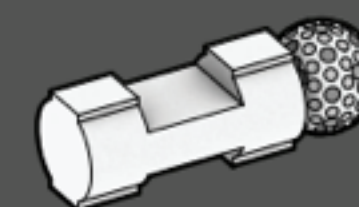
uck-0605m



uck-0607m



uck-0608m

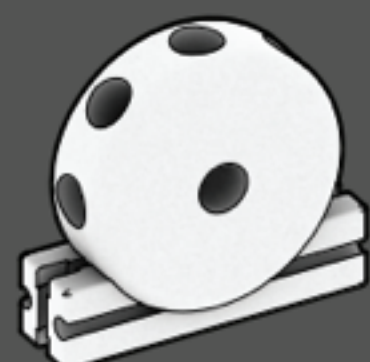


uck-0609m

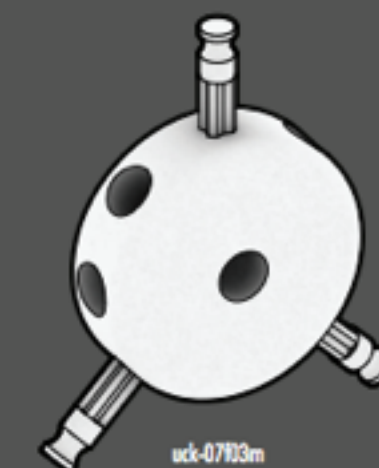
Lincoln Logs®



uck-0700m



uck-0701m



uck-0703m



uck-0704m



uck-0705m

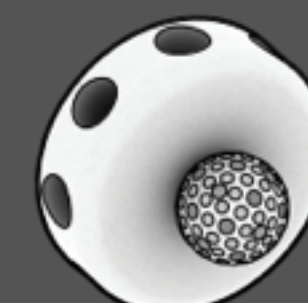


uck-0706m

(Tinkertoy to
Tinkertoy)

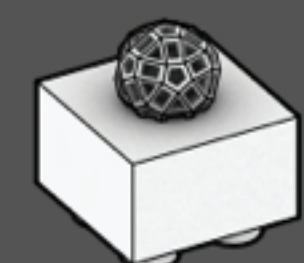


uck-0708m

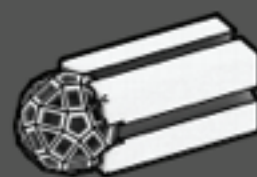


uck-0709m

Tinkertoy®



uck-0800m



uck-0801m



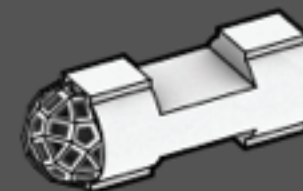
uck-0803m



uck-0804m



uck-0805m



uck-0806m



uck-0807m

(ZomeTool to
ZomeTool)



uck-0809m

ZomeTool®



uck-0900m



uck-0901m



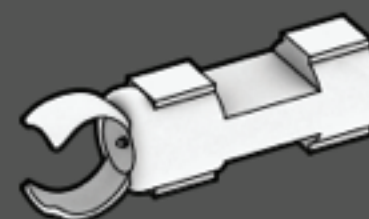
uck-0903m



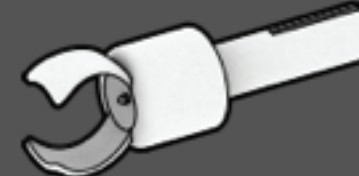
uck-0904m



uck-0905m



uck-0906m



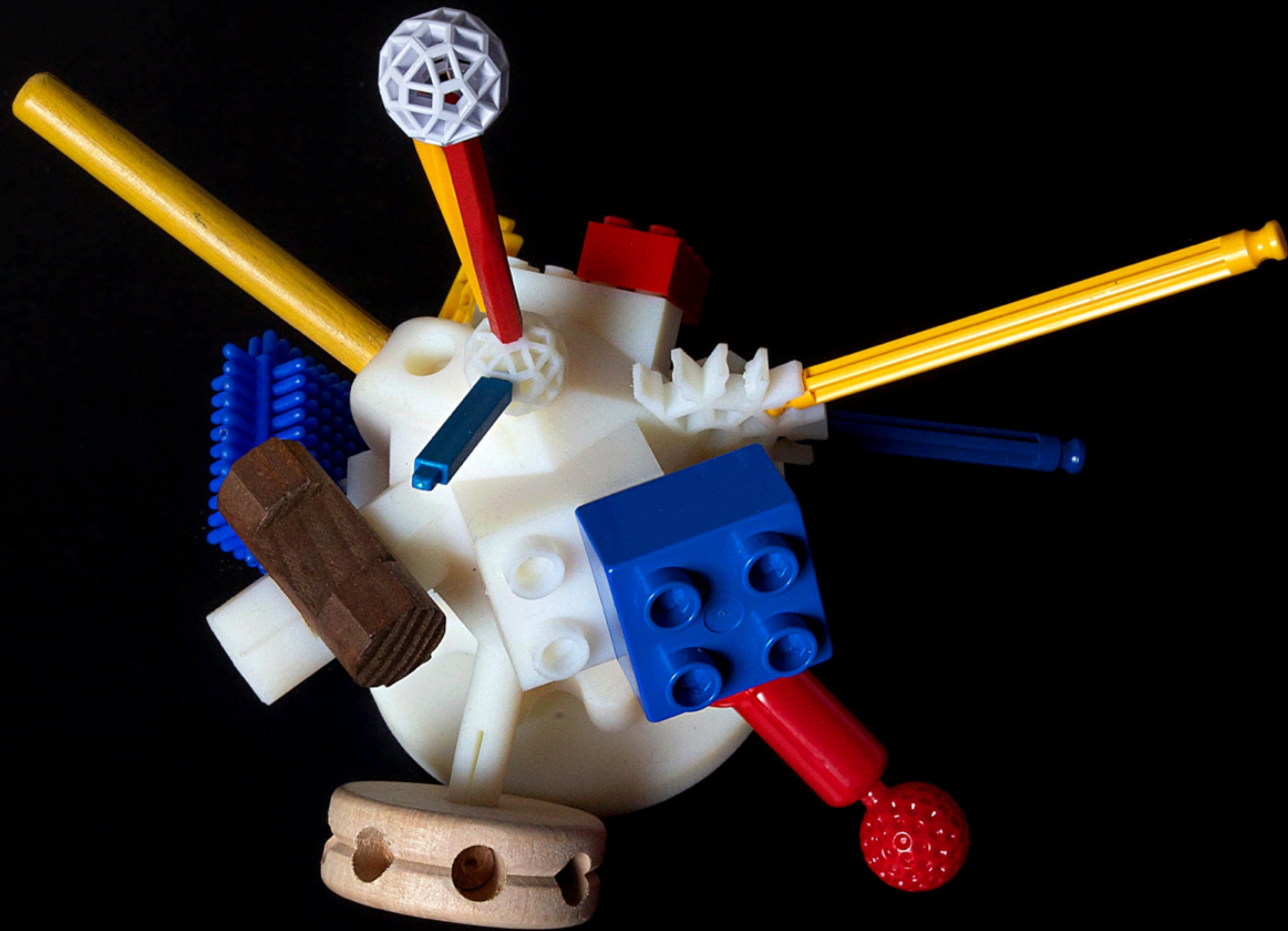
uck-0907m



uck-0908m

(Zoob to Zoob)

Zoob®



<http://fffff.at/free-universal-construction-kit/>

“I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages”

–ALAN KAY, writing on OOP

github.com/cognitect/transit-format

- A format & set of elements and extensions for representing typed values.
- Encoded as valid JSON or Msgpack
- Self-describing. Schemas and convention not required.
- Standard libraries for Clojure, JavaScript, Java, Python, and Ruby.

github.com/Prismatic/schema

github.com/intentmedia/schema-transform

- Transform an Avro schema to Prismatic Schema.

- Transform Prismatic to string in Avro format.

- Maintains major primitive types and supports complex types like arrays, maps, and records.

Fields can be optional or required.

```
(defschema Record
  {:name      String
   :keywords  [String]
   :updated-at Long})
```

Nulls

- no field at all
- NULL / NIL / NA / NAN
- “”
- “unspecified whitespace”
- “NULL”
- -1 or magic number
- “NOT_FOUND” / “MISSING” / “UNKNOWN”

- Have you weighed performance / convenience? Is it human readable?
Does it need to be?
- Are you maintaining type information? Can you maintain data structures?
- Do you have common semantics for boundary cases like NULLs?
- Can you extend and modify your schema safely? Is it versioned?
- Can you extend into new language ecosystems?

- *Easy to use.* Does communicating with other components feel native?
- *Easy to extend.* Can you add new components? New languages? New fields? Without downtime?
- *Easy to generate.* Can we pass around a data blueprint? Automate manual coding?

Fear no data

chetmancini.com

Github/Twitter: @chetmancini

Thanks to these companies:

intentmedia/schema-transform

prismatic/schema

cognitect/transit-format

Free Art and Technology [F.A.T.] Lab and Sy-Lab. “The Free Universal Construction Kit.”

fffff.at/free-universal-construction-kit