System Architecture

IAP Lecture 2

Ed Crawley

January 11, 2007

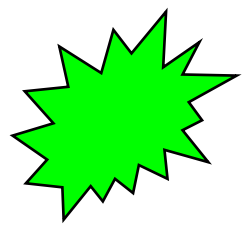
Rev 2.0

Today's Topics

- Definitions - Reflections
- Form
- Function
- Reference PDP - "In the Small"

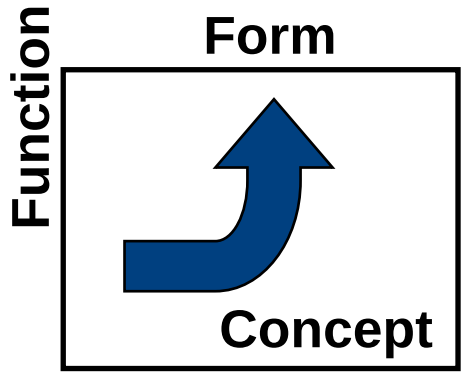
¿ Reflections on Definitions?

- System
- Complex
- Value
- Product
- Principle/Method/Tool

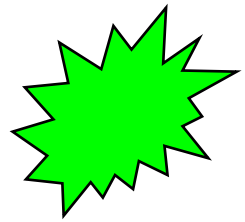


Architecture

- **Consists of:**
  - **Function**
  - **Related by Concept**
  - **To *Form***



¿ Form - Reflections?



- What is the form of simple systems?
- How did you express the decompositional view?
- How did you represent the graphical structural view?
- How did you represent the list like structural view?
- What do the lines in the graphical view, or connections in the list view represent? Is it form?
- Did you identify classes of structural relations?

Form Topics

- **Representing form and structure**
- **Whole product system and use context**
- **Boundaries and Interfaces**
- **Attributes and states**

Representations of Form

Form can be *represented* by:

- **Words (in natural language - nouns)**
- **Code**
- **Illustrations, schematics, drawings**
- **Each discipline has developed shorthand for representing their discipline specific form**

But the form is the actual suggested physical/informational embodiment.

Duality of Physical and Informational Form

- **Physical form *can* always be represented by informational form (e.g. a building and a drawing of a building)**
- **Informational form *must* always be stored or encoded in physical form (data in a CD, thought in neurons, poetry in print)**
- **So there is a duality of the physical and informational world (styled after the duality of particles and waves)**

Classes of Structural Connections

- **Connections that are strictly descriptions of *form*:**
  - **Relative spatial location or topology (e.g. above, is next to, is aligned with, is within, overlaps with, etc.) which refers to previous arranging process**
  - **Information about assembly/implementation (e.g. connected to, bolted to, compiled with, etc.) which refers to previous assembling/implementing process**
- **Connections that are description of *function* while operating (still to be discussed)**

Spatial/Topological Structure - Whistle

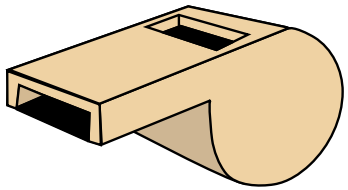
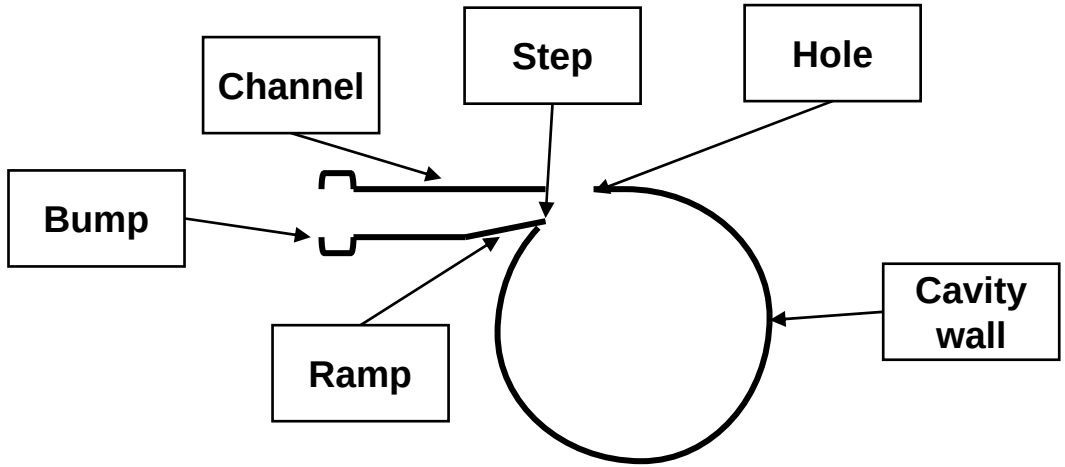
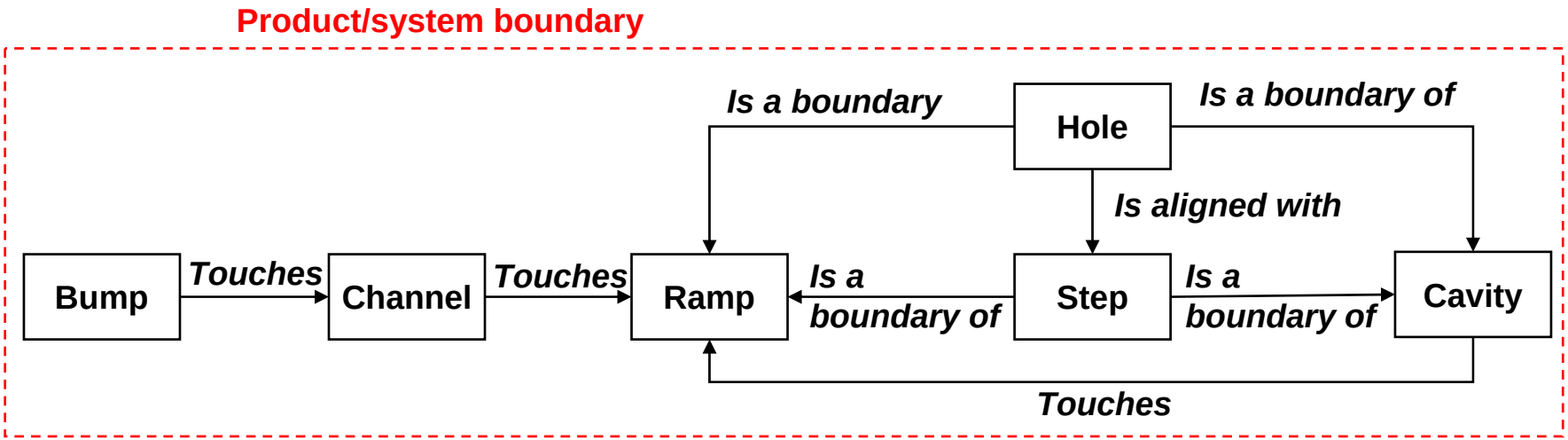


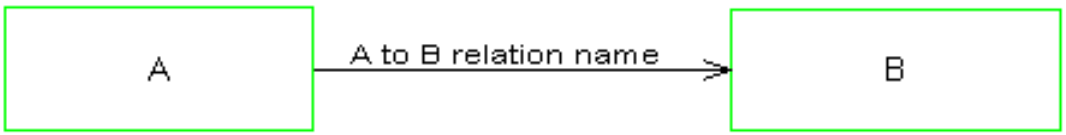
Figure by MIT OCW.



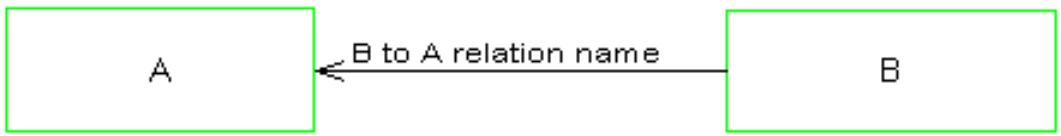


OPM Object-Object Structural Links

- **Defined: A structural link is the symbol that represents a binary relationship between two objects.**

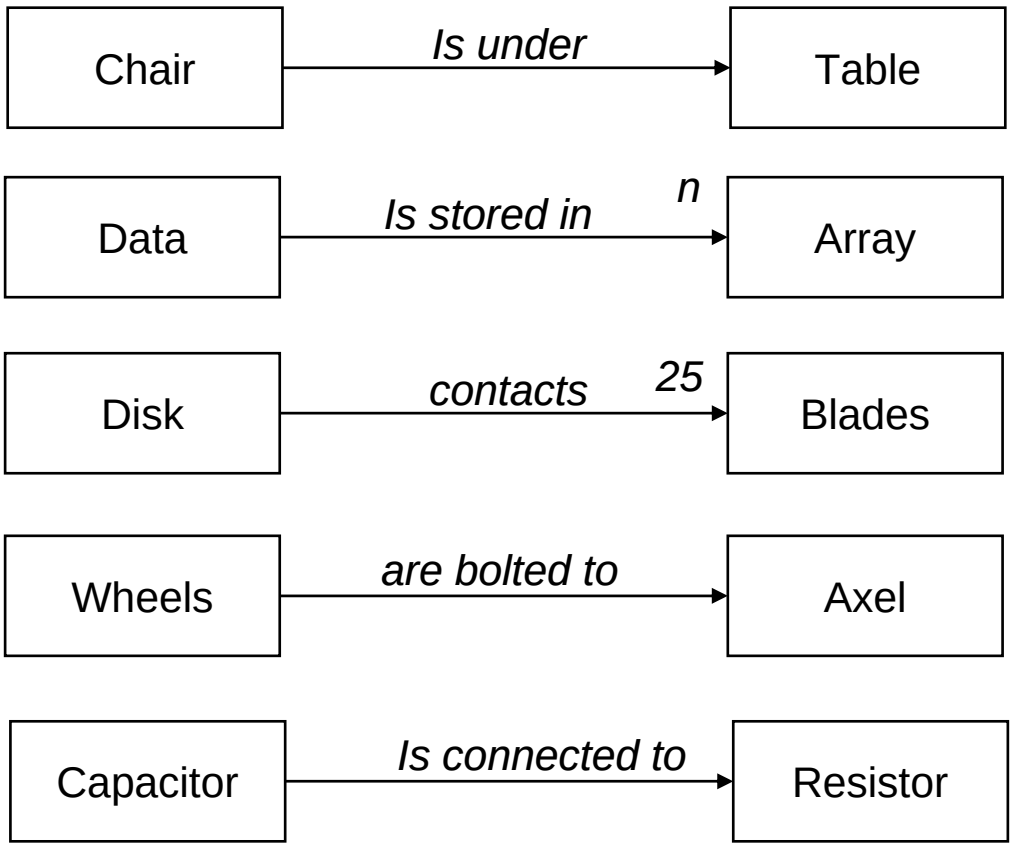


- **There is also a backward direction relation.**



- **Usually it is only necessary to show one, and the other is implicit.**

Structural Link Examples



Spacial/Topological Structure - Whistle - "List"

	Bump	Channel	Ramp	Hole	Step	Cavity
Bump		touches				
Channel	touches		touches			
Ramp		touches		Is bounded by	Is bounded by	touches
Hole			Is a boundary of		Is aligned with	Is a boundary of
Step			Is a boundary of	Is aligned with		Is a boundary of
Cavity			touches	Is bounded by	Is bounded by	

- "N-squared" matrix representation gives a list-like representation of connectivity (read from left row-wise)
- Symmetric (with transformation like "surrounds" to "within", "is a boundary of" to "is bounded by")
- Is non-causal - no sense of anything happening before anything else

Implementation Structure - Whistle

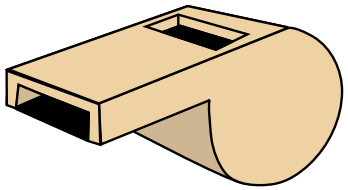
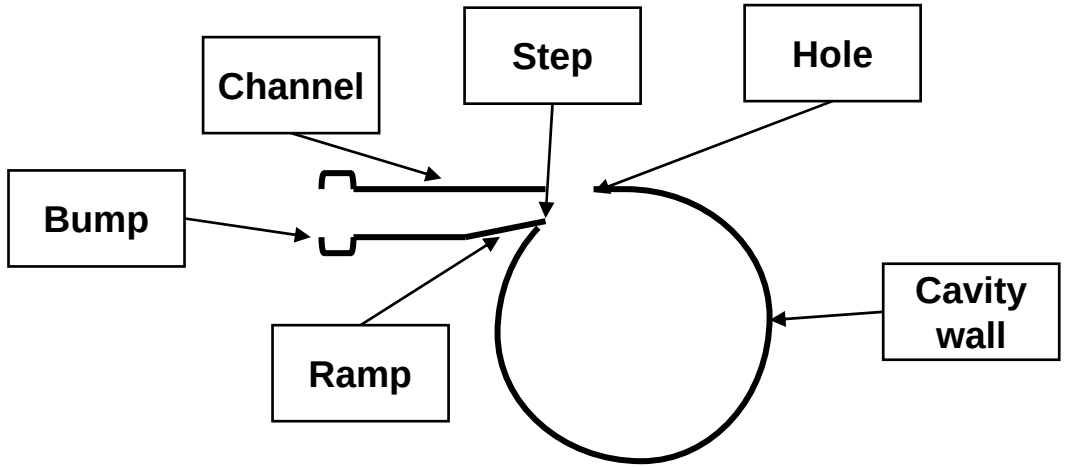
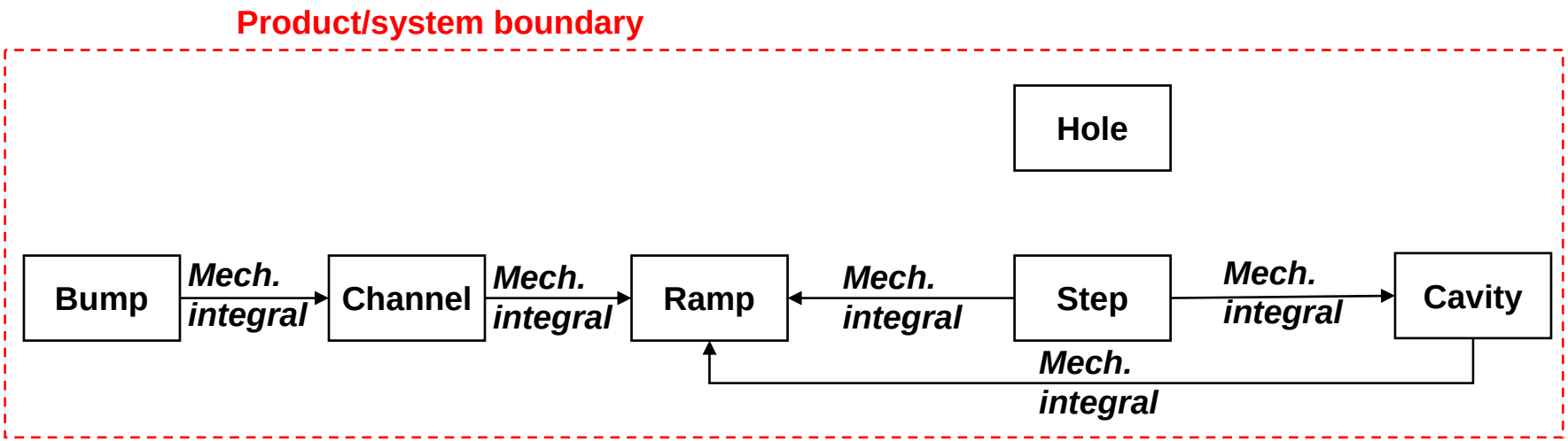


Figure by MIT OCW.





Implementation Structure - Whistle - “List”

	Bump	Channel	Ramp	Hole	Step	Cavity
Bump		Mech. integral				
Channel	Mech. integral		Mech. integral			
Ramp		Mech. integral			Mech. integral	Mech. integral
Hole						
Step			Mech. integral	Mech. integral		Mech. integral
Cavity			Mech. Integral		Mech. Integral	

- “N-squared” matrix representation gives a list-like representation of connectivity (read from left row-wise)
- Symmetric, non-causal - no sense of anything happening before anything else
- Spatial and implementation structure can be combined by showing one in the upper and one in the lower diagonal

Issues Raised

- How do you identify form independent of function?
- How do you define the atomic part level? For hardware? For software?
- Can you “decompose” an integral part? Can a part be a system? Can the features of a part be elements?
- How do you represent the structural interconnections of the elements, as opposed to their “membership” in a system?
- N occurrences of an element - count once or N times? The class or each instance?
- Connectors and interface elements - count as a separate element - or combined with other elements?
- Are there important elements lost in implementation?
- Are there elements important to delivering value other than the product?

Form of Product/System - Questions

- What is the product system?
- What are its principle elements?
- What is its formal structure (i.e. the structure of form)?

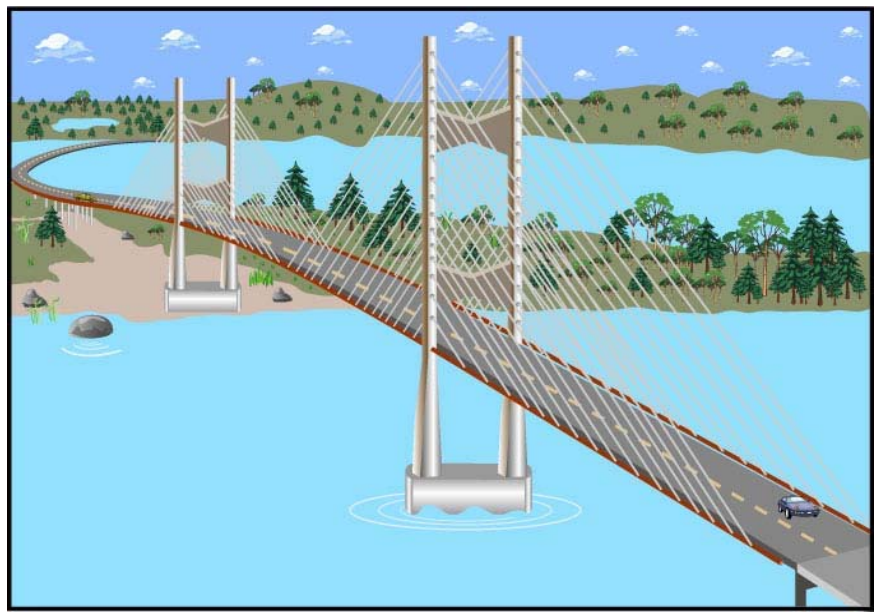


Figure by MIT OCW.

The Whole Product System

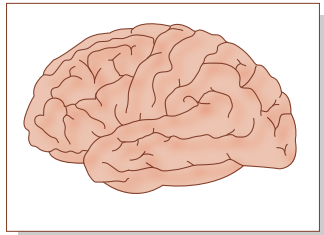
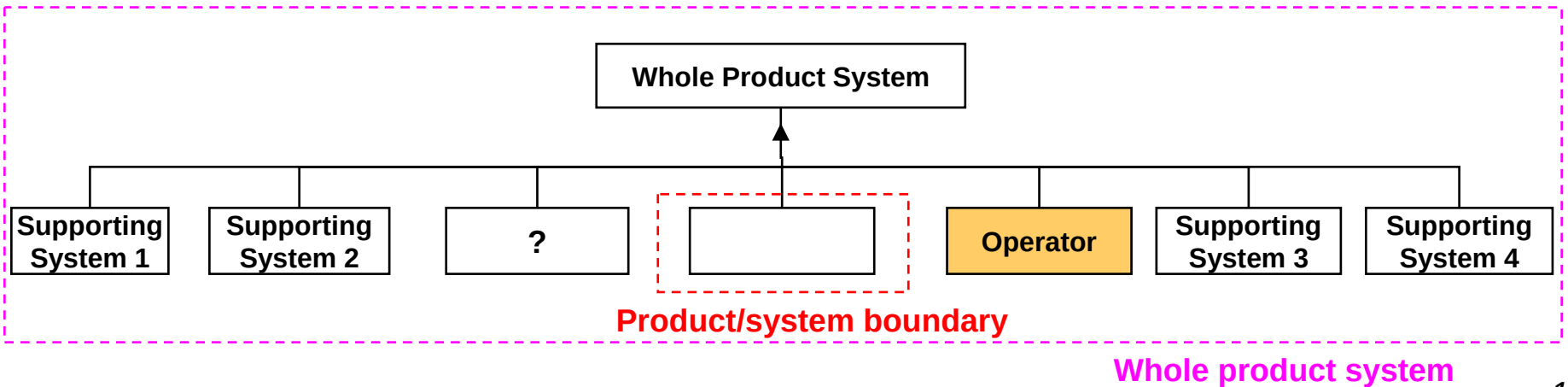


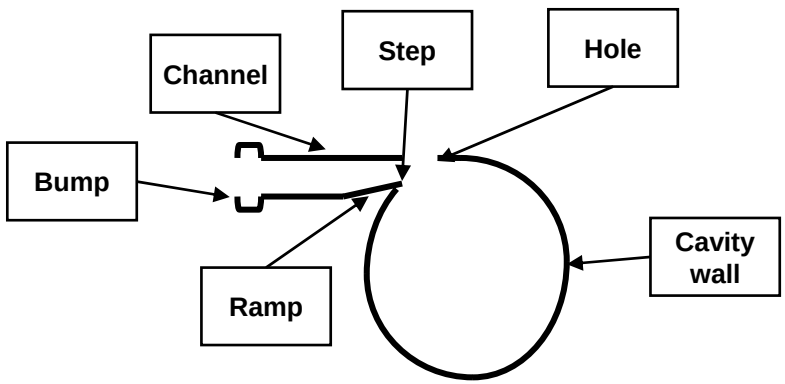
Figure by MIT OCW.

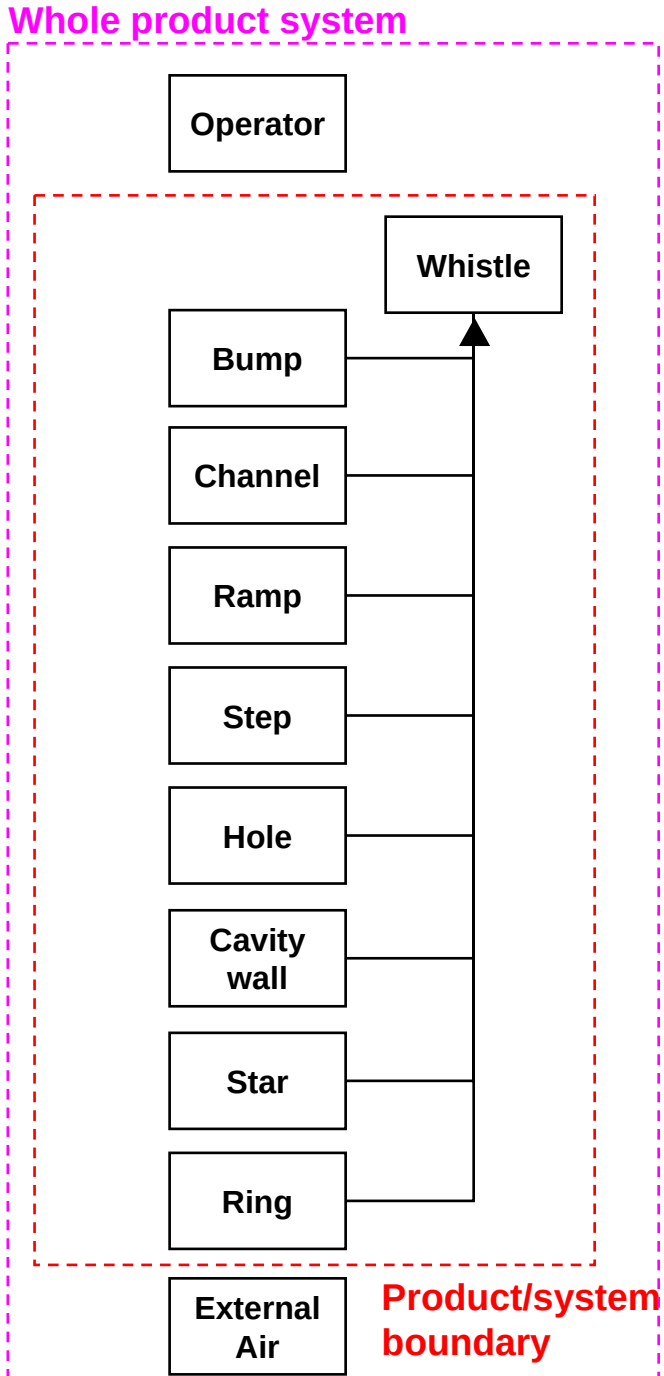
- We usually architect form which is both a product and a system, and which we designate the product/system
- Often for the product/system to deliver value, it must be joined and supported by other supporting systems
- Most often, one of the other objects in the supporting system is the operator, the intelligent (usually human) agent who actually operates the system when it is used
- Together, the product/system, plus these other supporting systems, constitute the whole product system



Whole Product System - Whistle

- **The whistle only requires an operator and the air to work**





Accountability for the Whole Product System

- **Even though you only supply the product/system, all of the other supporting systems must be present *and* work to yield value -BEWARE!!!**
- **They may fail to be present, or fail to work**

- **You will find that you are:**
- ***Responsible* for the product system - BUT**
- ***Accountable* for the whole product system**

- **What architecting approaches can you use to make sure this is not a problem?**

Use Context

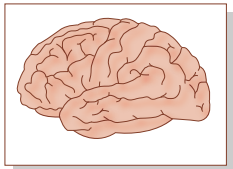
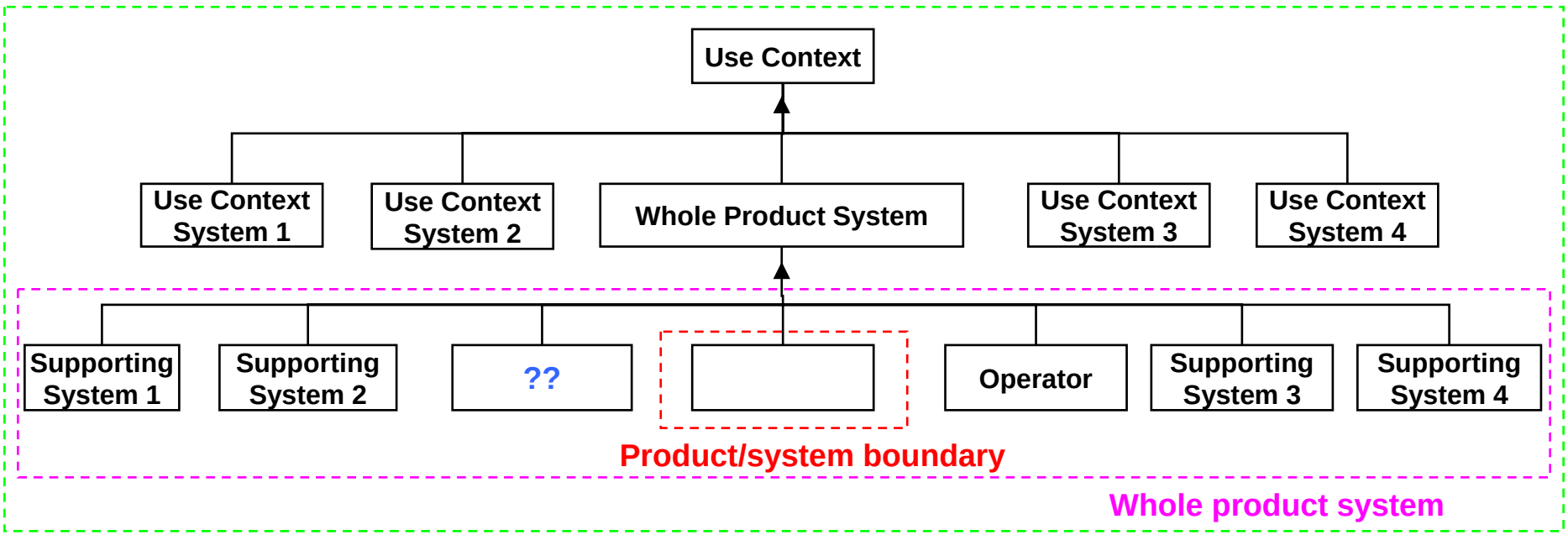


Figure by MIT OCW.

- The whole product system fits within a use context, which included the objects normally present when the whole product operates, but not necessary for it to deliver value
- The product/system usually interfaces with all of the objects in the whole product system, but not with the objects in the use context
- The whole product system is usually use independent, while the use context is context dependent
- The use context informs the function of the product/system



Use Context Informs Design

- **The use context informs the requirements for and the design of the product/system**

- **For example:**
  - **Whistle use context: sporting event, toys, Olympic stadium?**
  - **Op amp use context: lab bench, consumer audio, spacecraft?**

The architect must have a view of three layers of the form:
The product/system, the whole product and use context

Boundaries and Interfaces

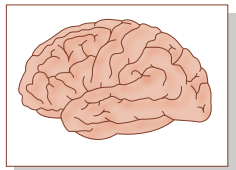


Figure by MIT OCW.

- **The product/system is separated from other supporting systems and the operand by a boundary**
- **The boundary is vital to the definition of architecture, because it defines:**
  - **What you architect, eventually deliver and are responsible for**
  - **What is "fixed" or "constrained" at the boundaries**
- **Everything that crosses a boundary must be facilitated by an interface**
- **It is good practice to draw the system boundary clearly on any product/system representation, and identify interfaces explicitly**

Camera - Whole Product & Boundaries

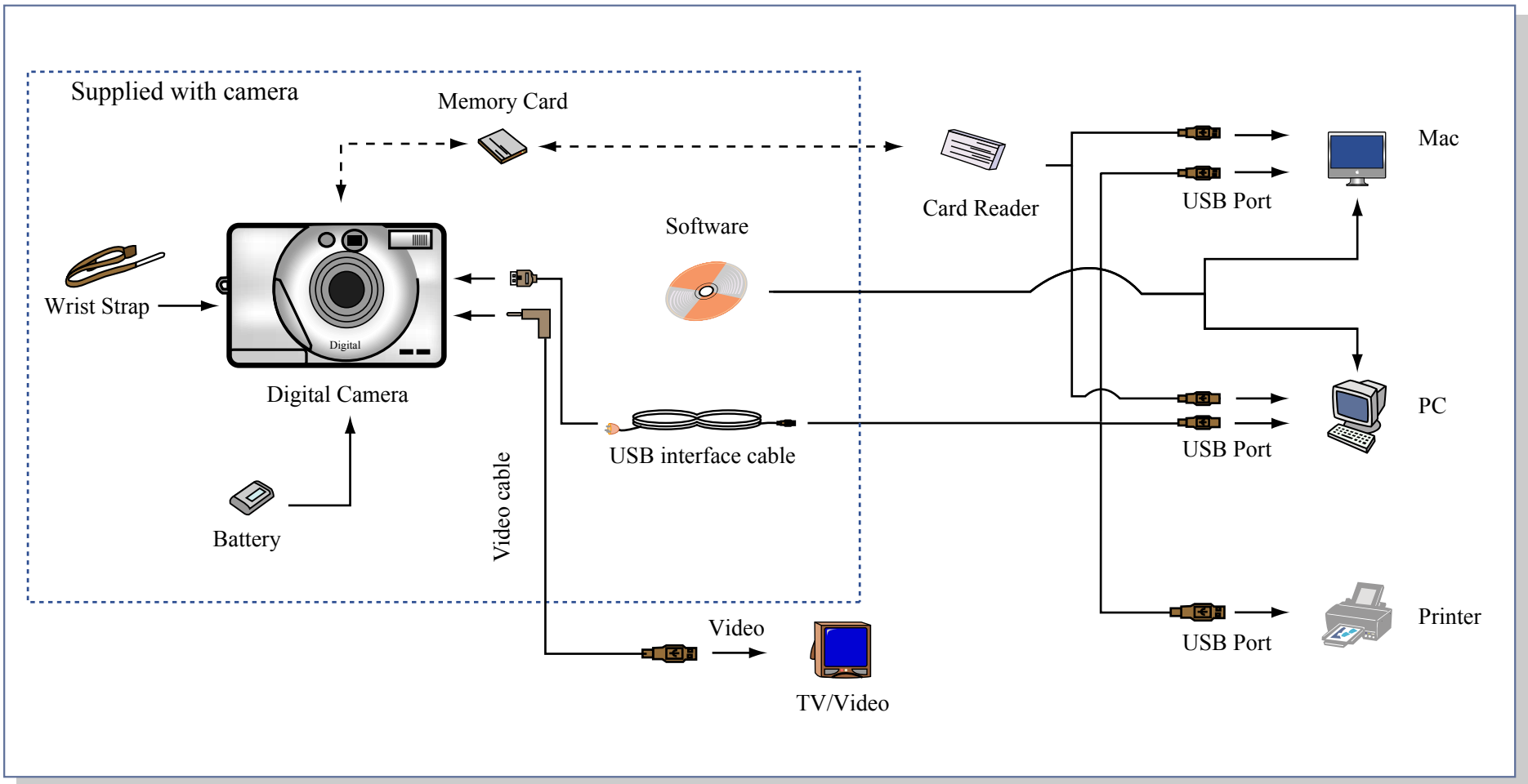
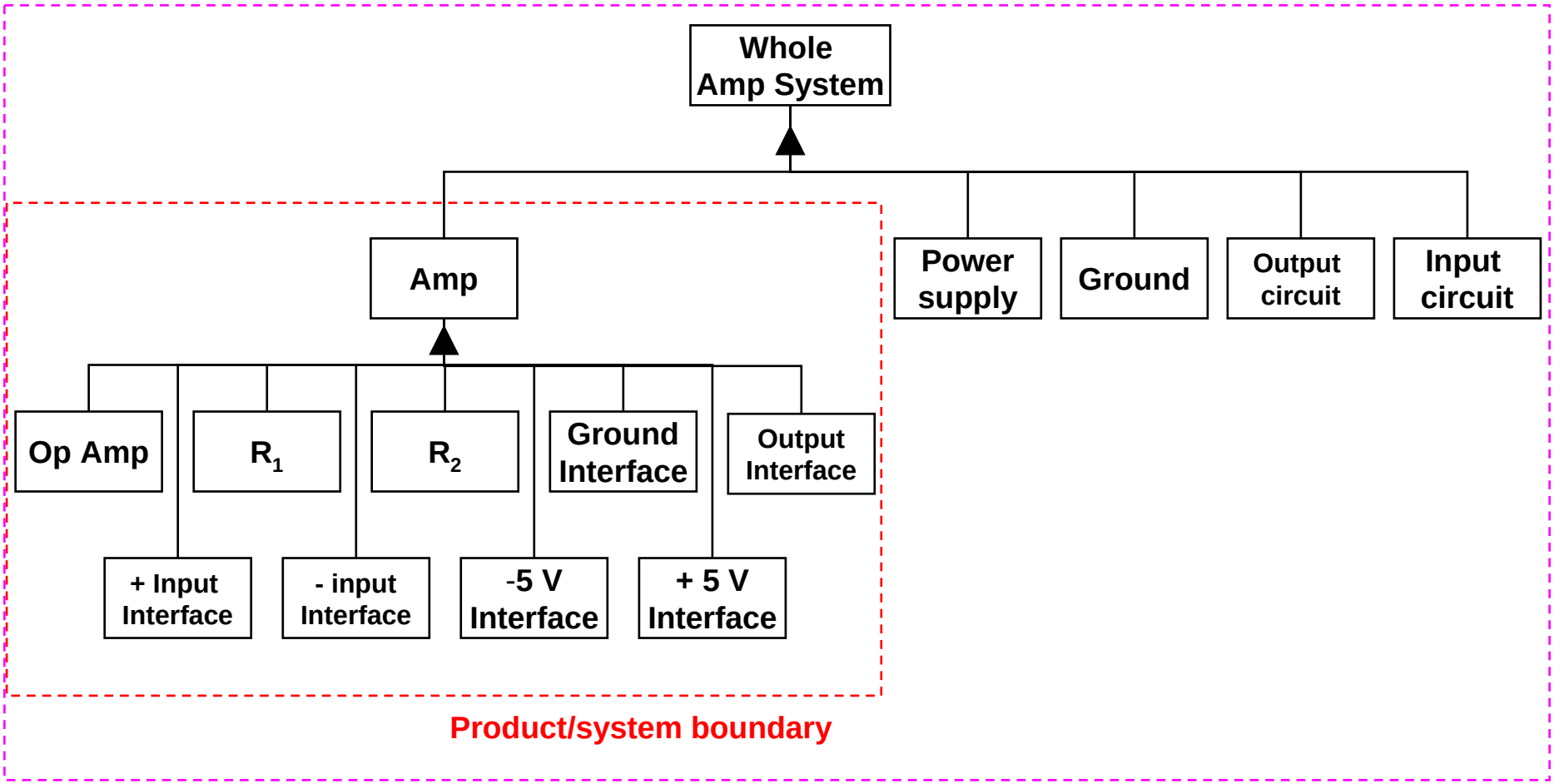


Figure by MIT OCW.

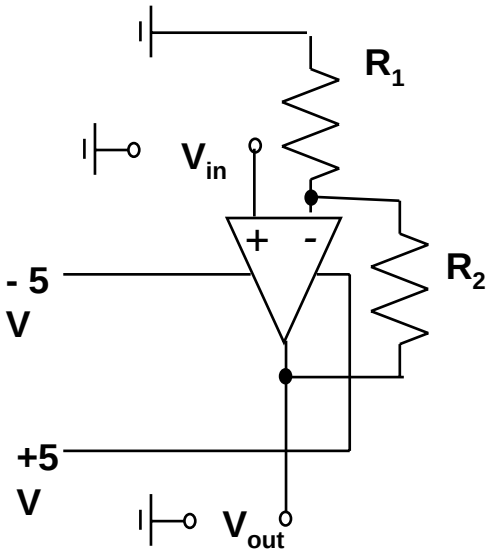
- What is the whole product system
- What is the use context?
- What are the boundaries?
- What are the interfaces?

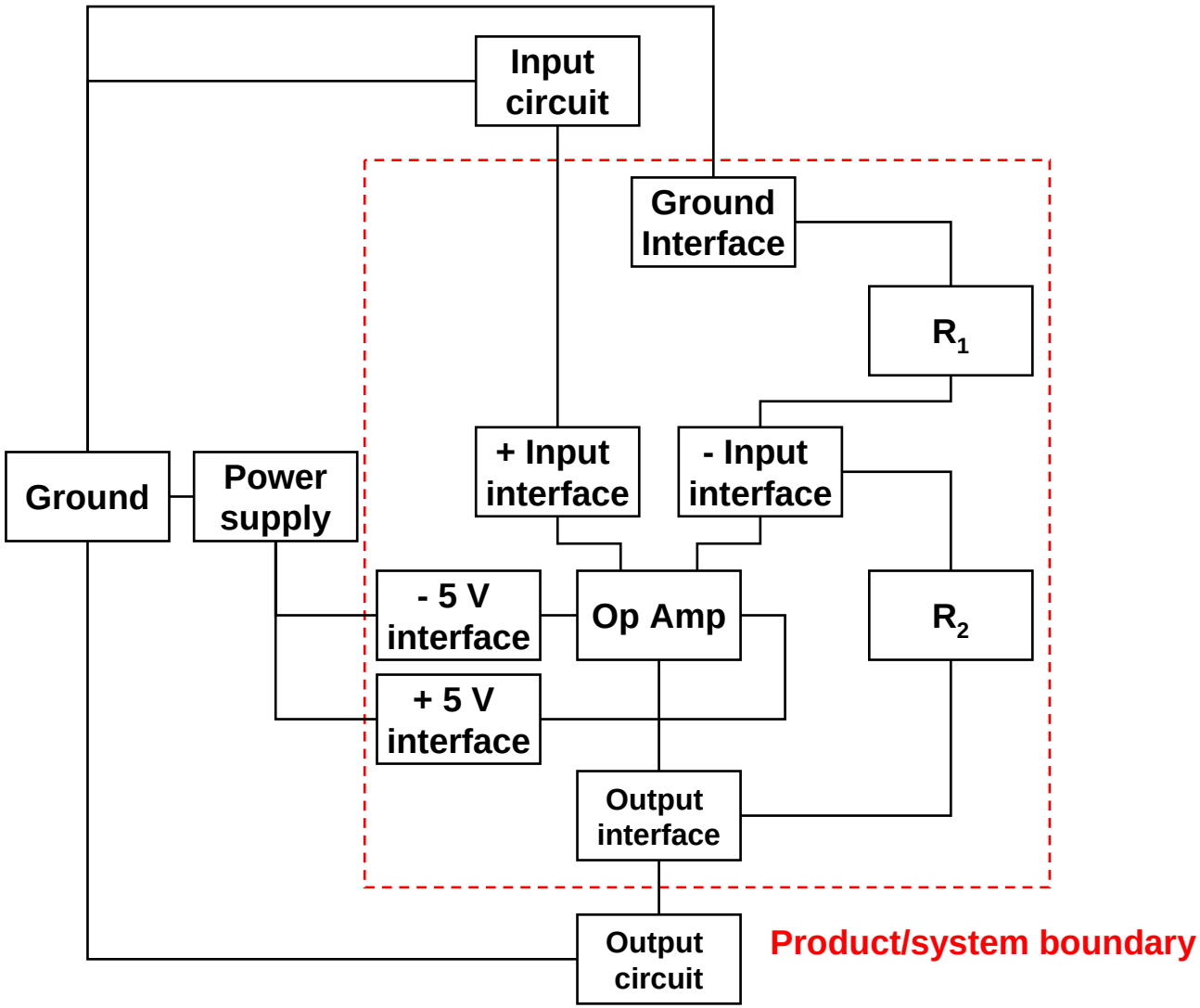
Op Amp - Whole Product System



- There will be an interface to other elements of the whole product system
- Therefore knowledge of these elements informs interface control

Amp - Whole Product and Boundaries





- What is the whole product system
- What is the use context?
- What are the boundaries?
- What are the interfaces?

Amp - Whole Product System Boundary and Interfaces

- In the matrix representation, the boundary is between rows and columns
- Interfaces are clearly identified in the block off diagonals

lower diagonal spacial/topological, upper implementation

	resitor 1	resistor 2	op amp	+input interface	-input interface	output interface	ground interface	-5 V interface	+5 V interface	input circuit	output circuit	power supply	ground
resitor 1	x				e		e						
resistor 2		x			e	e							
op amp			x	ei	ei	ei		ei	ei				
+input interface			t	x						e			
-input interface	t	t	t		x								
output interface		t	t			x					e		
ground interface	t						x						e
-5 V interface			t					x				e	
+5 V interface			t						x			e	
input circuit				t						x			e
output circuit						t					x		e
power supply								t	t			x	e
ground							t			t	t	t	x

Implementation Interfaces

Spatial Interfaces

t = touching, tangent
b = boundary
w = within, s = surrounding
ov = overlapping
da = at a specified distance or angular allignment

m = mechanical connected (pressing, bolted, bonded, etc.)
e = electrical connected (soldered, etc.)
c = compilation with

no second symbol implies connector of some sort
i as second symbol implies integral

Software - Boundaries and Interfaces

Interface?

```
temporary = array [ j+1 ]
array[ j+1 ] = array [ j ]
array[ j ] = temporary
```

Interface? Product/system boundary

```
Procedure exchange_contents(List array,
number j)
        temporary = array [ j+1 ]
        array[ j+1 ] = array [ j ]
        array[ j ] = temporary
return array
```

Product/system boundary

- **The act of trying to draw a boundary will often reveal a poorly controlled or ambiguous interface**

Whole Product System - Code Bubblesort

```
Procedure bubblesort (List array, number length_of_array)
    for i=1 to length_of_array
        for j=1 to length_of_array - i
            if array[ j ] > array [ j+1 ] then
                exchange_contents (array, j)
            end if
        end of j loop
    end of i loop
    return array
End of procedure
```

```
Procedure exchange_contents(List array,
number j)
        temporary = array [ j+1 ]
        array[ j+1 ] = array [ j ]
        array[ j ] = temporary
return array
```

Product/system boundary

- What is the whole product system
- What is the use context?
- What are the boundaries?
- What are the interfaces?

Bubblesort - Whole Product Boundary and Interfaces

- **The structural interfaces in software determine sequence and nesting, and compilation**

- **Example assumes that procedure exchange_contents is separately compiled and called**

lower diagonal spacial/topological
upper implementation

	proc state	l loop inst	j loop inst	if inst	temp inst	a(j+1) inst	a(j) inst	return state
proc state	x	c	c	c				c
l loop inst	f	x	c	c				c
j loop inst		fw	x	c				c
if inst			fw	x				c
temp inst				fw	x	c	c	
a(j+1) inst				w	f	x	c	
a(j) inst				w		f	x	
return state							f	x

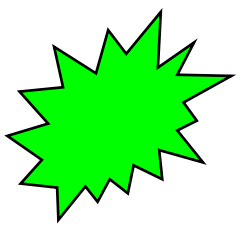
t = touching, tangent
b = boundary
w = within, s = surrounding
ov = overlapping
da = at a specified distance or angular allignment
f = follows in sequence, l = leads in sequence

m = mechanical connected (pressing, bolted, bonded, etc.)
e = electrical connected (soldered, etc.)
c = compilation with

no second symbol implies connector of some sort
i as second symbol implies integral

Static Graphical User Interface - Whole Product?

- What is the whole product system
- What is the use context?
- What are the boundaries?
- What are the interfaces?



Courtesy of ENPHO. Used with permission.

Form of Whole Product - Questions

- **What is the product system? Its elements? Its formal structure?**
- **What are the supporting systems?**
- **What is the whole product system?**
- **What is the use context?**
- **What are the boundaries?**
- **What are the interfaces?**

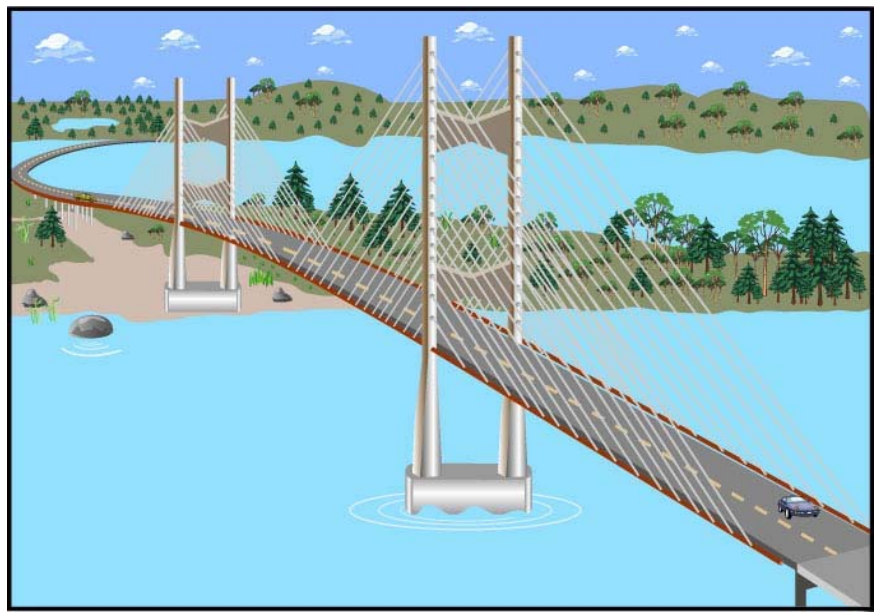


Figure by MIT OCW.

Characterization

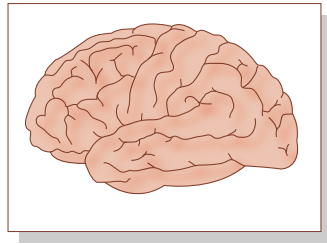
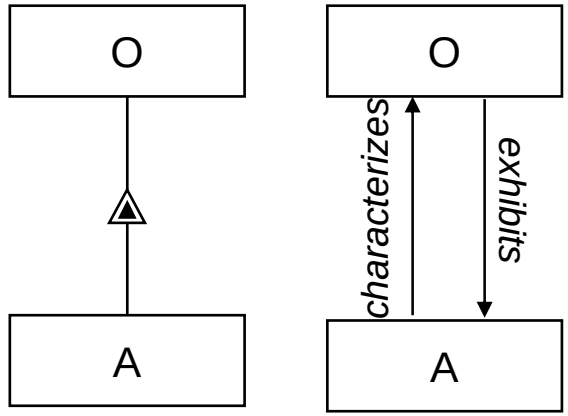
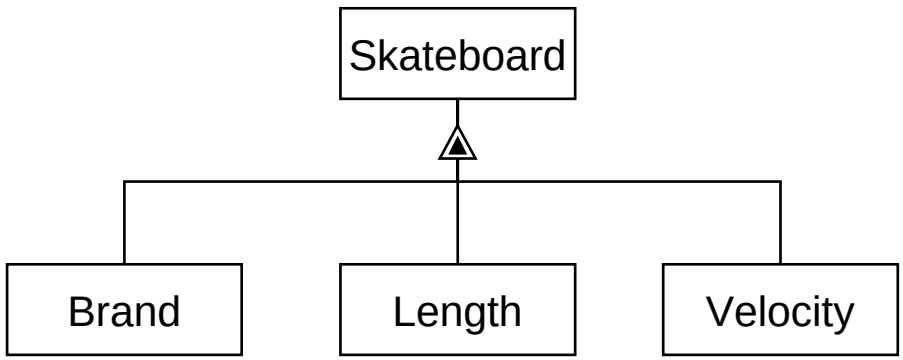


Figure by MIT OCW.

- **Characterization/Exhibition**
  - **The relation between an object and its features or *attributes***
  - **Some attributes are *states***





▲ Decomposes to

△ Has attribute of

State

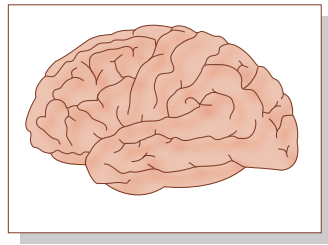
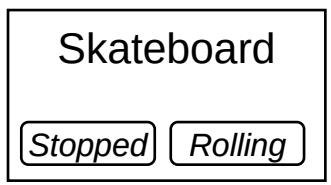
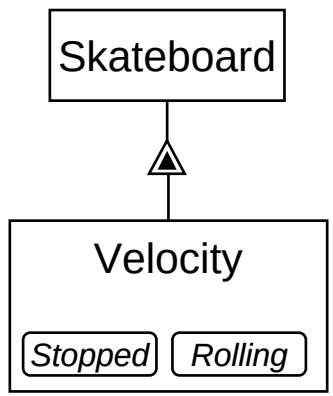


Figure by MIT OCW.

- **Defined: State is a situation in which the object can exist for some positive duration of time (and implicitly can or will change)**
- **The combination of all the states describes the possible configuration of the system throughout the operational time.**
- **The states can be shown with the object, or alternatively within an attribute object.**



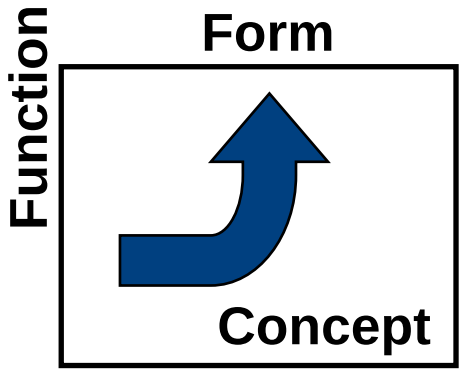


Summary - Form

- **The physical/informational embodiment which exists, or has the potential to exist**
- **Objects + Structure (of form)**
- **Is a system attribute, created by the architect**
- **Product/system form combines with other supporting systems (with which it interfaces at the boundary) to form the whole product system that creates value**

Architecture

- **Consists of:**
  - **Function**
  - **Related by Concept**
  - **To Form**



Function Topics

- Function
- Emergence
- Process + Operand
- External function
- Internal function

Function - Defined

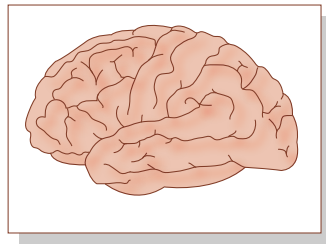


Figure by MIT OCW.

- **The activities, operations and transformations that cause, create or contribute to performance (i.e. meeting goals)**
- **The actions for which a thing exists or is employed**
- **Is a product/system attribute**

Function - Described

- **Is conceived by the architect**
- **Must be conceived so that goals can be achieved**
- **Is what the system eventually does, the activities and transformations which emerge as sub-function aggregate**
- **Should initially be stated in solution neutral language**

Function - Other views:

- **Function is what the system does**

  Form is what the system is [Otto]

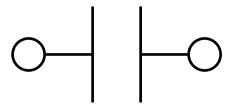
- **Function is the relationship between inputs and outputs, independent of form [Otto]**

- **Process is a transformation (creation or change) applied to one or more objects in the system [Dori]**

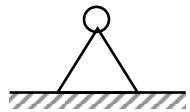
Function is Associate with Form

- **Change voltage proportional to current**

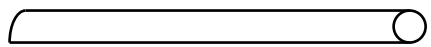
- **Change voltage proportional to charge**



- **React translation forces**



- **Carry moment and shear**



- **Conditionally transfer control**

if array[j] > array [j+1] then

- **Assign a value**

temporary = array [j+1]

Emergence

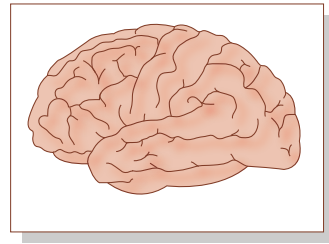
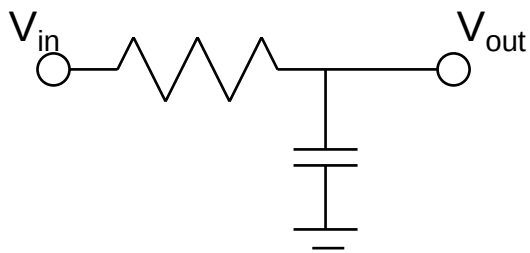


Figure by MIT OCW.

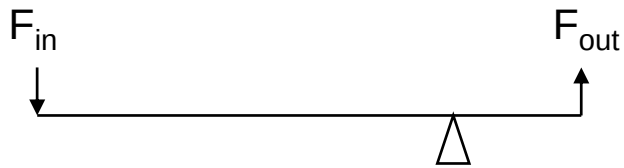
- As elements of form are brought together, new function emerges
- Unlike form, which simply aggregates, functions do not combine in any “linear” way - it is often difficult or impossible, *a priori*, to predict the emergent function
- In design, the expected function may appear, may fail to appear, or an unintended function may appear
- It is exactly this property of emergence that gives systems their “power”- definition of ‘system’:
  - A set of interrelated elements that perform a function, whose functionality is greater than the sum of the parts

Function Emerges as Form Assembles

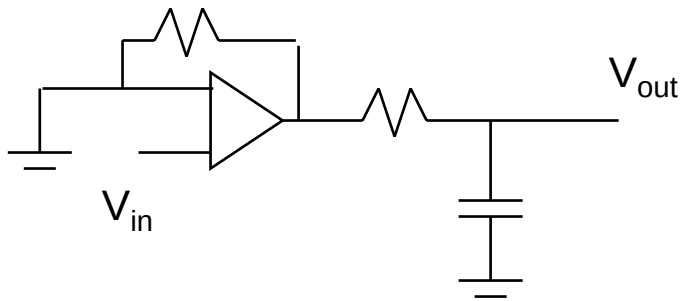
- **Attenuate high frequency**



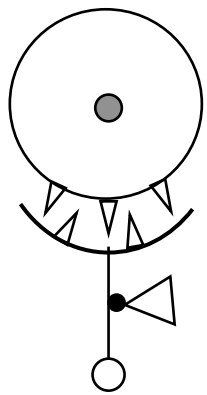
- **Increase force**



- **Amplify low frequency signal**

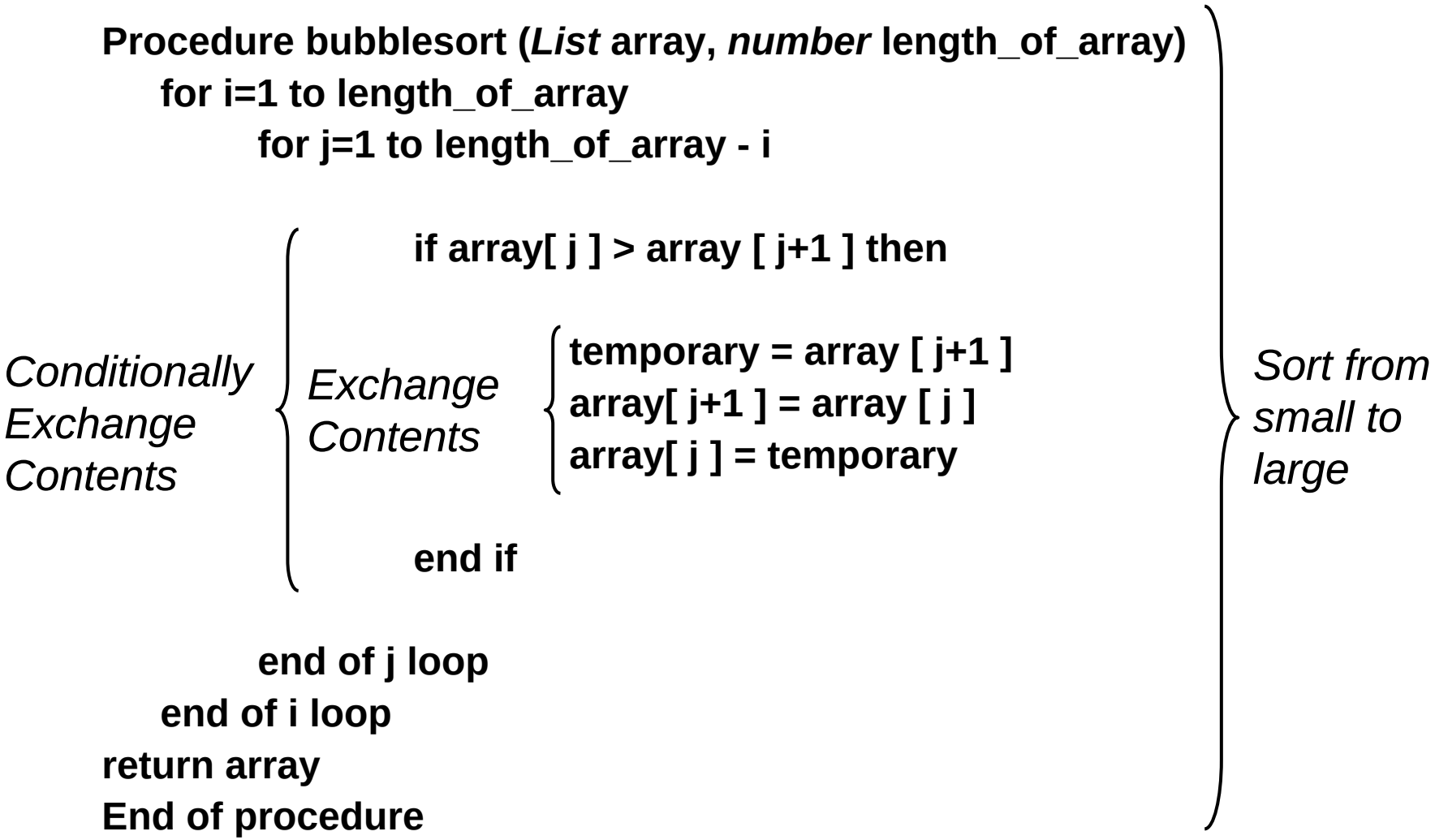


- **Regulate shaft speed**

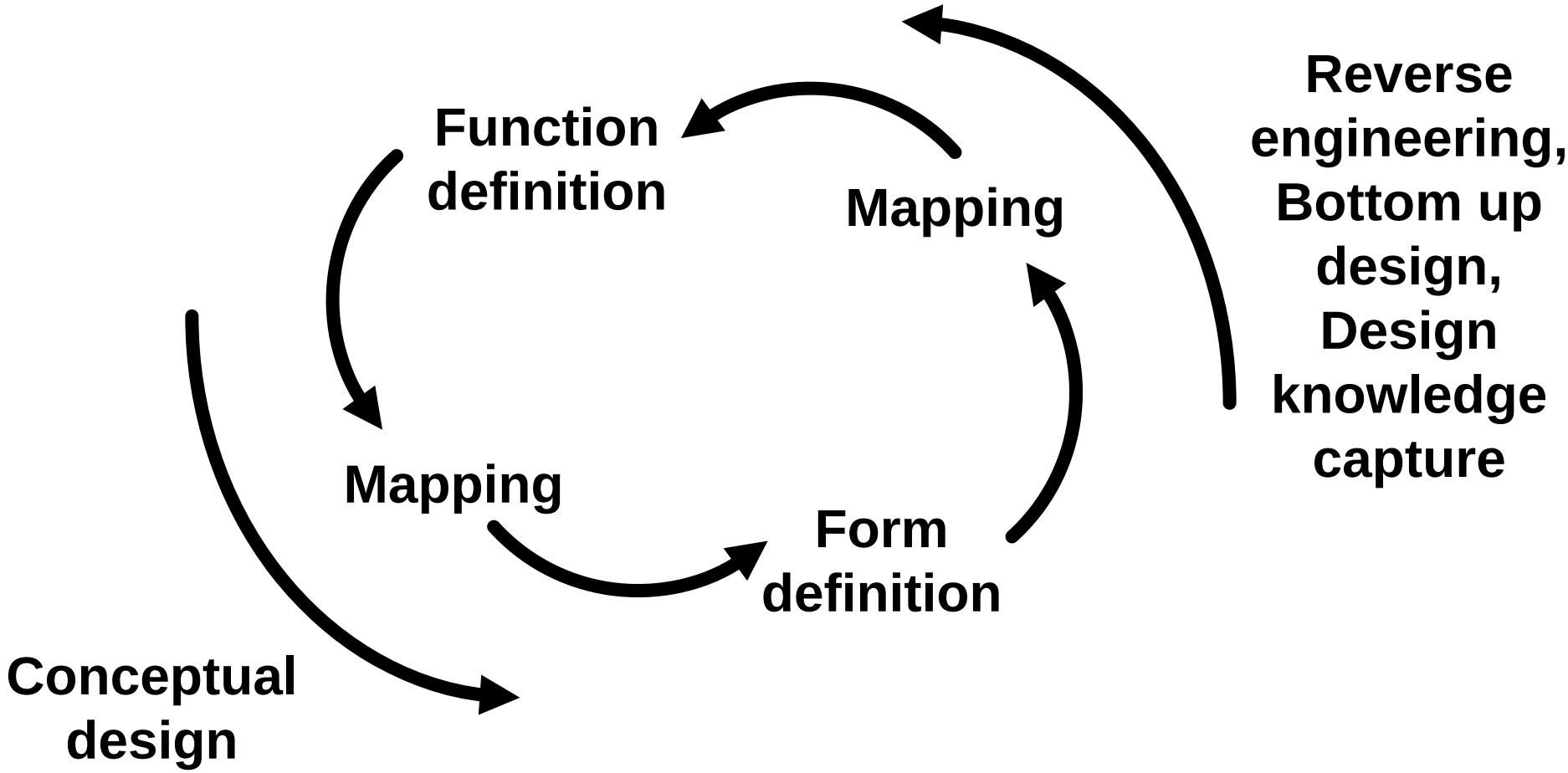


Emergent function depends on elements *and* structure

Function Emerges as Form Assembles



Form - Function Sequence

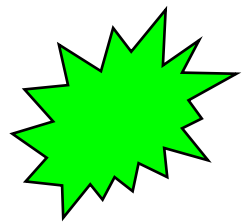


Design vs. Reverse Engineering

- **In (conventional, top down) design, you know the functions (and presumably the goals) and try to create the form to deliver the function**
- **In reverse engineering, you know the form, and are trying to infer the function (and presumably eventually the goals)**
- **In bottom up design, you build elements and discover what emerges as they come together**
- **In design knowledge capture, you record the function along with the design of form**

Exercise: Reverse Engineering of Function

- **This is an exercise to demonstrate reverse engineering of function of objects with which you are not personally familiar**
- **What is the function of:**
  - **A notebook?**
  - **A pen?**
  - **A glass?**
  - **A set of instructions?**
  - **An object I give you?**



- **Examine the object, and describe its form**
- **Try to discern and describe its function**

Function and form are completely separate attributes.
Even when form is evident, function is not easily inferred without a working knowledge of operation, operand and whole product system.

Function = Process + Operand

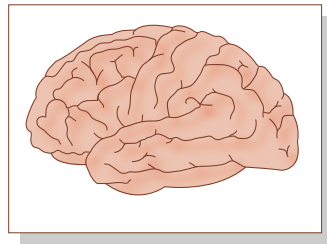


Figure by MIT OCW.

- Note that function consists of the action or transformation + the operand, an object which is acted on, or transformed:
  - Change voltage
  - React force
  - Amplify signal
  - Regulate shaft speed
  - Contain liquid
  - Lay down ink
  - Stop an operator
  - Count circles

- The action is called a process
- The object that is acted upon is called the operand

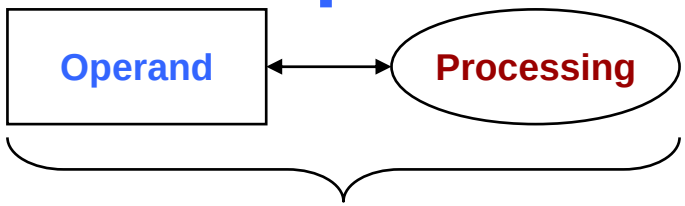
Processes

- **Defined: A process is the pattern of transformation applied to one or more objects**
- **Cannot hold or touch a process - it is fleeting**
- **Generally creation, change, or destruction**
- **A process relies on at least one object in the pre-process set**
- **A process transforms at least one object in the pre-process set**
- **A process takes place along a time line**
- **A process is associated with a verb**

About Processes

- **Processes are pure activity or transformation:**
  - **Transporting**
  - **Transporting**
  - **Powering**
  - **Supporting**
  - **Sorting**
- **Time dependent, they are there and gone**
- **What a system does, vs. what it is (the form)**
- **Imagine a process that has been important to you - can you recreate it?**
- **Try to draw a picture of a process**

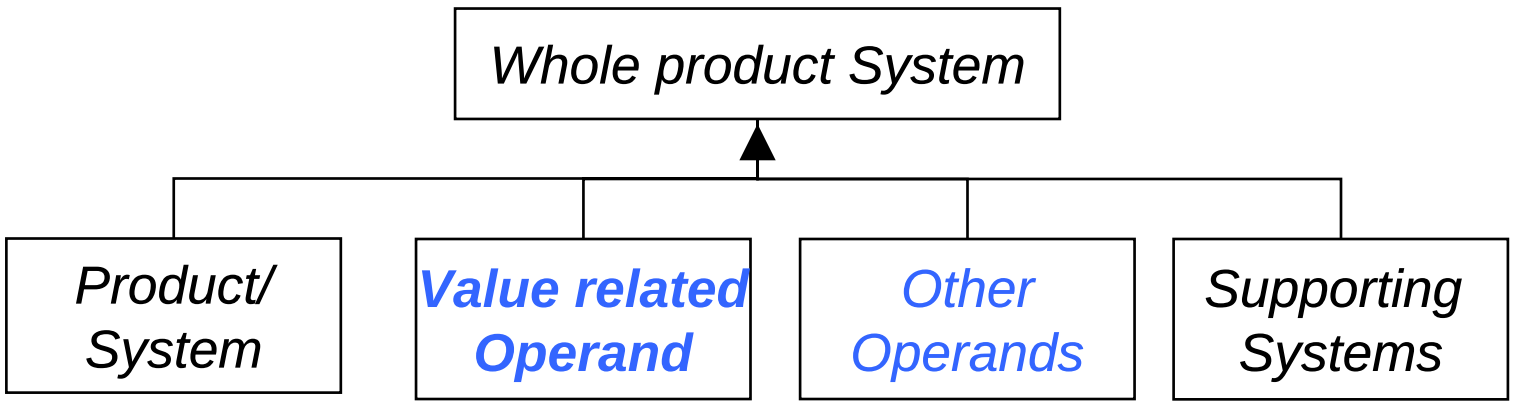
The Operand



- **Operand is the object which is acted upon by the process, which may be created, destroyed, or altered:**
  - **Image is captured**
  - **Signal is amplified**
  - **Array is sorted**
- **You often do not supply the operand, and there may be more than one operand to consider**
- **The double arrow is the generic link, called "effecting"**
- **A single headed arrow can be used to represent "producing" or "consuming"**

Value Related Operand

- **The product/system always *operates on* one or several operands**
- **It is the change in an attribute of one of an operands that is associated with the delivered value of the product system**
- **Because of its importance to value, it is useful to specially designate this value related operand**



610

Camera - Value Related Operand?

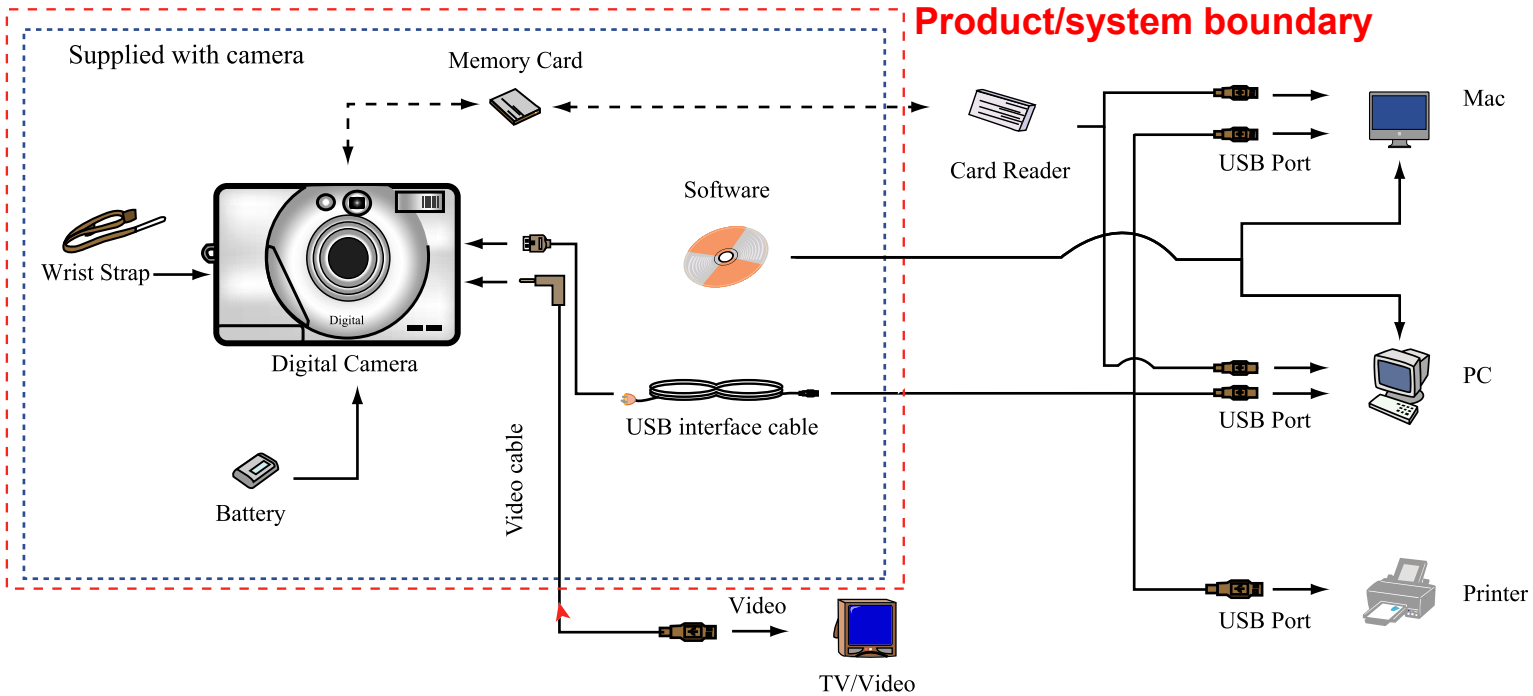


Figure by MIT OCW.

What's the value related operand?

External Function Produces Benefit

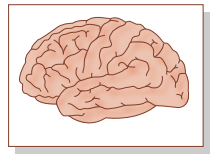


Figure by MIT OCW.

- **The system function which emerges at the highest level is the externally delivered function = process + value related operand**
  - **Amplify low frequency signal**
  - **Regulate shaft speed**
  - **Sort Array**
  - **Ensure safety of passenger**
  - **Provide shelter to homeowner**
- **It is the external function that delivers benefit, and therefore value, of the product system**
- **The form generally is an instrument of delivering benefit, but is not unique to the benefit delivery**
  - **If a competitor delivers the same function with another object which is superior, they will displace you**

Externally Delivered Function

- External function is delivered across a boundary - the value related operand is external to the product/system
- External function is linked to the delivery of benefit

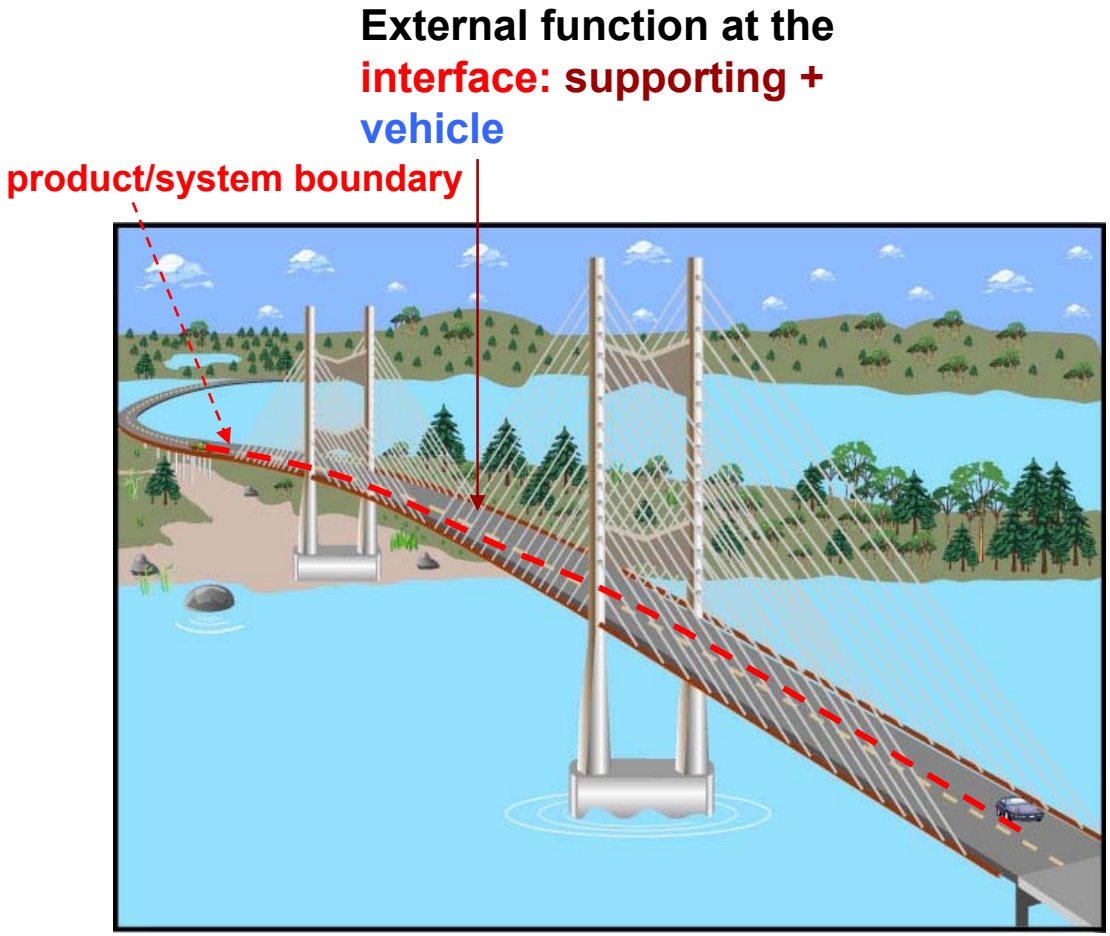


Figure by MIT OCW.

- What is the value related operand?
- What are the value related states?
- What is the externally delivered function?

610

Externally Delivered Function

External function at the interface: produce + digital image

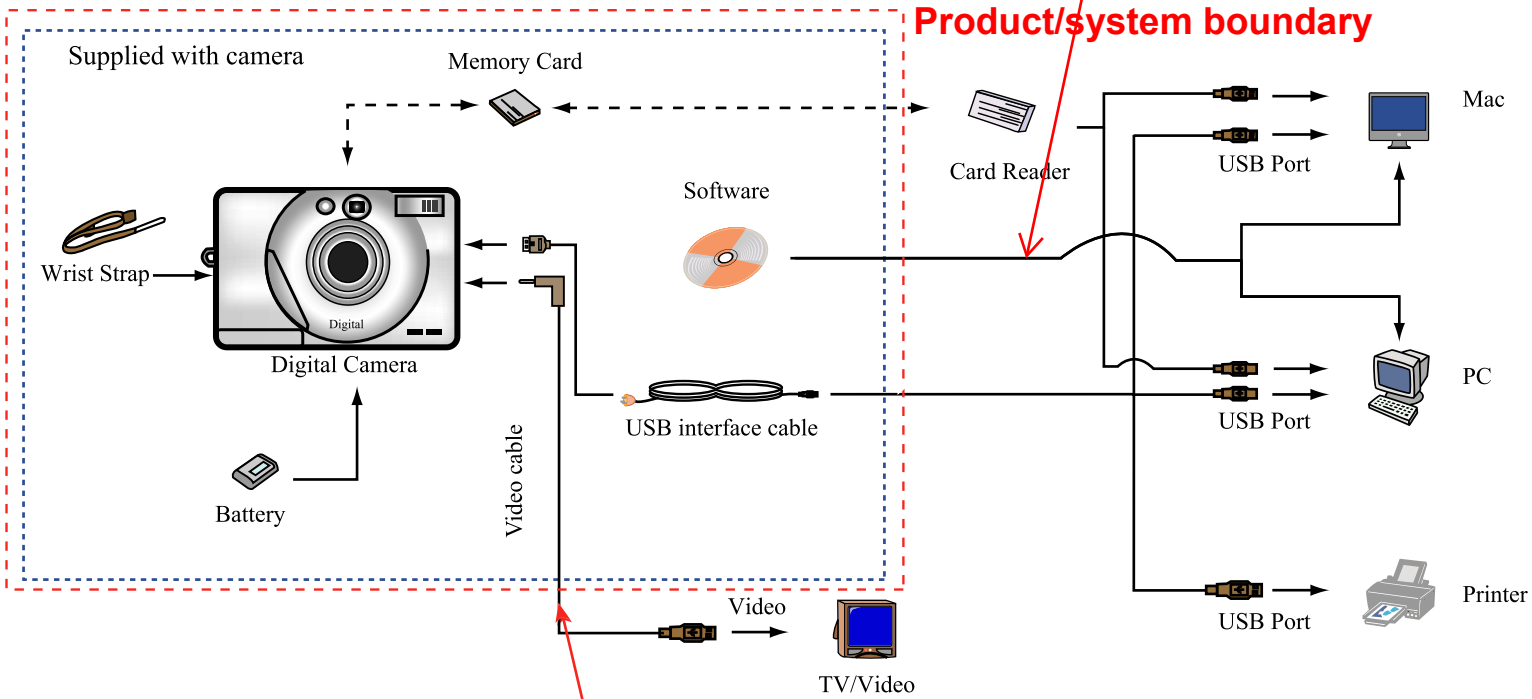
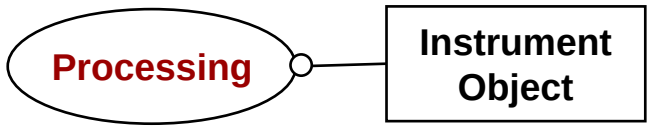


Figure by MIT OCW.

External function at the interface: produce + analog image

610

Form Enables Process



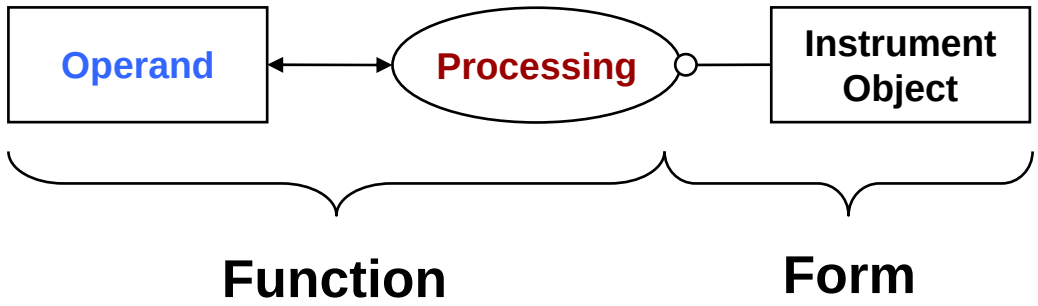
- **Enablers of a process is an object that must be present for that process to occur, but does not change as a result of the occurrence of the process**
- **Examples:**
  - **Capturing by a digital camera**
  - **Amplifying with an operational amplifier**
  - **Sorting with bubblesort routine**
- **The "arrow" with the circular end represents and enabler, which can be an instrument (non-human) or an agent (human)**

Three Core Ideas

- **Operand - an element of form external to the product system, whose change is related to benefit and therefore value**
- **Process - the transformation or change in the operand**
- **Form - the instrument that executes the transformation (and attracts cost!) and consists of elements plus structure**

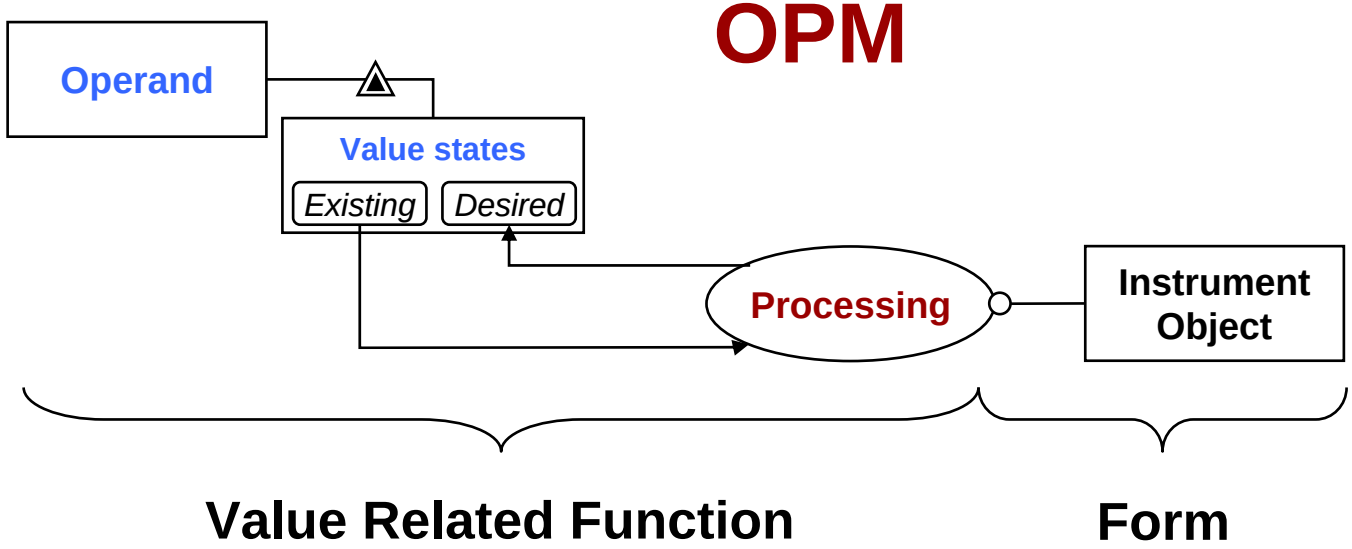
- **Wouldn't it be nice if we had a semantically exact way of representing these 3 core ideas**
- **How can we do with conventional representations?**

Semantically Exact Representation with OPM



- **Architecture is made up of operands + processes (functions) plus instrument object (form)**
- **Examples:**
  - **Image is captured by digital camera**
  - **Low frequency signal is amplified with an operational amplifier**
  - **Tone is created by whistle**
  - **Array is sorted by bublesort routine**

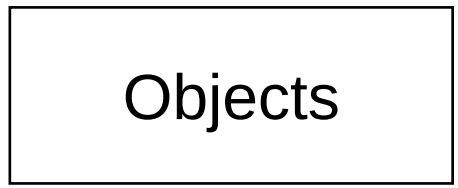
Value Related State Representation with OPM

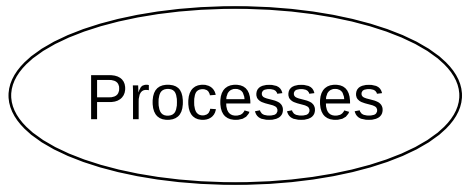


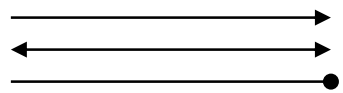
- **Architecture delivers value when the externally delivered operand changes its state through the action of the processes enabled by instrument object (form)**
- **Examples:**
  - **Image (none to existing) is captured by digital camera**
  - **Low frequency signal (low amplitude to higher amplitude) is amplified with an operational amplifier**
  - **Tone (none to existing) is created by whistle**
  - **Array (unsorted to sorted) is sorted by bubblesort routine**

A Tool - Object Process Modeling

- **Object: that which has the potential of stable, unconditional existence for some positive duration of time.**
- **Form is the sum of objects and their structure**
- **Process: the pattern of transformation applied to one or more objects. Processes change an object or its attribute.**
- **Function emerges from processes acting on operands**
- **All links between objects and processes have precise semantics**

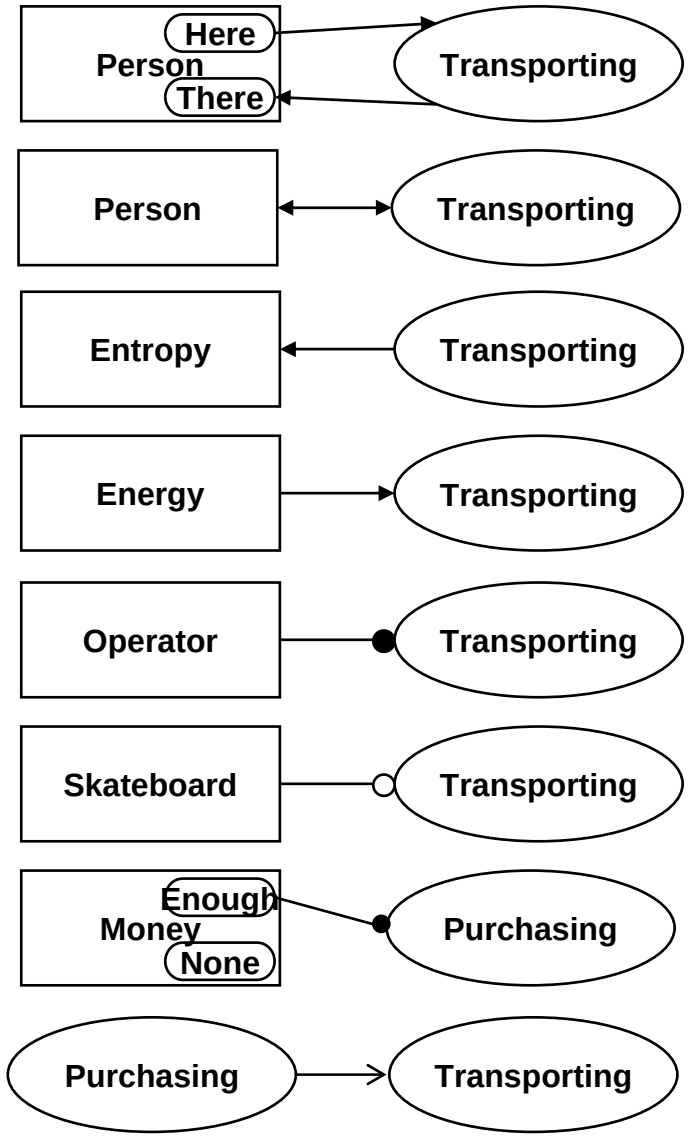






OPM Process Links

- **P changes O (from state A to B).**
- **P affects O**
- **P yields or creates O**
- **P consumes or destroys O**
- **O is an agent of P (agent)**
- **O is and instrument of P**
- **P occurs if O is in state A**
- **P1 invokes P2 directly**



Value

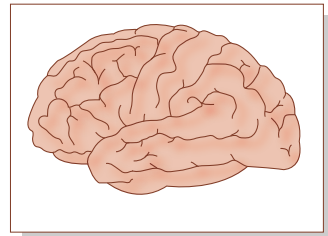


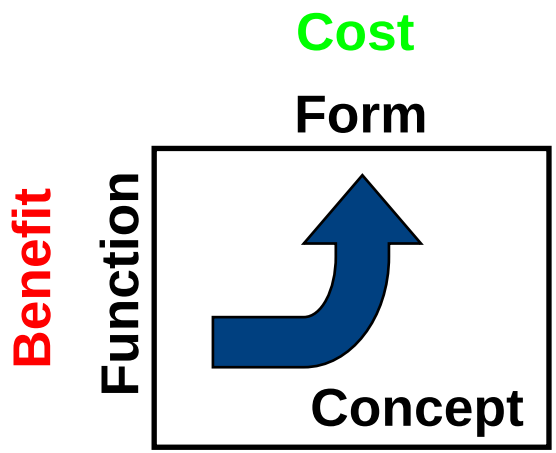
Figure by MIT OCW.

- **Value is benefit at cost**
  - **Simple metrics are benefit/cost, benefit - cost, etc.**
  - **Sometimes more complex metrics**
- **Benefit is worth, importance, utility as judged by a subjective observer (the beneficiary)**

- **Another view:**
  - **Value: "how various stakeholders find particular worth, utility, benefit, or reward in exchange for their respective contributions to the enterprise" [Murman, et al. LEV p178]**

Value and Architecture

- Value is benefit at cost
- Benefit is driven by function externally delivered across the interface
- Cost is driven by the design of the form - “parts attract cost”
- The relationship of function to form is therefore the relationship of benefit to cost



A good architecture delivers benefit at competitive cost

Value - Questions?

- **What is the value related operand?**
- **What are the value related states that change?**
- **What is the externally delivered function?**

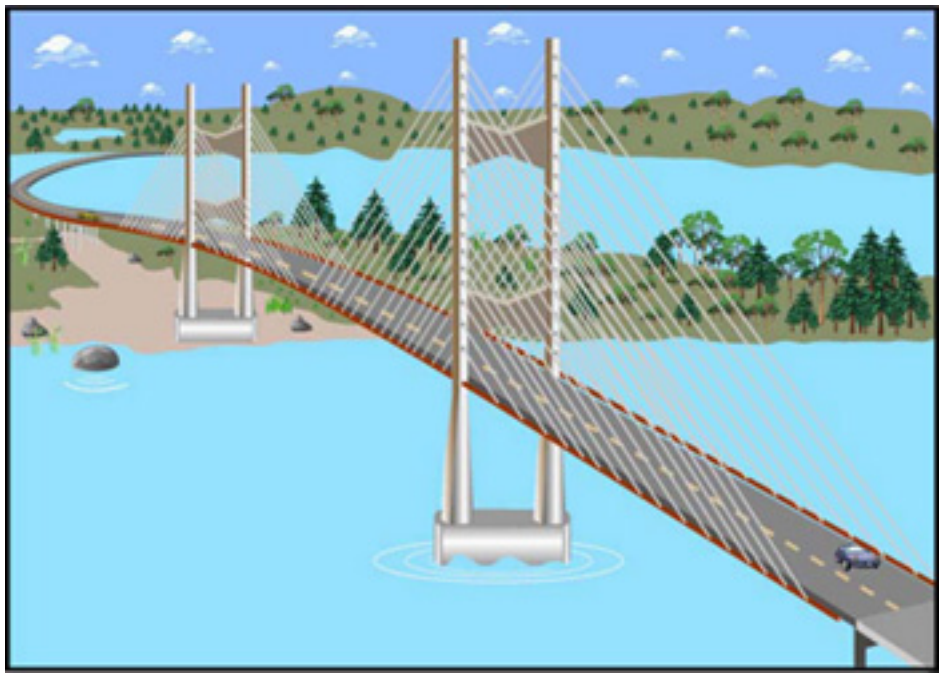


Figure by MIT OCW.

Externally Delivered Function Emerges from Internal Function

- **System function which emerges at the highest level is the externally delivered function which usually acts on the operand**
  - **Example: amplify low frequency signal**
- **Systems have internal functions, which combine to produce the emergent externally delivered function**
  - **Example: amplify …, change voltage ...**

Externally Delivered and Internal Function

- **External function is delivered across a boundary - the operand is external to the product/system**
- **External function emerges from internal function, which is executed by form**

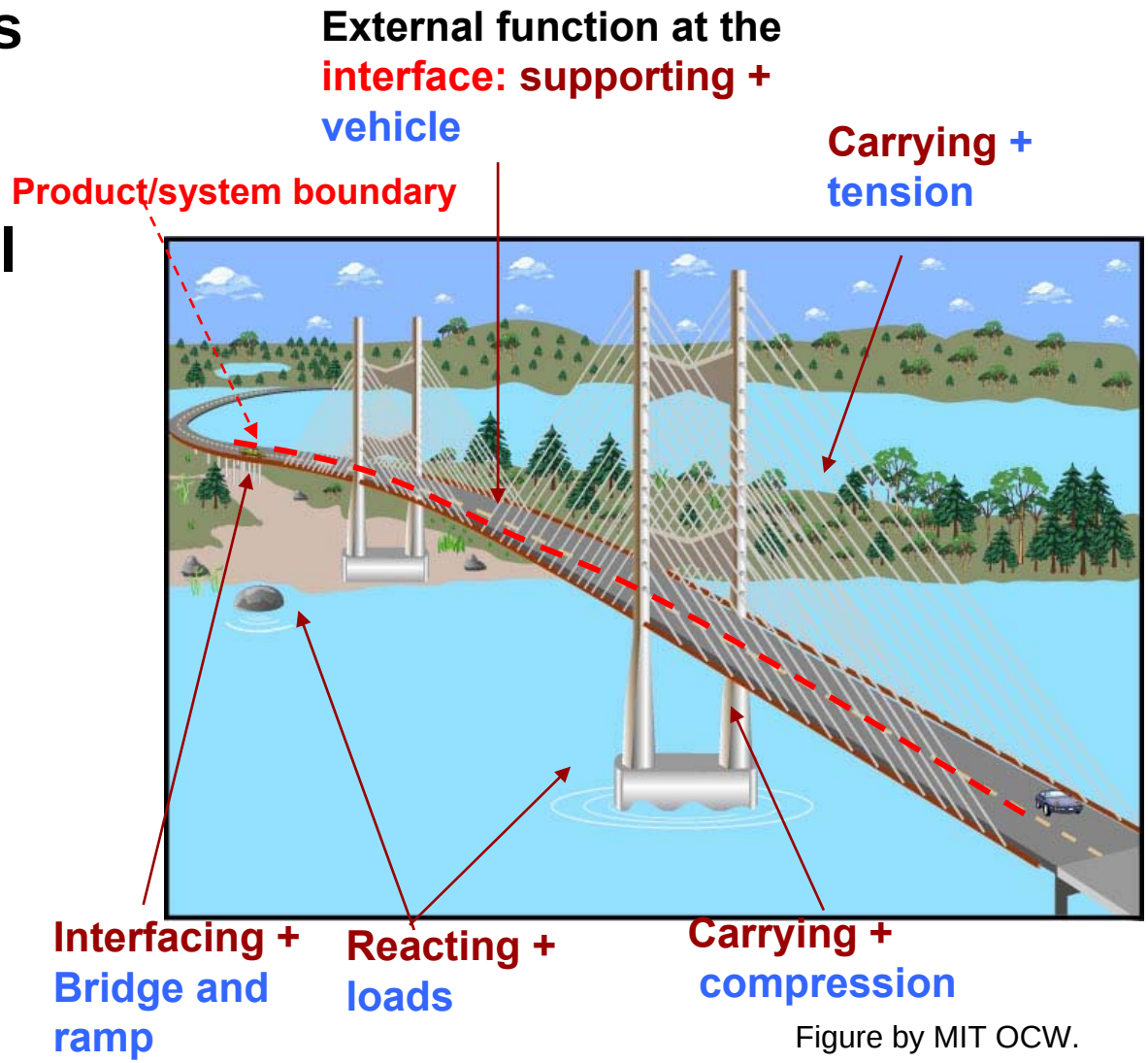


Figure by MIT OCW.

610

Externally Delivered and Internal Function

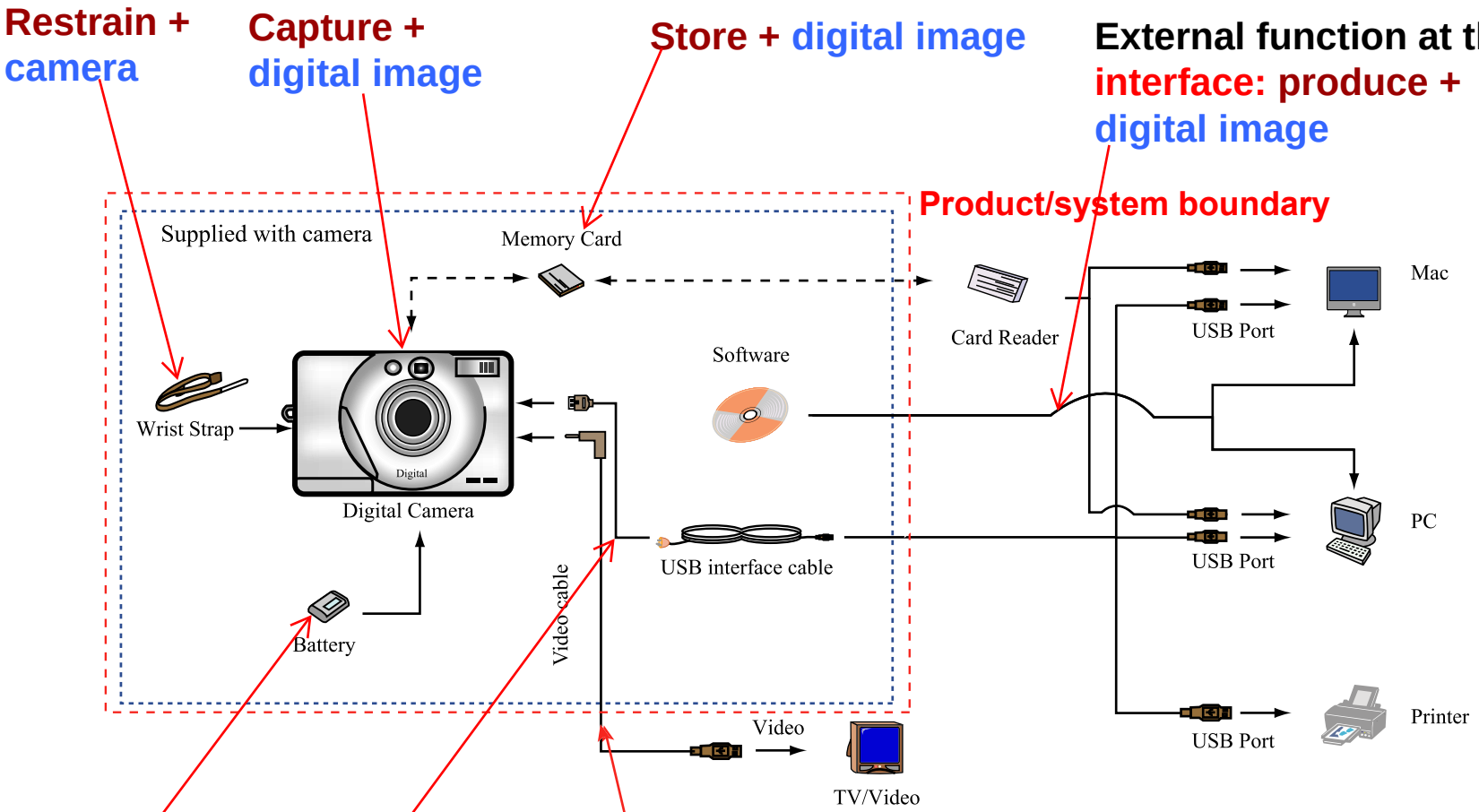


Figure by MIT OCW.

610

Externally Delivered and Internal Function

- External function is delivered across a boundary - the operand is external to the product/system
- External function emerges from internal function, which is executed by form

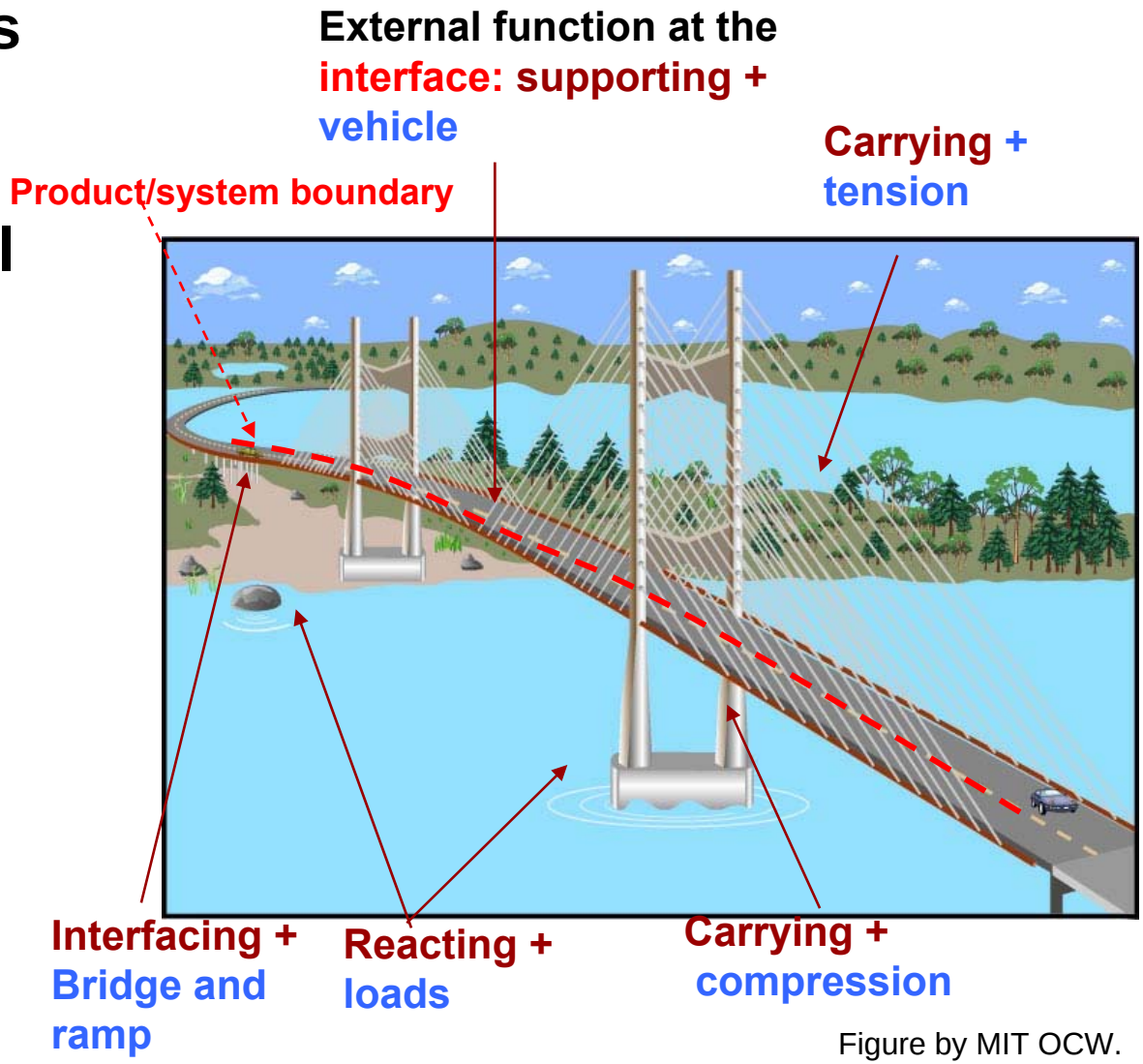
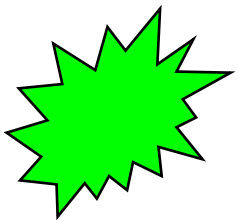


Figure by MIT OCW.

610

Delivered and Internal Function - Whistle



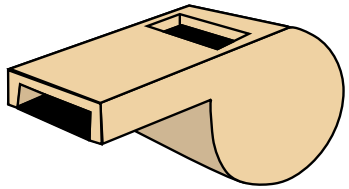
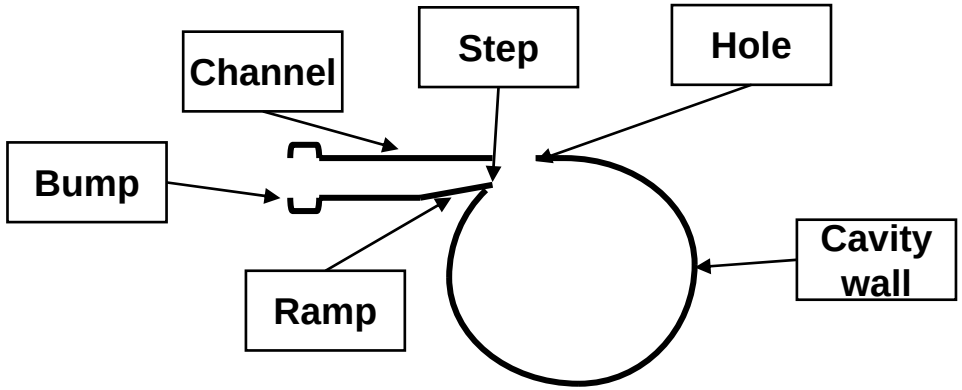
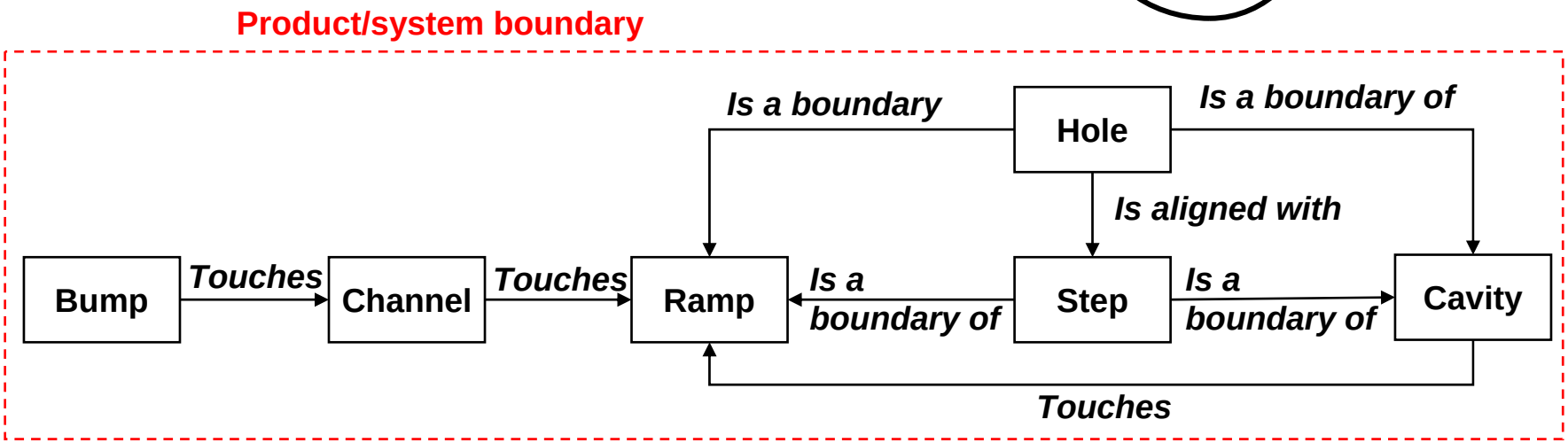


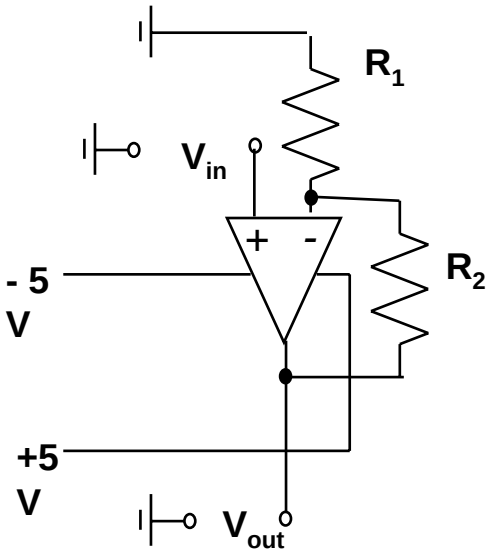
Figure by MIT OCW.

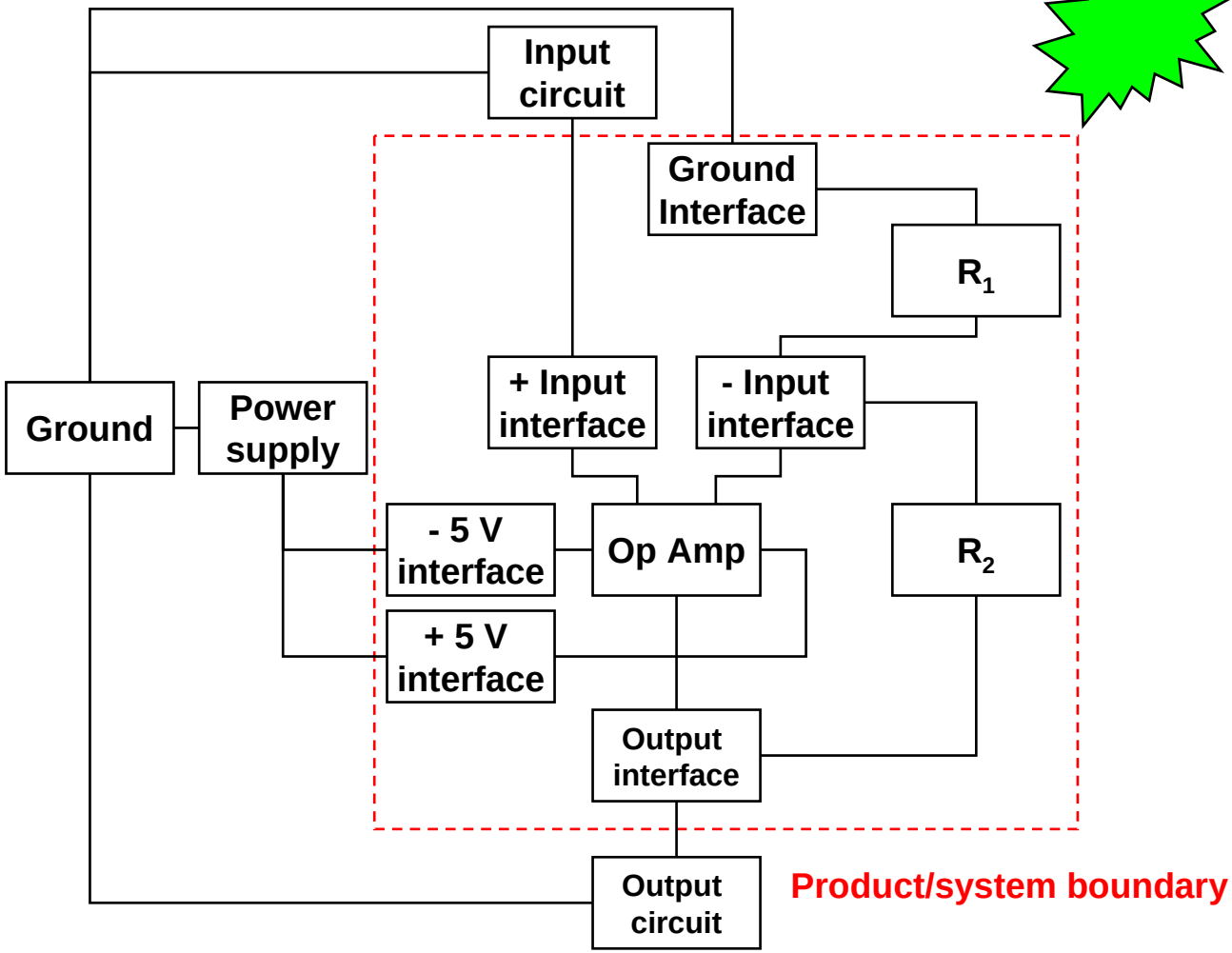




- What is the value related operand and states?
- What is the externally delivered function?
- What are the internal functions?
- How are they mapped to form?

Delivered and Internal Function - Amp

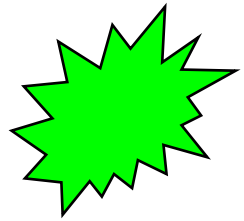




- What is the value related operand and states?
- What is the externally delivered function?
- What are the internal functions?
- How are they mapped to form?

Delivered and Internal Function - Bubblesort

```
Procedure bubblesort (List array, number length_of_array)
    for i=1 to length_of_array;
        for j=1 to length_of_array - i;
            if array[ j ] > array [ j+1 ] then
                exchange_content (array[ j ], array [ j+1 ])
            end if
        end of j loop
    end of i loop
return array
End of procedure
```



```
Procedure exchange_contents(List array,
number i, number j)
        temporary = array [ j+1 ]
        array[ j+1 ] = array [ j ]
        array[ j ] = temporary
return array
```

Product/system boundary

- What is the value related operand and states?
- What is the externally delivered function?
- What are the internal functions?
- How are they mapped to form?

Summary - Function

- **Function is the activity, operations, transformations that create or contribute to performance - it is operand + process**
- **Function is enabled by form, and emerges as form is assembled**
- **Externally function delivered to the operand is linked to the benefit of a product/system**
- **Function is a system attribute, conceived by the architect**

Reference PDP

- **PDP of Ulrich and Eppinger gives an “in the box” reference**
- **Starting point for Identify/Develop/Compare/Synthesize process**
- **Usable *method* for small groups (what are the principles?)**

¿Reference PDP?

- **Why would you have a formalized PDP?**
- **What are the essential features of the Ulrich and Eppinger model?**
- **What are the key limiting assumptions?**
- **How does this compare with the PDP in your enterprise?**

Goals of a Structured PDP

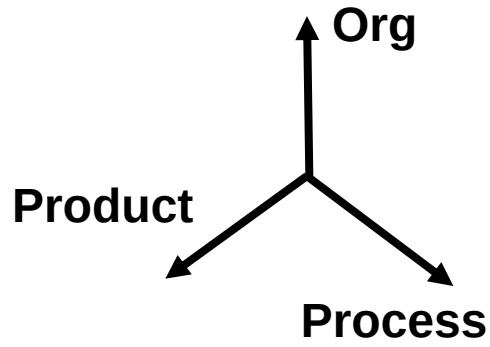
- Customer-focused products
- Competitive product designs
- Team coordination
- Reduce time to introduction
- Reduce cost of the design
- Facilitate group consensus
- Explicit decision process
- Create archival record
- Customizable methods

Ref: Ulrich and Eppinger

Key Assumptions of U&E PDP

- In the “small”
- Technology in hand
- Not a corporate stretch
- Relatively simple (probably physical) form
- Build/Bust developmental approach
- Focused on “the PDP proper”, not upstream + life cycle

PDP
In the SMALL v. In the LARGE



Product:

SIMPLE — Countable interfaces; Parts identifiable

v.

COMPLEX — Very many interfaces; Parts abstracted

Process:

STATIC — Goals & resources constant

v.

EVOLVING — Goals & resources changing

Organization:

SMALL — Team at a table

v.

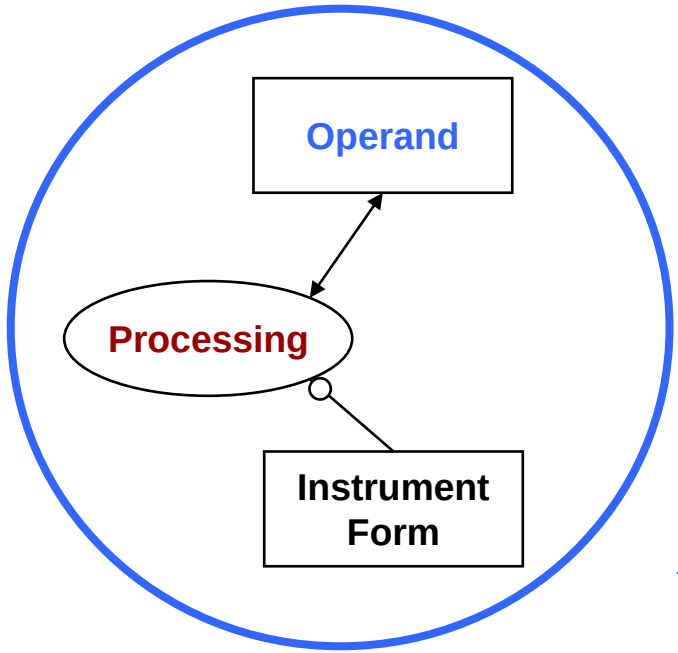
LARGE — Big team

Other Factors: clean slate vs. reuse of legacy components; single product vs. platform; collocated vs. distributed team; team in enterprise vs. supplier involvement

Summary - Reference PDP

- **A useful reference for in-the-small consumer focused mechanically based products**
- **A tool to reduce ambiguity by defining roles and processes**
- **Adaptable to a variety of contexts, staying within the key assumptions**

Summary to Date



Architecture?

Form = Elements + Structure
Function = Process + Operand
Form in the instrument of function