System Architecture

IAP Lecture 4

Ed Crawley

January 23, 2007

Rev 1.4

Today’s Topics

- Reflections on Architecture of Medium Systems
- Architecture, Example - Refrigerator
- Complexity
- Example - Skateboard
- Downstream processes - Operations, Operator, Design, Implementation
- Sequence and Timing
- Example - TCP

Concept to Architecture

- **Identify the operand, and value related attribute, and solution neutral transformation**
- **Identify the concept process and instrument object**
- **Identify one or two other concepts. Do they differ in process or instrument object?**
- **For the reference concept, identify other aspects of the whole product system and use context**
- **Identify aspects of multifunctional concepts, if applicable**
- **Informed by the concept form, identify the:**
  - **Idealized internal processes that touch directly on the delivery of value - the “value related internal processes”**
  - **The intermediate operands along that path**
  - **(perhaps) More realizable internal value related processes**

Concept to Architecture (2)

- Develop a detailed list of element, and then a rationalized list of level 2 elements (20-30)
- Show the formal structure (in this case absolutely complete, with every part enumerated, not the usual case!)

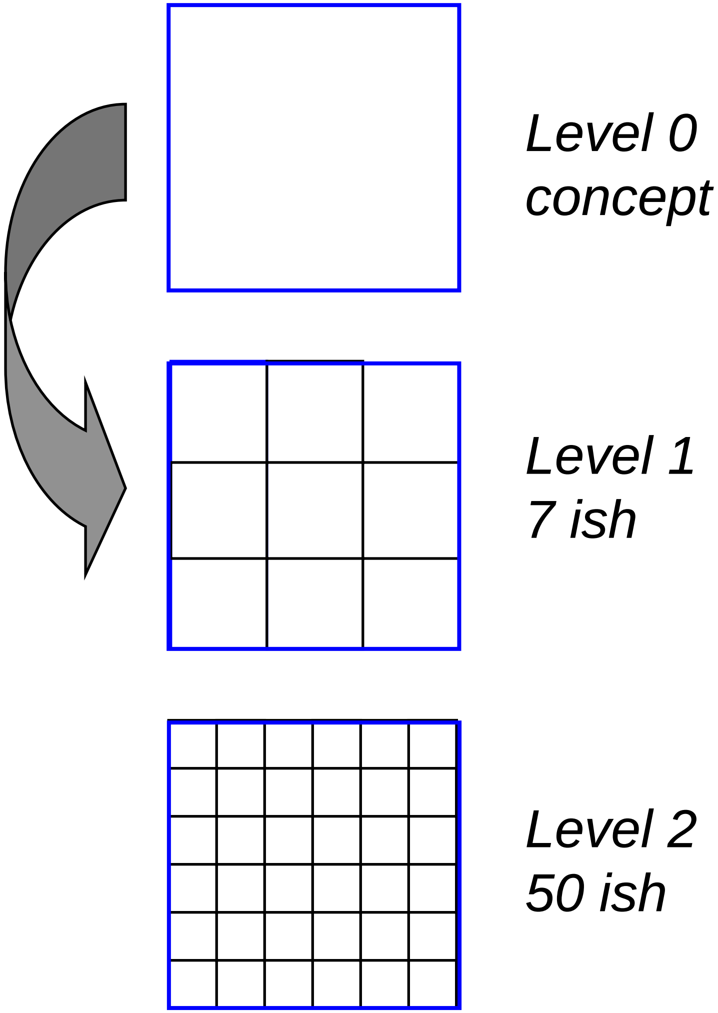
- How do the elements and their structure allow the higher level value related function to emerge? What is the mapping of level 2 elements to the value related internal functions?

- Is their evidence of interfacing functions?
- Is their evidence of value related functions other than the primary one?
- Is their evidence of internal “supporting functions” other than primary functions

Objective - Understanding Level 1 Architecture

- **The reason for analyzing the level 1 processes (at least the internal processes related to value) and the level 2 elements of form is to try to understand the *level 1 architecture* of the product/system**

- **This will probably have only 7+/-2 internal functions and elements of form, however the designation of these (and their interfaces) is critical to the success of the system**



Decompose 2 and Modularize 1

“Down 2, up 1”

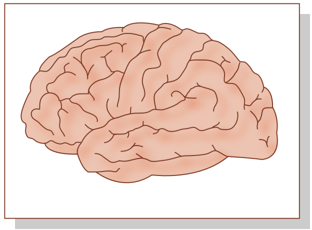
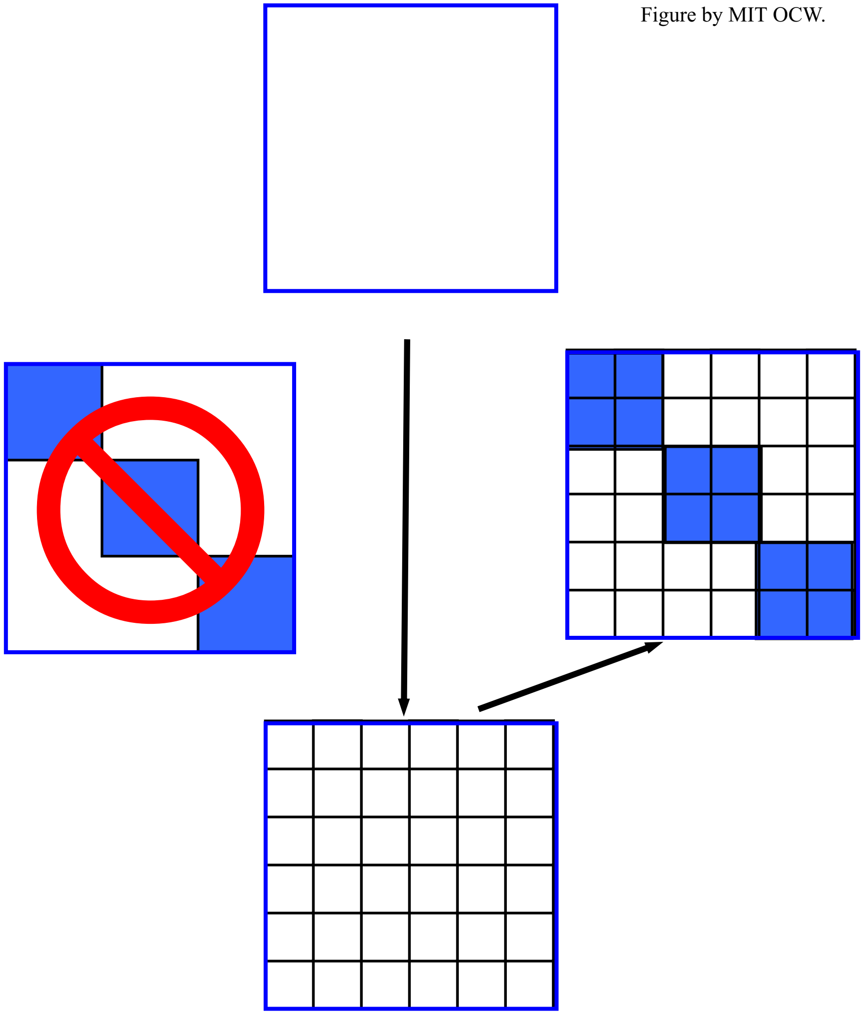


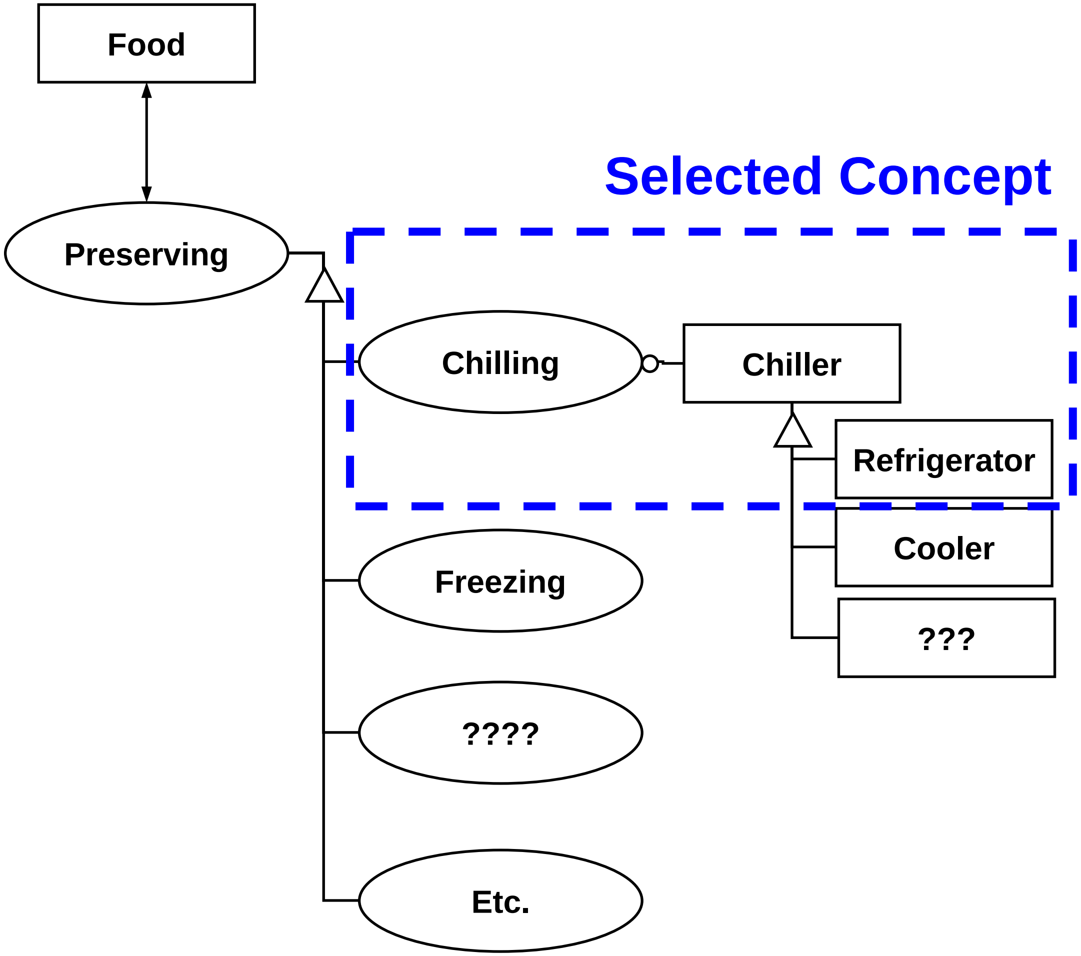
Figure by MIT OCW.

- Start with the system at level 0
- “Dip down” to level 2, and determine structure their
- “Pull up to “recombine” to at level 1 modules or abstractions
- The real wisdom about connectivity at level 1 is found at level 2!



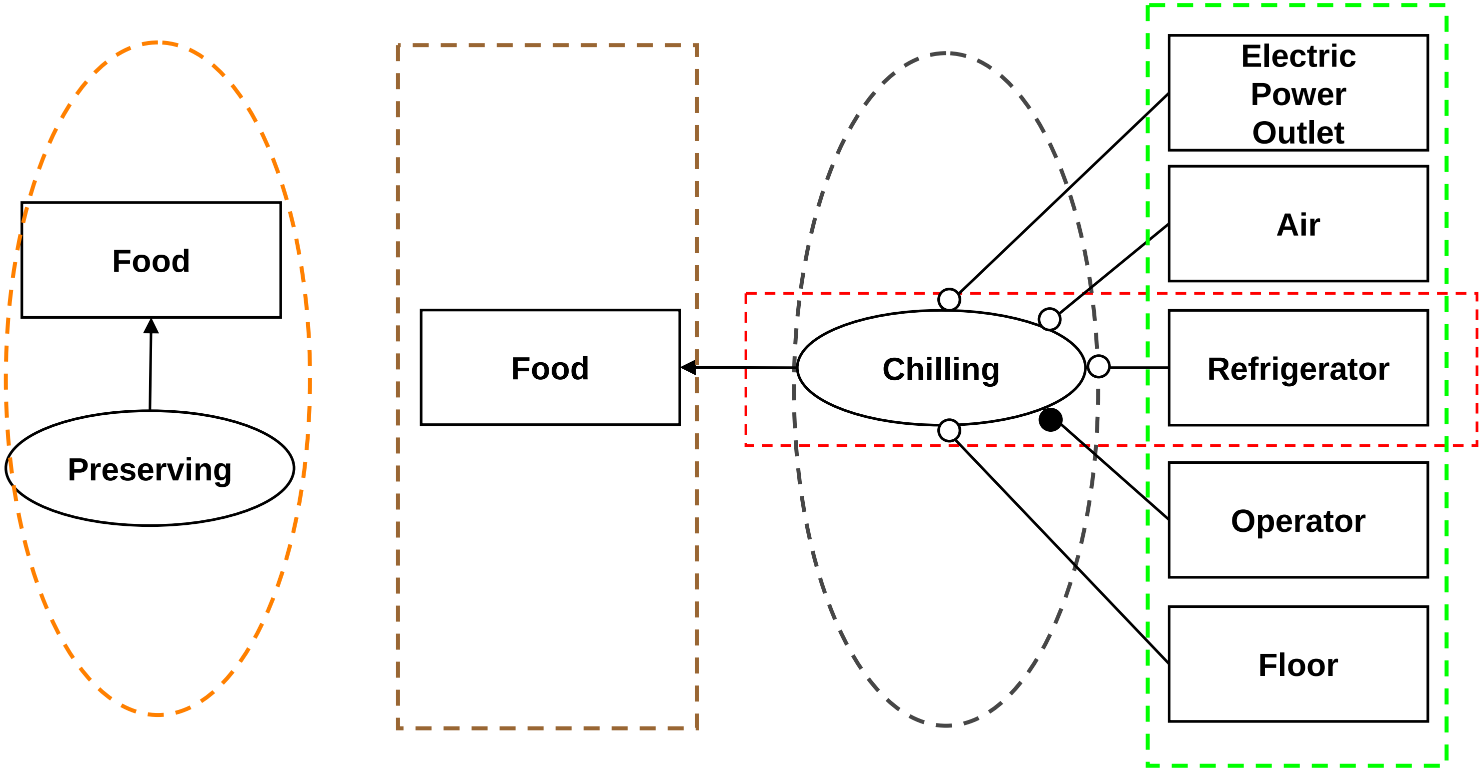
Concepts - Preserving Food

- **Solution neutral statement is: 'preserving food'**
- **Solution specific processes: chilling, freezing, etc.**
- **Solution specific form for chilling: refrigerator, cooler, etc.**
- **Selected concept is chilling with a refrigerator**



▲ Decomposes to
△ Specializes to
◬ Has attribute of

Preserving Food Concept - Refrigerator



Project/system boundary

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

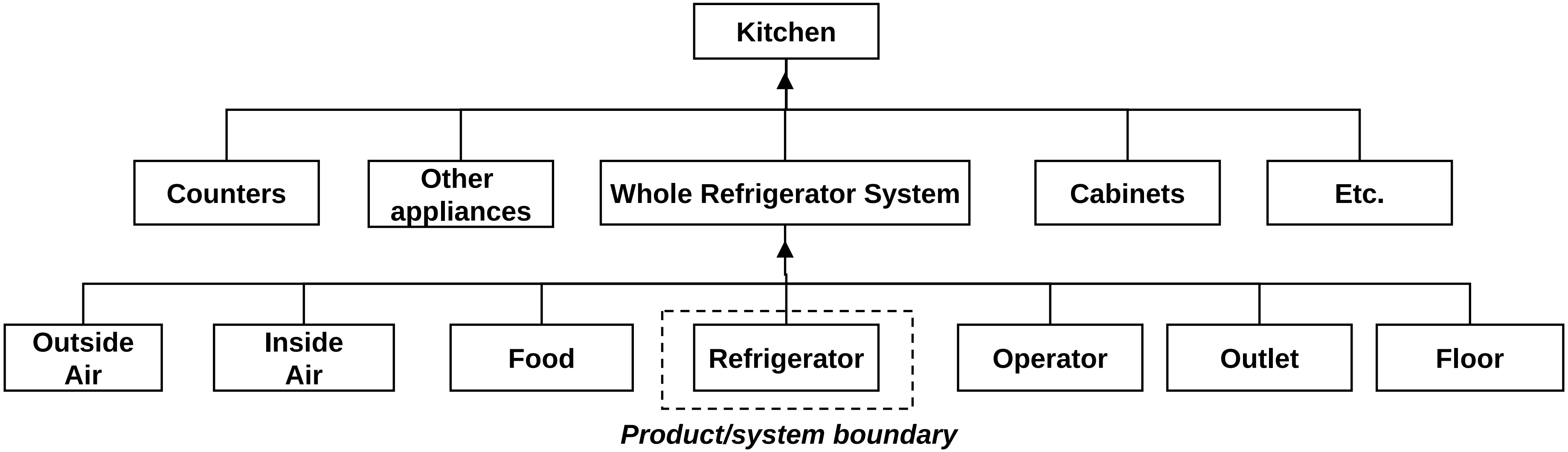
Operand and solution neutral transformation

Operand -value related external object that changes state

Externally delivered value related process

Value related instrument object

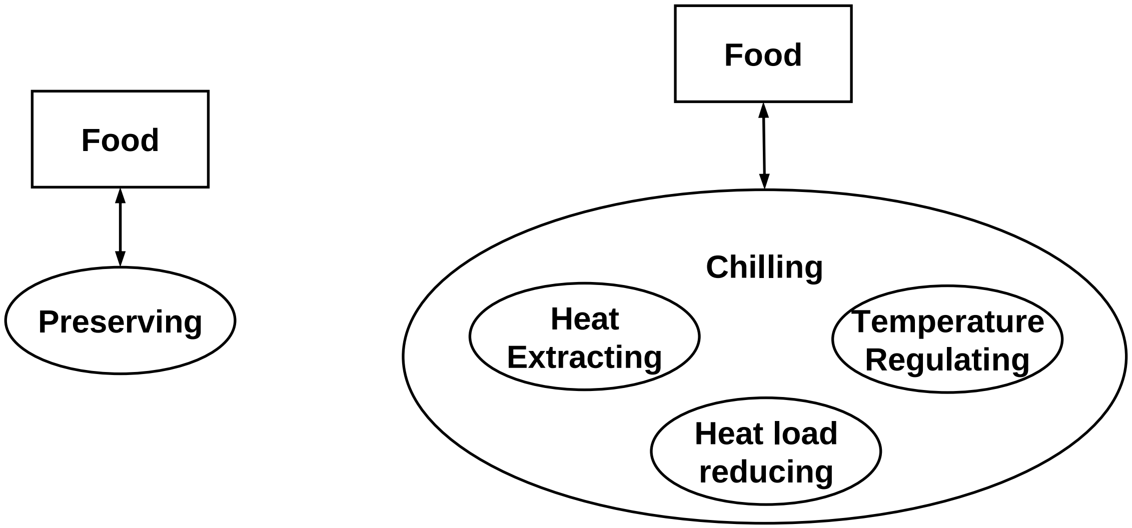
Context - Refrigerator



- **What is the whole product system?**
- **What is the usage context in which it fits?**

NB: a complete job would include the architecture of the usage context not just the objects

Multi-function Concepts for Chilling



▲ Decomposes to
△ Specializes to
◭ Has attribute of

"Chilling" implies cooling, but at a relatively constant temperature above freezing, and hence temperature regulating. Chilling efficiently implies that the ambient heat load on the process be reduced.

Refrigerator - Idealized internal value related processes

- Idealized internal value related processes and operands informed by the concept refrigerator

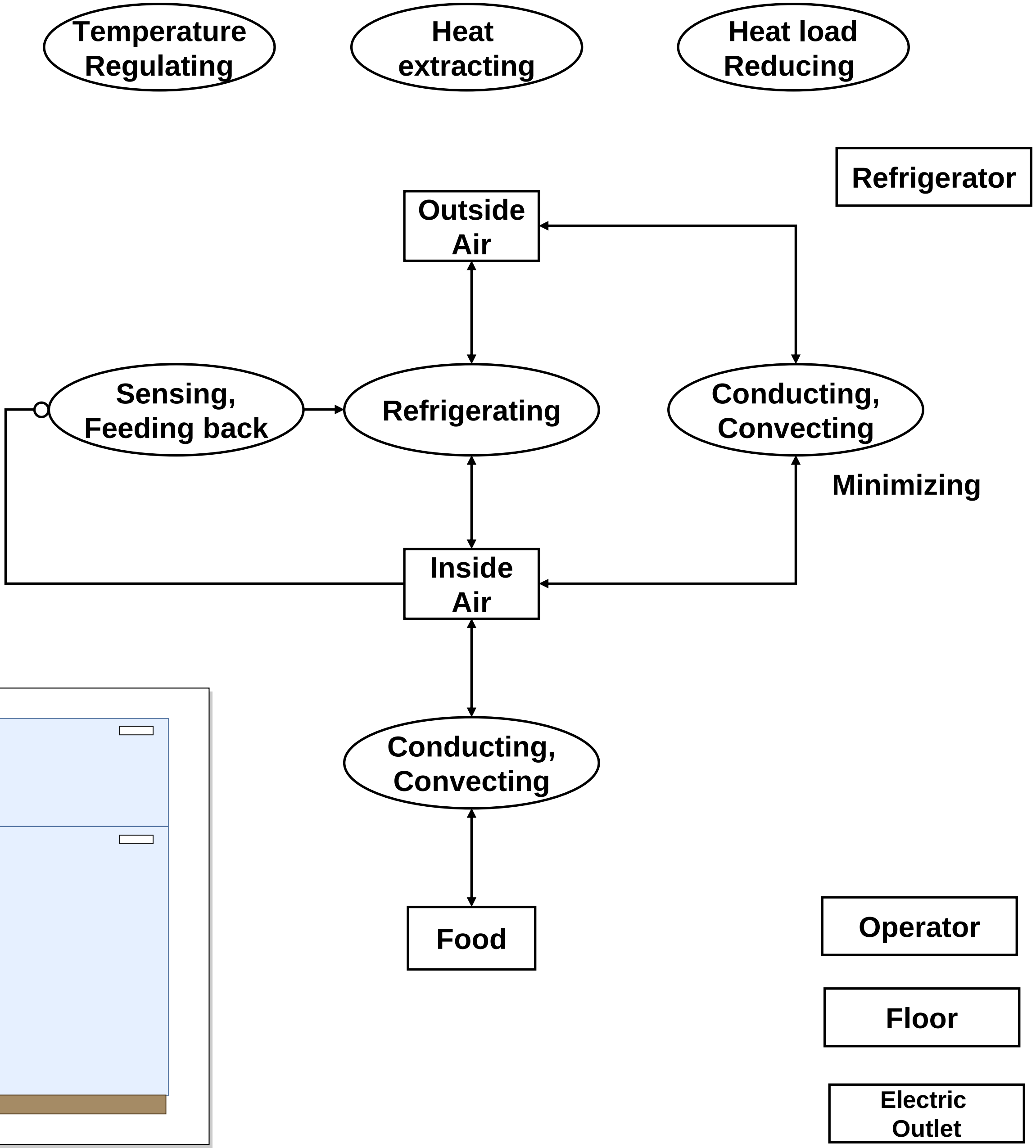
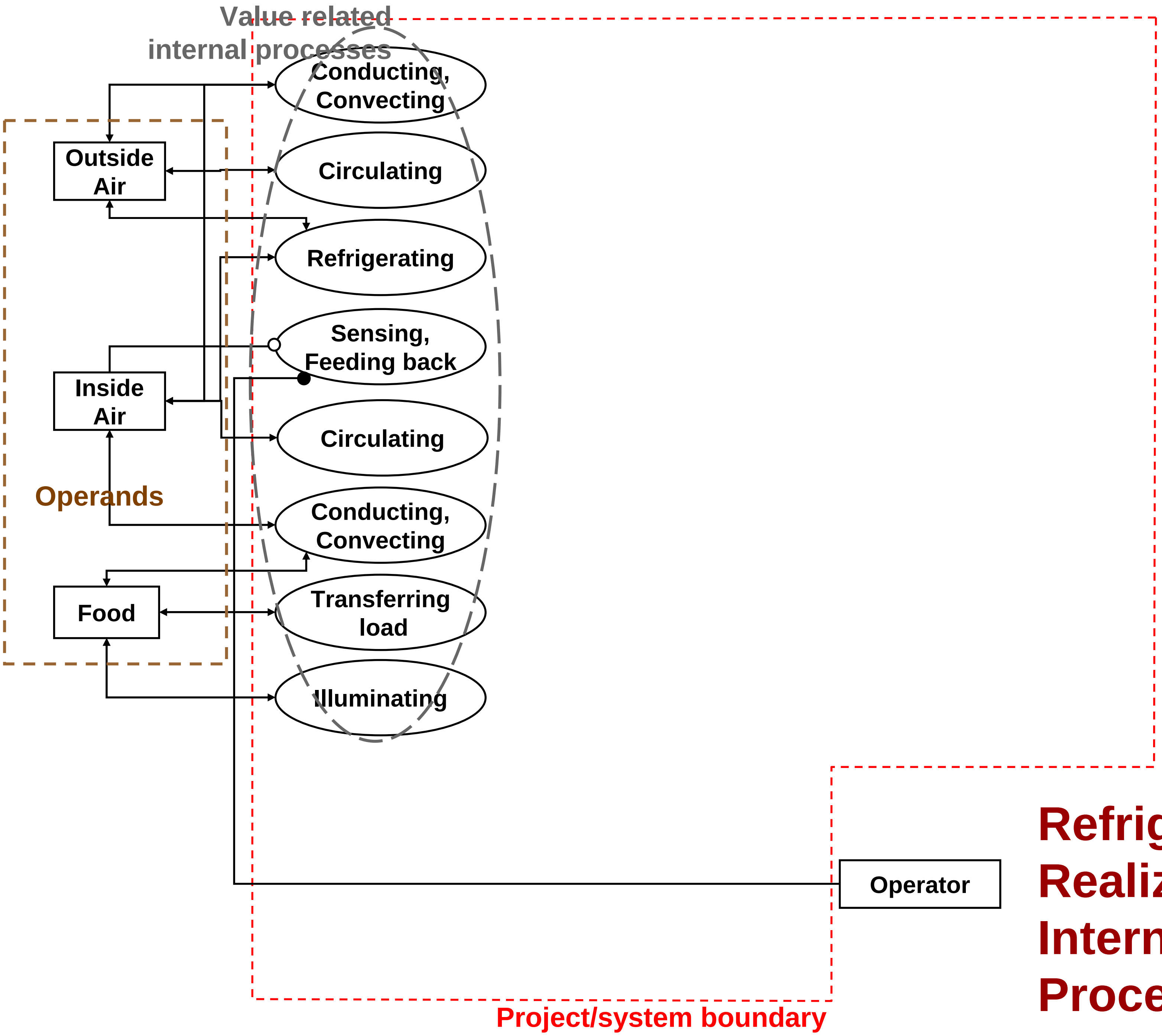


Figure by MIT OCW.



Refrigerator - Realizable Internal Processes

Form of A Simple Refrigerator - List

evaporator fan, motor
evaporator shroud
wiring harness
heat exchanger
accumulator
evaoporator coil
drain tube
drain trought assembly
drain pan
condenser
fan switch
light switch
switch housing
condensor fan, schroud, motor
condensor schroud
compressor
condensate heater loop
compressor wiring harness
running capacitor
compressor mount
starting relay
overload protector
fan bracket (condensor fan)
control knob and indicator
controller (refig temp)
control bracket
light bulb (4)
light stand off
light socket
control light and socket
power cord
light terminator
egg tray
dairly compartment assembly
door shelf assembly (3)
door gasket
door trim
door pannel
door handle
door
switch depressor
light diffuser
fan guard
door hinge (top)
door hinge (bottom)
door frame (top, sides)
back cover
legs, rollers
base assembly
kickplate
cabinet shelf ladder (l,r) support
glass shelf assembly
shelf assembly with track for basket
roll-out basket assembly
crisper roller (l,r,l,r)
crisper slide (l,r,l,r)
center crisper assembly
crisper tray assembly
crisper glass assembly
criper draw assembly
louvered grille
compressor fan shroud assembly
compressor shroud assembly
control pannel
evaporator cover
cabinet assembly

- **Plus whole product system elements: floor, electric outlet, operator, plus the operands - air, food**
- **Parts list for a simple refrigerator, no ice maker, cold water dispenser, freezer, etc.**
- **66 part types in list is already simplified**
- **Actually about 210 part numbers on bill of material**

Rationalize Element List

- Start with a list of 40-70 elements $(7+/-2)^2$
- Try to rationalize element listing to a more manageable number 20-40
- Some important elements that are highly integral will have to be *expanded* in order to identify elements important to the level 1 architecture - e.g. cabinet assembly to cabinet, insulation, structure, outer panels
- Some elements can be grouped into abstractions because their fine structure will not influence and understanding of the level 1 architecture- e.g. top and bottom door hinges and parts to hinge
- Some elements can be grouped into a *class* (the elements are instances), so long as the topology does not require them to be identified individually - e.g. crisper roller, door trim, light bulbs (4)
- Some elements should be explicitly maintained as external interfaces - better to keep them in the list - e.g legs, power cord
- Some elements appear to be associated with other value functions - e.g. outer panels, condensate heater loop
- Set aside for the moment the elements associated with what appeart to be tertiary elements or connectors that do not affect level 1 architecture - e.g. light diffuser, fan guard

Abstraction - Class/Instance Relationship

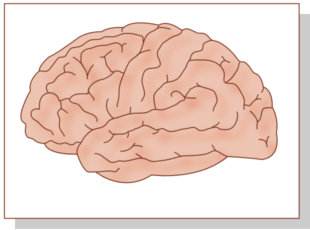
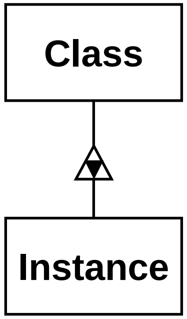


Figure by MIT OCW.

- One type of abstraction that is used is the relationship between a class of something, and an instance of the class
- The instance is referred to as an instantiation of the class
- For example:
  - A class of Ford Explorer vs. my Ford Explorer (VIN #)
  - A class of procedure vs. a instance of a procedure call
  - What is a part number?



▲ Decomposes to
△ Specializes to
◬ Has attribute of
⟁ Has an instance of

Structure Refrigerator

	cabinet assembly	hinges	legs, rollers	glass shelves	roll-out basket assembly	crisper assemblies and draws	evaporator fan, motor	heat exchanger	accumulator	evaporator coil	condenser	compressor	condensor fan, motor	wiring harness	fan switch	light switch	control knob and indicator	controller (refig temp)	light bulb (4)	power cord	door	door handle	dairly compartment assembly	door shelf assembly (3)	condensate disposal asbl.	louvered grille	door pannel	electic outlet	floor	operator	ourside air	inside air	food
cabinet assembly	x	m	m	m		m	m	m		m	m	m	m		m	m		m	m	m					m	m							
hinges	tb	x																			m												
legs, rollers	t		x																														
glass shelves	tw			x	m																												
roll-out basket assembly	w			t	x																												
crisper assemblies and draws	tw					x																											
evaporator fan, motor	tw						x							e																			
heat exchanger	tw							x	mi	mi	m	m													m								
accumulator								t	x	mi																							
evaporator coil	t						da	t	t	x															m								
condenser	tw							tt			x	m																					
compressor	tw							t			t	x		e																			
condensor fan, motor	tw										da		x	e																			
wiring harness	w						t					t	t	x	e	e	e	e	e	e													
fan switch	t													t	x																		
light switch	t													t		x																	
control knob and indicator	tw													t			x	m															
controller (refig temp)	w													t			t	x															
light bulb (4)	tw													t					x														
power cord	t													t						x								e					
door	tb	t													t	t					x	m	m	m			m						
door handle																					t	x											
dairly compartment assembly																					t		x										
door shelf assembly (3)																					t			x									
condensate disposal asbl.	t							t		t															x								
louvered grille	t													da												x							
door pannel																					t						x						
electic outlet																				t								x					
floor			t																										x				
operator																	t					t								x			
outside air	t										t										t				t	ov					x		
inside air	t			s	s	s	t			t								t			t		s	s								x	
food				t	t		t																t	t						w			x

- **Structure reveals 4 bus like elements: cabinet, door, wire harness, heat exchanger**
- **Other elements can lump**
- **Observed half band width of about 7!! For non bus elements**

Observations - Structure Refrigerator

- Some first level elements identifiable
  - As tightly interacting clusters
  - As busses
  - Others stand alone
- Criteria applied was to minimize “interaction” in form, which will minimize interfaces
- What about left over elements like hinge (which is mechanically connected to door and cabinet) and switches (which are connected to the cabinet, but interact with the door)
- Note that there is no information on function here!

Hierarchy

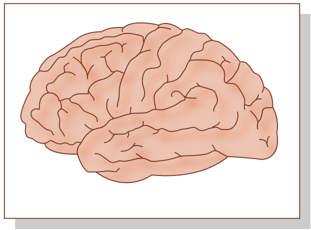


Figure by MIT OCW.

- Hierarchy is defined as: a system in which grades or classes are ranked one above the other

- Hierarchy in technical systems occurs due to:
  - The level of decomposition of form, function, etc.
  - The creation of layered systems

- Hierarchy tends to hide information more than one layer away from the reference point.

- Examples?

A Hierarchy of Elements and Complexity

- **Use context**
- **Whole product system**
- **Product/system**
- **Module**
  - **A a collection of (1...n) parts which are defined by some intent (e.g. integration) to be a distinct sub-system**
- **Part**
- **Detail**

Complexity can be measured at the atomic part level, which is either the part of functional detail level - you must which when define the complexity of a system

A product that is a part (to the customer) is integral. Others are modular.

Parts and Details

Part

- **A part is an element that you cannot take apart and then reconstitute in its original form - it has been irreversibly implemented [no link to function],** ***or***
- **A part is an element that you cannot take apart without destroying its ability to deliver its function [explicit link to function]**

Detail

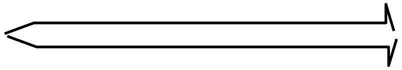
- **An element of a part (so a part can be a system)**

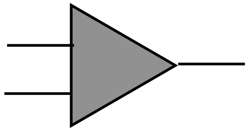
Atomic part

- **A part,** ***or***
- **The details of a part which have independent function**

Making Judgments about Details

- **Even the simplest part has details:**
  - **Nail has details**
  - **Op amp has details**
  - **Instruction has details**





If x > y, then …..

- **The judgment is: are these details of parts important to consider at the level of the system you are examining?**

Structure Refrigerator

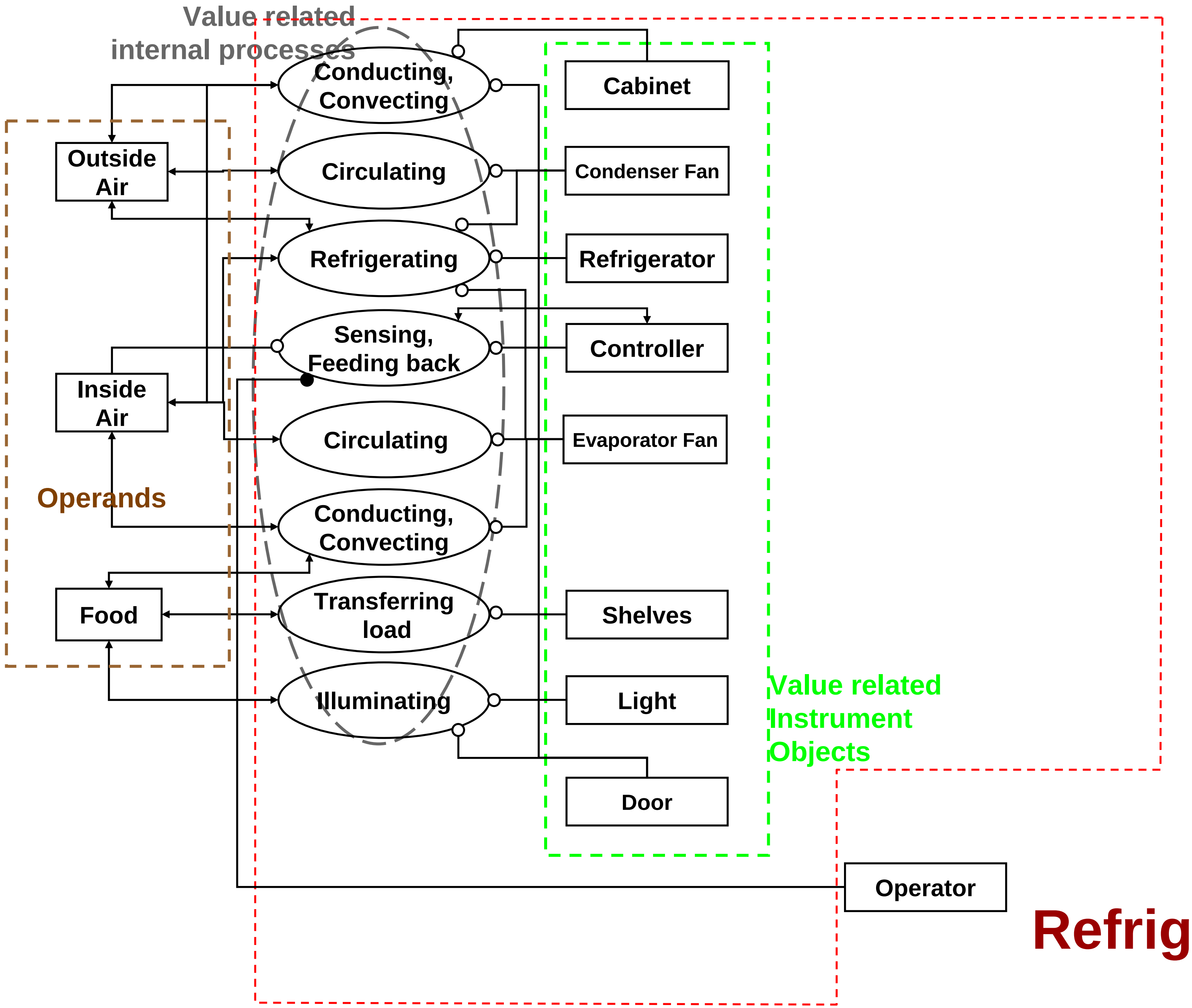
	cabinet assembly	hinges	legs, rollers	glass shelves	roll-out basket assembly	crisper assemblies and draws	evaporator fan, motor	heat exchanger	accumulator	evaporator coil	condenser	compressor	condensor fan, motor	wiring harness	fan switch	light switch	control knob and indicator	controller (refig temp)	light bulb (4)	power cord	door	door handle	dairly compartment assembly	door shelf assembly (3)	condensate disposal asbl.	louvered grille	door pannel	electic outlet	floor	operator	ourside air	inside air	food
cabinet assembly	x	m	m	m		m	m	m		m	m	m	m		m	m		m	m	m					m	m							
hinges	tb	x																			m												
legs, rollers	t		x																														
glass shelves	tw			x	m																												
roll-out basket assembly	w			t	x																												
crisper assemblies and draws	tw					x																											
evaporator fan, motor	tw						x							e																			
heat exchanger	tw							x	mi	mi	m	m													m								
accumulator								t	x	mi																							
evaporator coil	t						da	t	t	x															m								
condenser	tw							tt			x	m																					
compressor	tw							t			t	x		e																			
condensor fan, motor	tw										da		x	e																			
wiring harness	w						t					t	t	x	e	e	e	e	e	e													
fan switch	t													t	x																		
light switch	t													t		x																	
control knob and indicator	tw													t			x	m															
controller (refig temp)	w													t			t	x															
light bulb (4)	tw													t					x														
power cord	t													t						x								e					
door	tb	t													t	t					x	m	m	m			m						
door handle																					t	x											
dairly compartment assembly																					t		x										
door shelf assembly (3)																					t			x									
condensate disposal asbl.	t							t		t															x								
louvered grille	t													da												x							
door pannel																					t						x						
electic outlet																				t								x					
floor			t																										x				
operator																	t					t								x			
outside air	t										t										t				t	ov					x		
inside air	t			s	s	s	t			t								t			t		s	s								x	
food				t	t		t																t	t						w			x

- **Structure reveals 4 bus like elements: cabinet, door, wire harness, heat exchanger**
- **Other elements can lump**
- **Observed half band width of about 7!! For non bus elements**

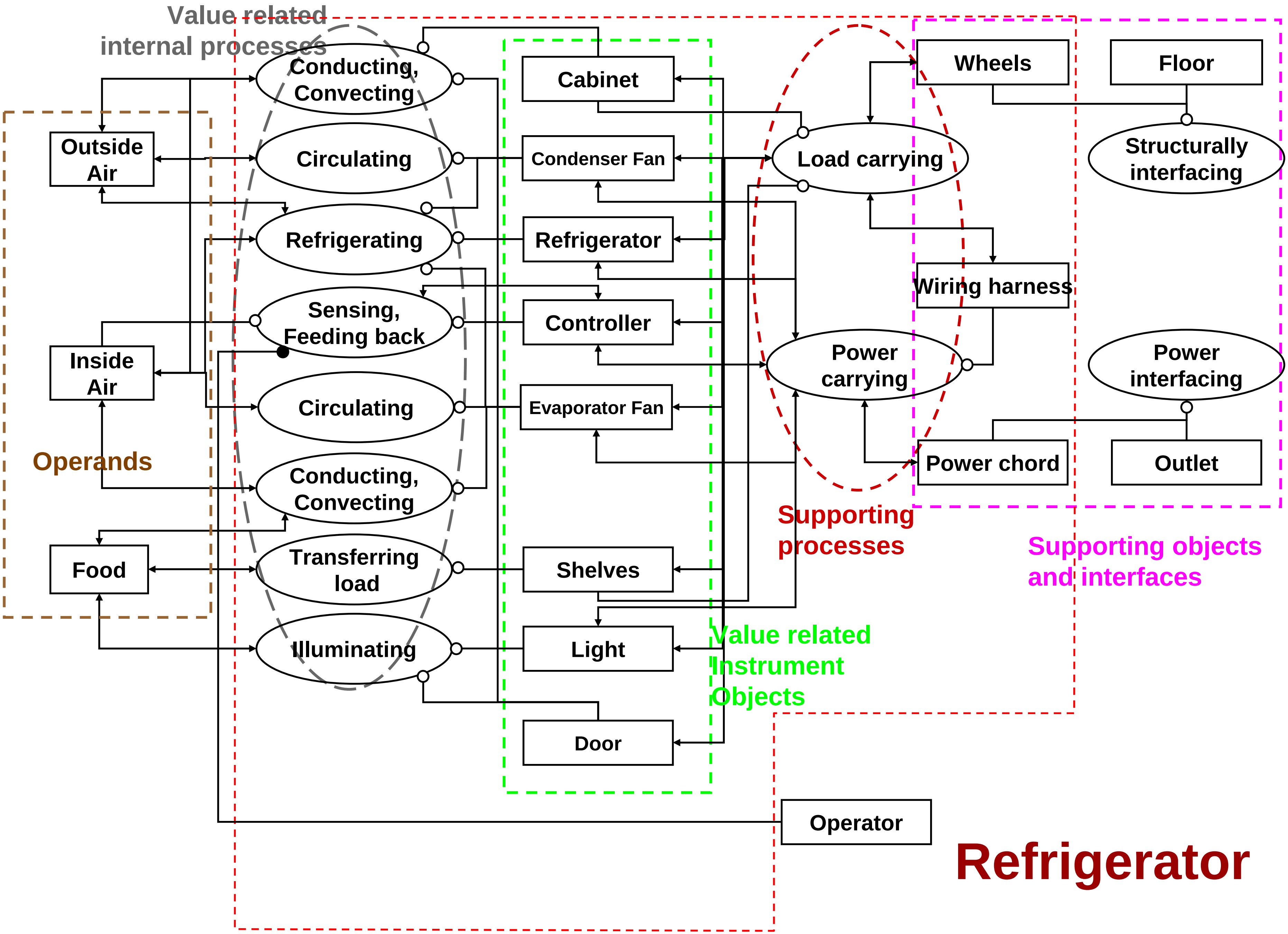
Mapping of Level 2 Elements onto Function

	cabinet assembly	hinges	legs, rollers	glass shelves	roll-out basket assembly	crisper assemblies and draws	evaporator fan, motor	heat exchanger	accumulator	evaporator coil	condenser	compressor	condensor fan, motor	wiring harness	fan switch	light switch	control knob and indicator	controller (refig temp)	light bulb (4)	power cord	door	door handle	dairly compartment assembly	door shelf assembly (3)	condensate disposal asbl.	louvered grille	door pannel	electic outlet	floor	operator	ourside air	inside air	food
conducting and convecting(load)	I																				I										e	e	
circulating (outside air)													I																		e		
refirgerating							I	I	I	I	I	I	I					I													e	e	
sensing, feedback																	I	I												a		I	
circulating (inside air)							I								I																	e	
conducting and convecting(inside)							I																									e	e
transfering load (from food)				I	I	I																	I	I									e
illuminating																I			I														e
load carrying	I	I		e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e		I	e	e	e	e	e	e						
power carrying							e					e	e	I	e	e	e	e	e														
structurally interfacing			I																										I				
power interfacing																				I								I					
appearance improving	I																									I	I						
condensate evaporating																									I								

- **Mapping on functions identifies some alignment and some misalignment with clustering done on form**



Refrigerator



Value related internal processes

Operands

Outside Air

Inside Air

Food

Conducting, Convecting

Circulating

Refrigerating

Sensing, Feeding back

Circulating

Conducting, Convecting

Transferring load

Illuminating

Cabinet

Condenser Fan

Refrigerator

Controller

Evaporator Fan

Shelves

Light

Door

Value related Instrument Objects

Load carrying

Power carrying

Supporting processes

Wheels

Floor

Structurally interfacing

Wiring harness

Power interfacing

Power chord

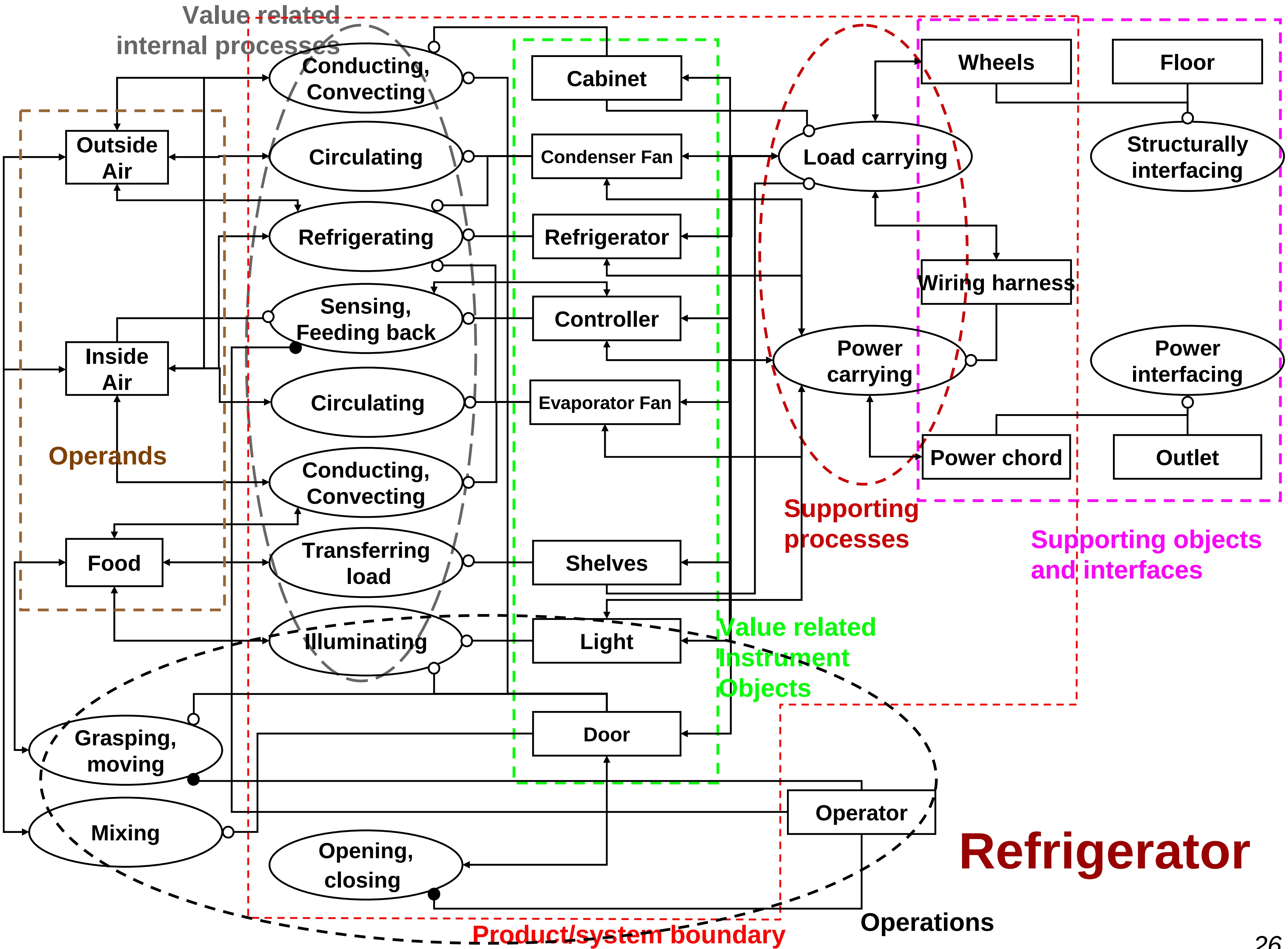
Outlet

Supporting objects and interfaces

Operator

Refrigerator

Refrigerator



Value related internal processes

Conducting, Convecting

Circulating

Refrigerating

Sensing, Feeding back

Circulating

Conducting, Convecting

Transferring load

Illuminating

Outside Air

Inside Air

Food

Operands

Cabinet

Condenser Fan

Refrigerator

Controller

Evaporator Fan

Shelves

Light

Door

Value related Instrument Objects

Load carrying

Power carrying

Supporting processes

Wheels

Floor

Structurally interfacing

Wiring harness

Power interfacing

Power chord

Outlet

Supporting objects and interfaces

Grasping, moving

Mixing

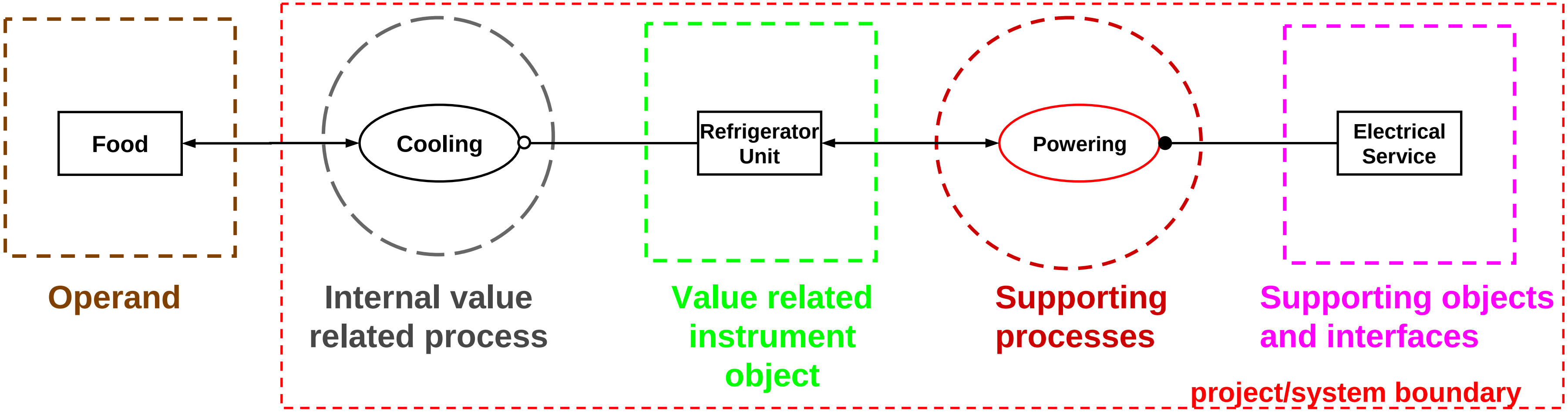
Opening, closing

Operator

Operations

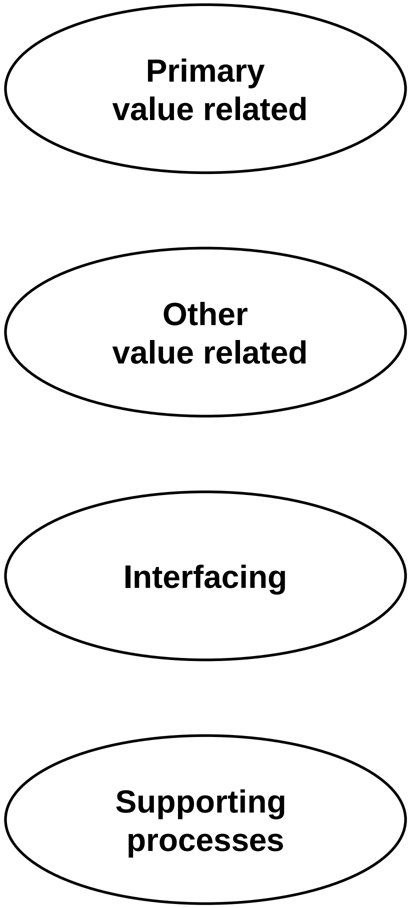
Product/system boundary

Architecture in Layers



- Architecture can be though of as existing in layers
- The value related operand and other intermediate operands
- The processes the touch on the direct value delivery
- The instrument objects
- Several more layers of supporting processes (typically structurally supporting, powering, controlling) and associated instrument objects

Primary Value, Other Value, Interfacing, and Supporting Internal Functions - Refrigerator



Primary value related

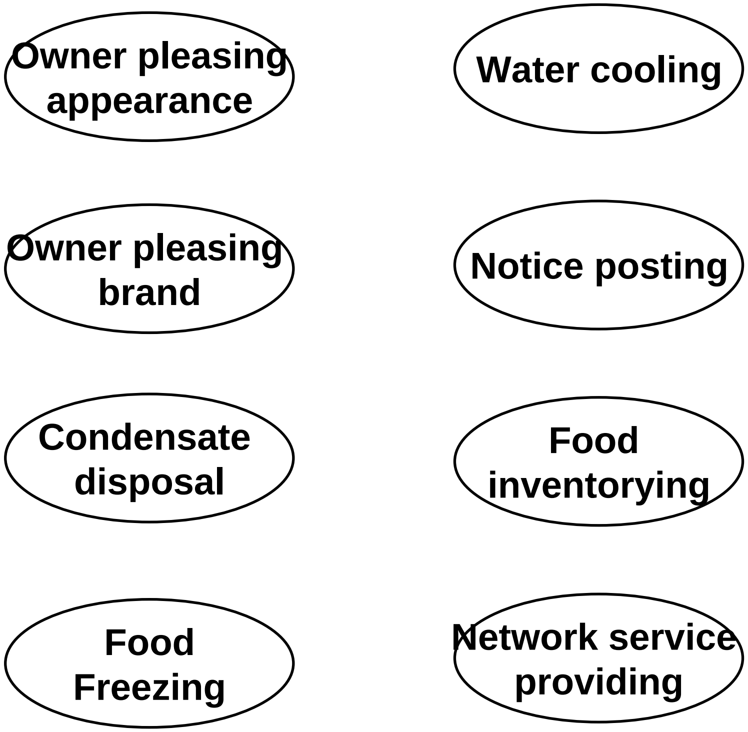
Other value related

Interfacing

Supporting processes

- All product/systems have a primary value related process - chilling food
- Many have other value related processes - ice making, dispensing cold water, freezing, ?
- All have interface processes - with food, floor, ?
- Most have other internal processes that support the value processes - structurally supporting, ?

Refrigerator - Additional Value Processes



Owner pleasing appearance

Owner pleasing brand

Condensate disposal

Food Freezing

Water cooling

Notice posting

Food inventorying

Network service providing

What influence with these processes have on the architecture?

A Notional Precedence of Processes

- **Sub-processes which support the primary externally-delivered process linked to value**

Then -

- **Other sub-processes linked to other necessary externally-delivered processes**

Then -

- **Interfacing processes**
- **Supporting/connecting (matter related) processes**

Then-

- **Powering (energy related) processes**
- **Controlling/regulating (information related) processes**

Zig - Zagging (after Suh)

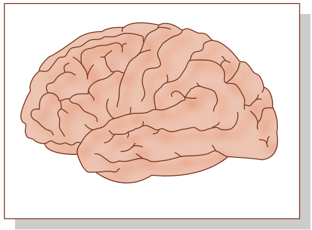
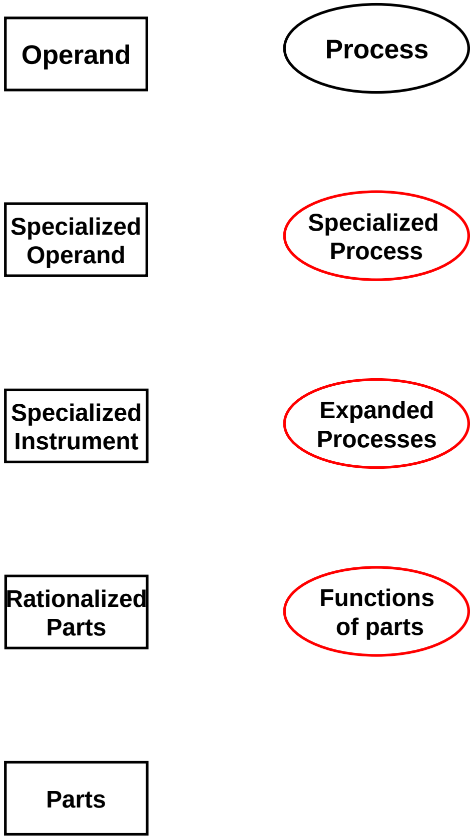


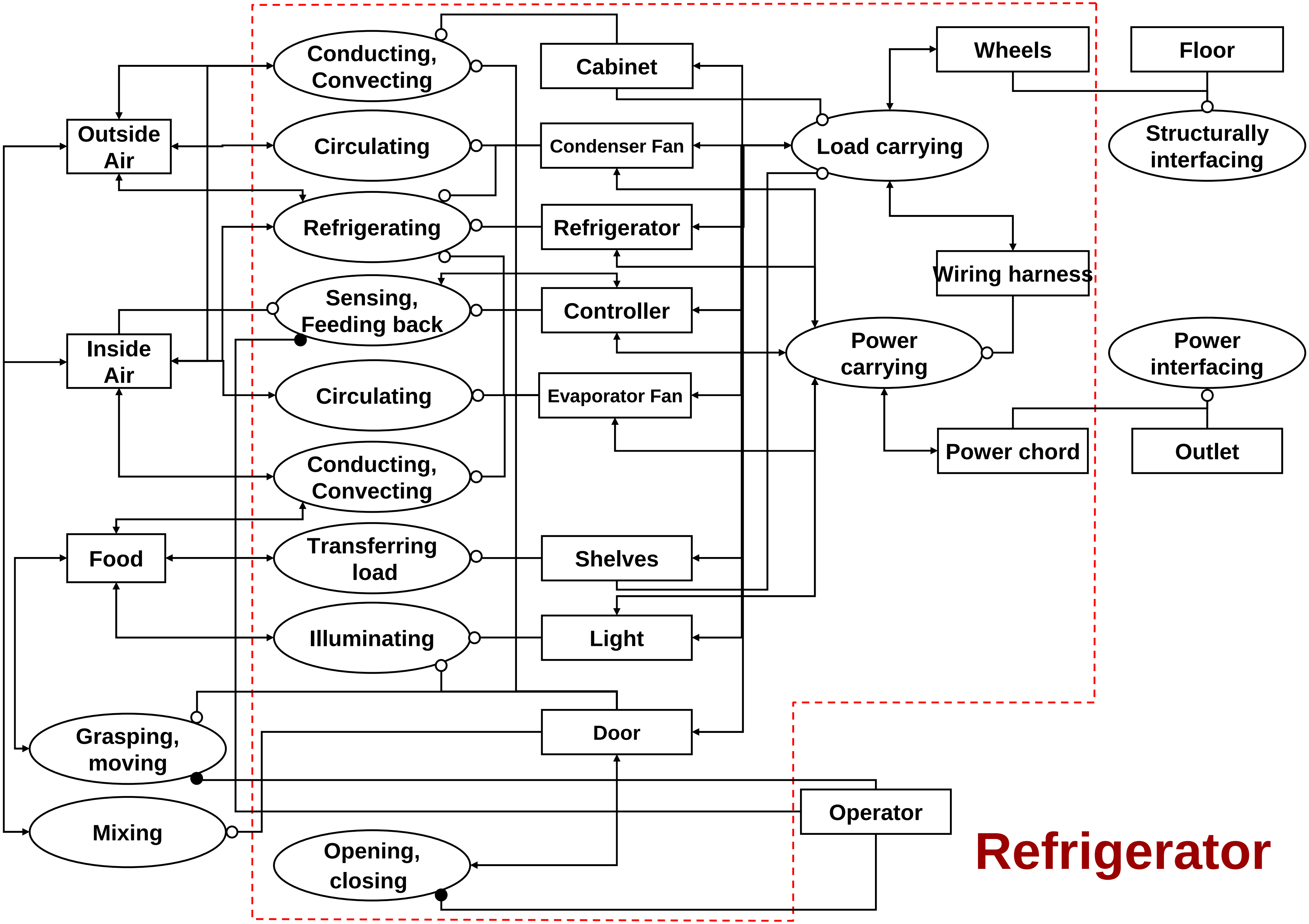
Figure by MIT OCW.

- In thinking through system, you tend to start in one domain (object or process)
- Think and work in that domain as long as it makes sense
- Then switch to the other
- And do this recursively
- Called by Suh **zig-zagging**



More Magic

- **Refrigerator**
- **Cooler**
- **So why do we use refrigerators? The complexity must be worth it!**



Conducting, Convecting

Circulating

Refrigerating

Sensing, Feeding back

Circulating

Conducting, Convecting

Transferring load

Illuminating

Opening, closing

Outside Air

Inside Air

Food

Grasping, moving

Mixing

Cabinet

Condenser Fan

Refrigerator

Controller

Evaporator Fan

Shelves

Light

Door

Operator

Load carrying

Power carrying

Wheels

Wiring harness

Power chord

Floor

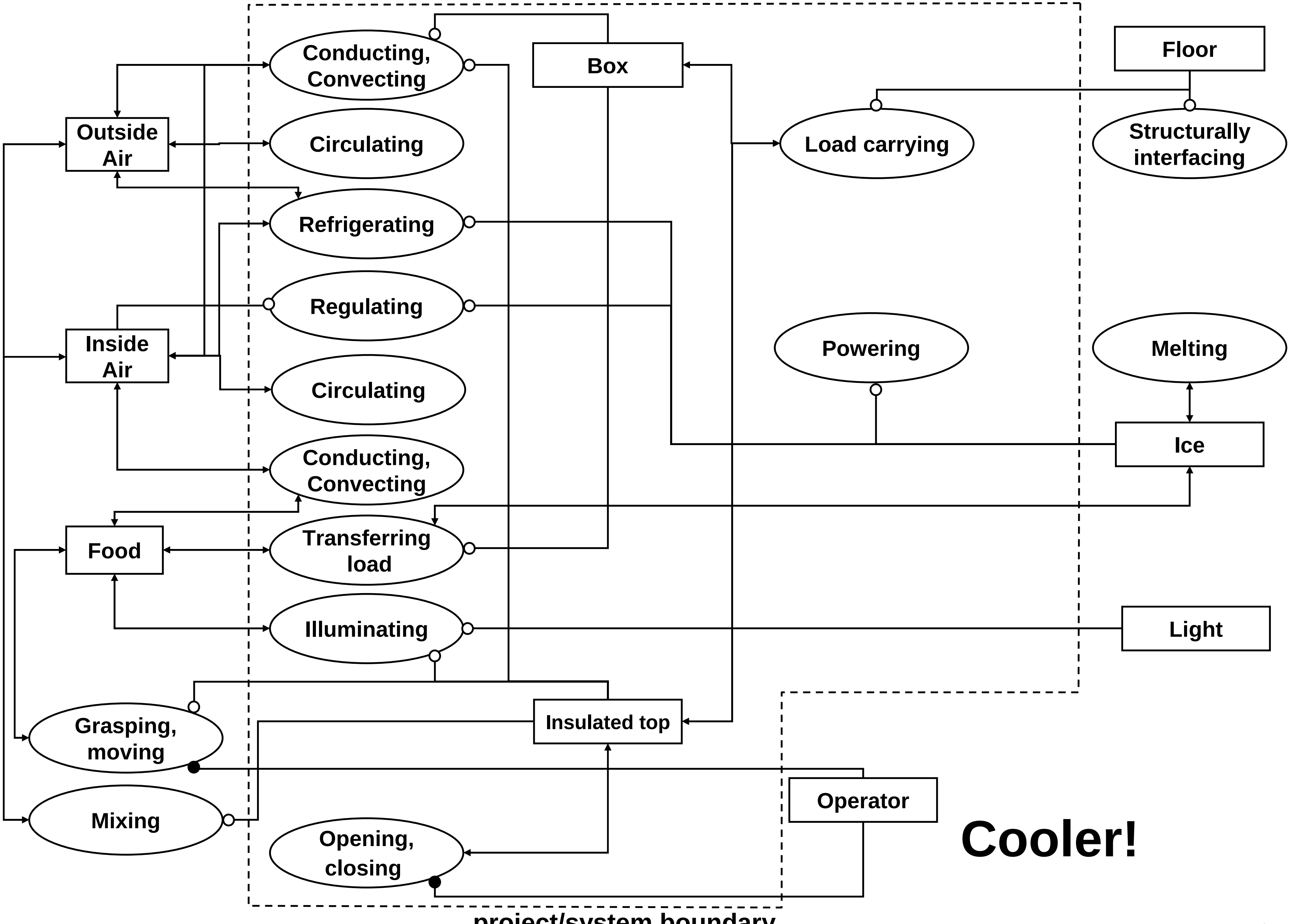
Structurally interfacing

Power interfacing

Outlet

Refrigerator

Product/system boundary


Outside Air
Conducting, Convecting
Box
Floor
Circulating
Load carrying
Structurally interfacing
Refrigerating
Regulating
Inside Air
Powering
Melting
Circulating
Ice
Conducting, Convecting
Food
Transferring load
Illuminating
Light
Grasping, moving
Insulated top
Operator
Mixing
Opening, closing
Cooler!
project/system boundary

Process-Object Observation

- **The externally delivered processes of a system act on its operand(s), whose states are changed by the processes (e.g. person/rider)**
- **The *primary* externally delivered process of a system acts on one or more of the operand(s), and is linked to value (e.g. transporting)**
- **Other externally delivered processes potentially act on the same or other operands, and deliver additional value, which may enhance or differentiate the product, but should not be confused with its primary function (e.g. ride smoothing, aesthetics)**

Process-Object Observation (cont.)

- Zooming will identify the internal processes which combine to produce the emergent external process
- Zooming to internal processes will often reveal additional operands which are concealed or otherwise incidental to the processes and operands associated with value (e.g. flow and vortex in the whistle example). These additional operands should also be considered in the whole product system
- Zooming to internal processes will identify the objects which act as agents and instruments of the internal processes that are linked to value delivery
- There is almost always one or two layers of process-object pairs in a layer below those that are directly linked to value, which support, connect, power, control, etc. the value related instrument objects
- There is then often an additional layer of interfaces to the things external to the product/system

Summary Process-Object Architecture

- Operand object, processes and instrument objects can be shown with OPM in one graph of an architecture, which is interchangeable with a matrix representation
- Layers appear of processes and objects that directly link to value, and supporting (connecting, controlling, powering) objects and processes
- Interfaces also have object and process character that must be identified and controlled
- Note that we have now created a view of the architecture that contains 5 layers:
  - The specific concept - the specific operand, process and instrument
  - 2 layers down of decomposition of operands, processes and instruments
  - 2 layers up - the whole product system and use case
  - Remarkably close to one of the deliverables of the architect!

Complexity

Defined:

- **Having many interrelated, interconnected or interwoven elements and interfaces**

Therefore

- **A complex system requires a great deal of information to specify**
- **Complexity is an absolute and quantifiable system property (once a measure and atomic level are defined)**
- **Apparent complexity is the perception that something is complex. <u>Complicated</u> things have high apparent complexity**

It is the role of the architect to manage the evolution of complexity in such a way that a complex system does not appear complicated

¿ What Makes a System Complex?

- Think back to the structure of simple systems analyzed earlier
- How have we dealt with trying to “simplify it, so that complex systems are not so complicate?
- How might you measure it?

Approaches to Managing Complexity

- **Abstractions**
  - **Class - Instance**
  - **Generalization - Specialization**
- **Hierarchy**
- **Decomposition (and zooming)**
- **Recursion**

Abstraction

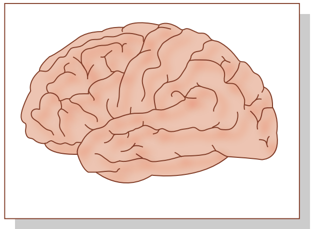


Figure by MIT OCW.

- Abstraction defined as:
  - expression of quality apart from the object
  - having only the intrinsic nature rather than the form
- Abstraction can be used in both function and form
- Abstraction can be used to characterize and hide more detailed structure and behavior within them, allowing simpler representation of the "surface"
- Examples: cup, pen, routine_name

Abstraction - Class/Instance Relationship

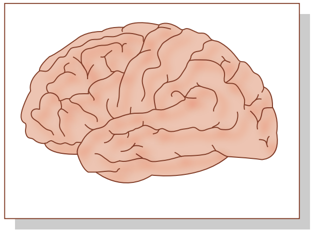
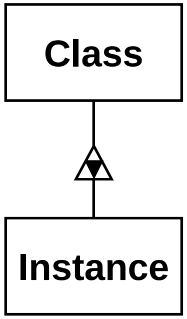


Figure by MIT OCW.

- One type of abstraction that is used is the relationship between a class of something, and an instance of the class
- The instance is referred to as an instantiation of the class
- For example:
  - A class of Ford Explorer vs. my Ford Explorer (VIN #)
  - A class of procedure vs. a instance of a procedure call
  - What is a part number?



▲ Decomposes to
△ Specializes to
◬ Has attribute of
⧊ Has an instance of

Specialization

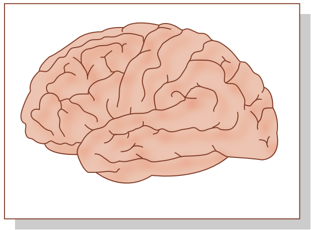
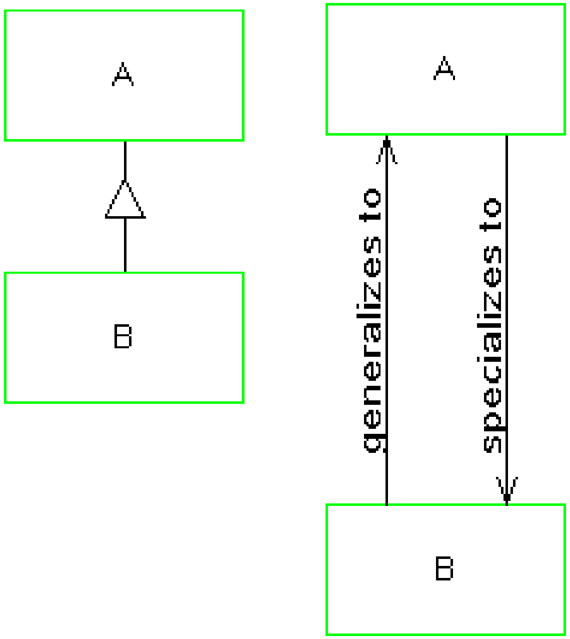
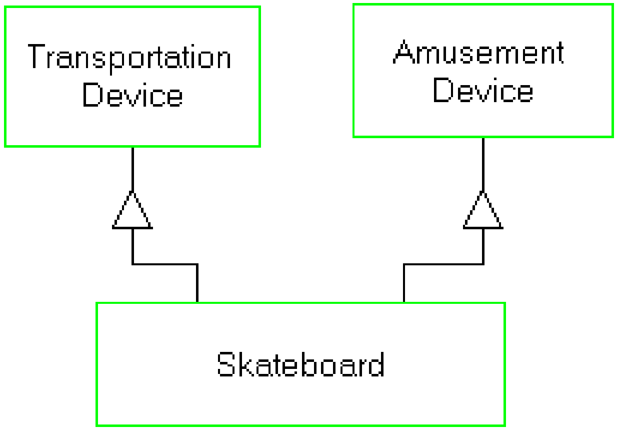


Figure by MIT OCW.

- **<u>Specialization/Generalization</u>**
  - **The relationship between a general object and its specialized forms**





Hierarchy

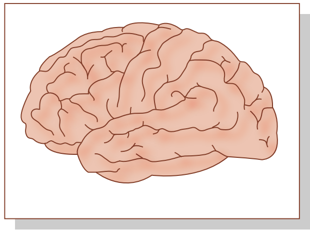


Figure by MIT OCW.

- Hierarchy is defined as: a system in which grades or classes are ranked one above the other

- Hierarchy in technical systems occurs due to:
  - The level of decomposition of form, function, etc.
  - The creation of layered systems

- Hierarchy tends to hide information more than one layer away from the reference point.

- Examples?

Decomposition of a System

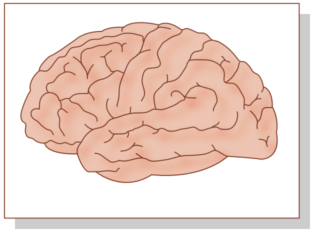
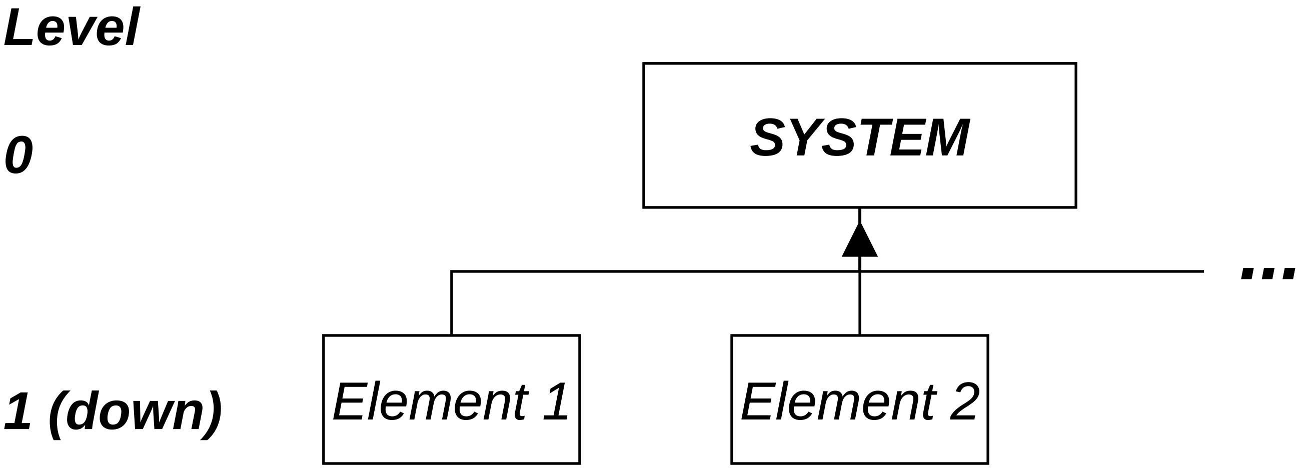


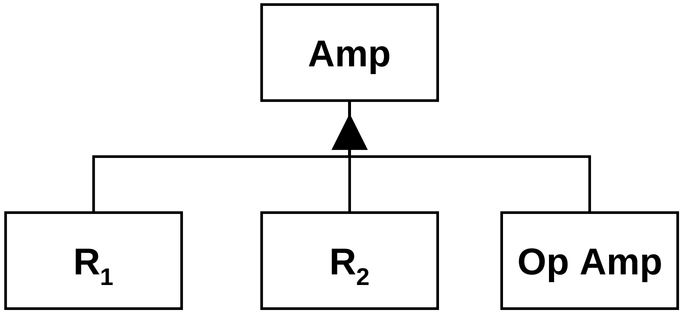
Figure by MIT OCW.

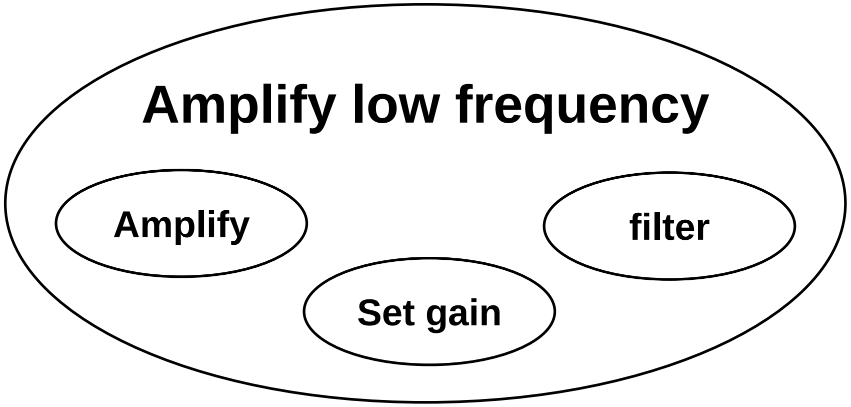


- Decomposition is the division into smaller constituents
- The system object at level 0 decomposes to the element objects at level 1
- The element objects at level 1 aggregate to the system
- This is the *whole - part relationship*, so common that it has its own symbol ▲

Emergence and Zooming of Processes

- **A process can be zoomed into sub-processes**
- **A process *emerges* from sub-processes**
- **The process and sub-processes are not linked in any explicit manner, as the system decomposes into elements or the elements aggregate into the whole**
- **Emergence is a powerful feature of systems - elements and sub-processes can come together to cause a process to emerge**





Repetition and Recursion

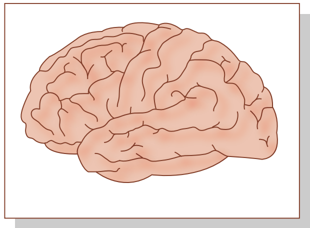
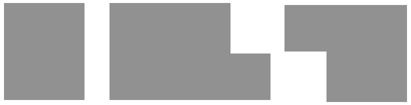


Figure by MIT OCW.

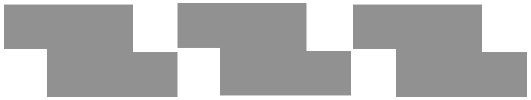
- **<u>Repetition</u> is the use of repeated steps or elements**

- **Processes can be used repeatedly**
  - **Turn left, turn right,   turn left, turn right, …**

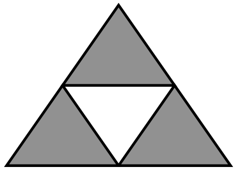
- **Objects can be used recursively as well**
  - **No repetition**



  - **Repetition**



- **<u>Recursion</u> has the additional sense that the process or object uses itself within the whole**



Massachusetts Institute of Technology © Ed Crawley 2007

Ways to Think About Complex Systems

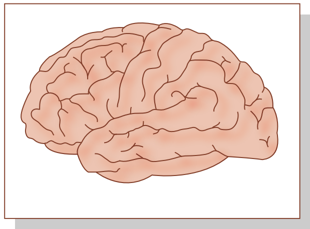


Figure by MIT OCW.

- Holism
- Critical thinking

- Objects (instruments and operands) and processes
- Whole product systems and use context

- Top down, bottom up, middle out, outer in
- 2 down, 1 up levels of decomposition
- Layer of value delivery, supporting, interfacing
- Zig zag between form and function

Dirt Simple Measure of Complexity

- Number of things: N_{things}
- Number of types of things: $N_{types\_of\_things}$
- Number of connections among things: $N_{connections}$
- Number of types of connections: $N_{types\_of\_connections}$

- Simplest measure that captures *all* of these is the sum:
- $C = N_{things} + N_{types\_of\_things} + N_{connections} + N_{types\_of\_connections}$

- A lot of work to calculate!

Summary - Complexity

- Complexity is a technical system is an absolute thing that can be quantified and measured (once a metric and an atomic part have been defined)

- It is the role of the architect to manage complexity (and reduce apparent complexity)

- Common approaches to managing complexity are abstraction, decomposition, hierarchy and recursion, which are separate ideas, but often used in combination

Exercise: Form of a Medium System

- Examine the form of two medium systems, which has already been decomposed to “atomic parts”:
  - skateboard (mechanical) 69 elements of 21 types
  - naval signal flags (informational) 26 elements of 26 types
- Creating abstractions and hierarchy, aggregate the “atomic parts” (at level 2) into level 1 abstractions

¿Reflections on Medium Systems?

- **How did you decompose 0 level product/system or aggregate 2nd level parts to create first level elements?**
- **What techniques did you use to manage complexity in the system?**

The wisdom on 1st level decomposition is found at the 2nd level, not the 1st. The architect must always "dip down" two levels to understand/create the first level down.

Form of a Medium System - Skateboard

- Skateboard is composed of about 69 elements of 21 types
- These can be aggregated into about 9 elements of 5 types

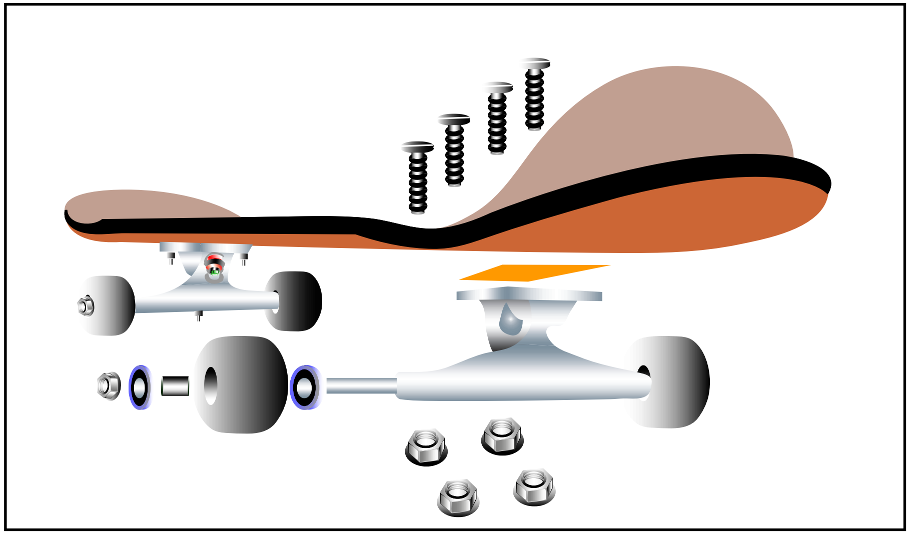
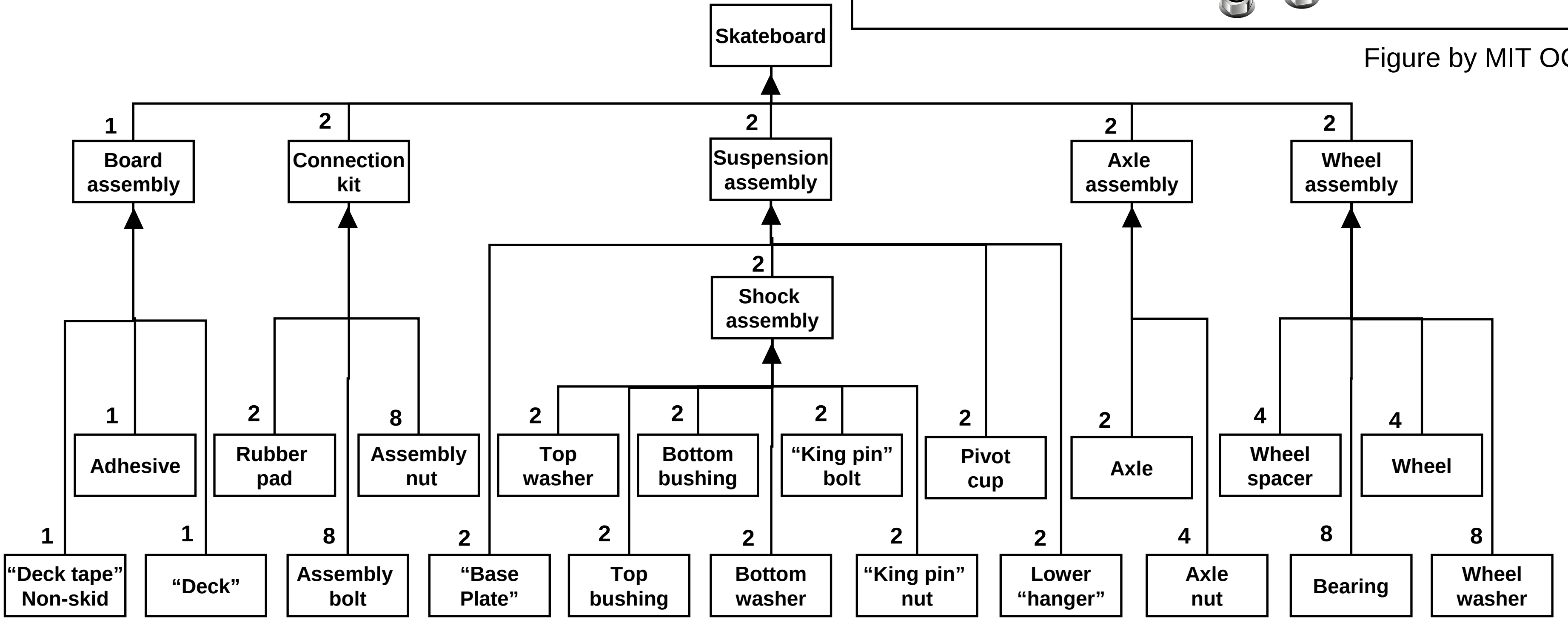


Figure by MIT OCW.



Skateboard

1 Board assembly
2 Connection kit
2 Suspension assembly
2 Axle assembly
2 Wheel assembly

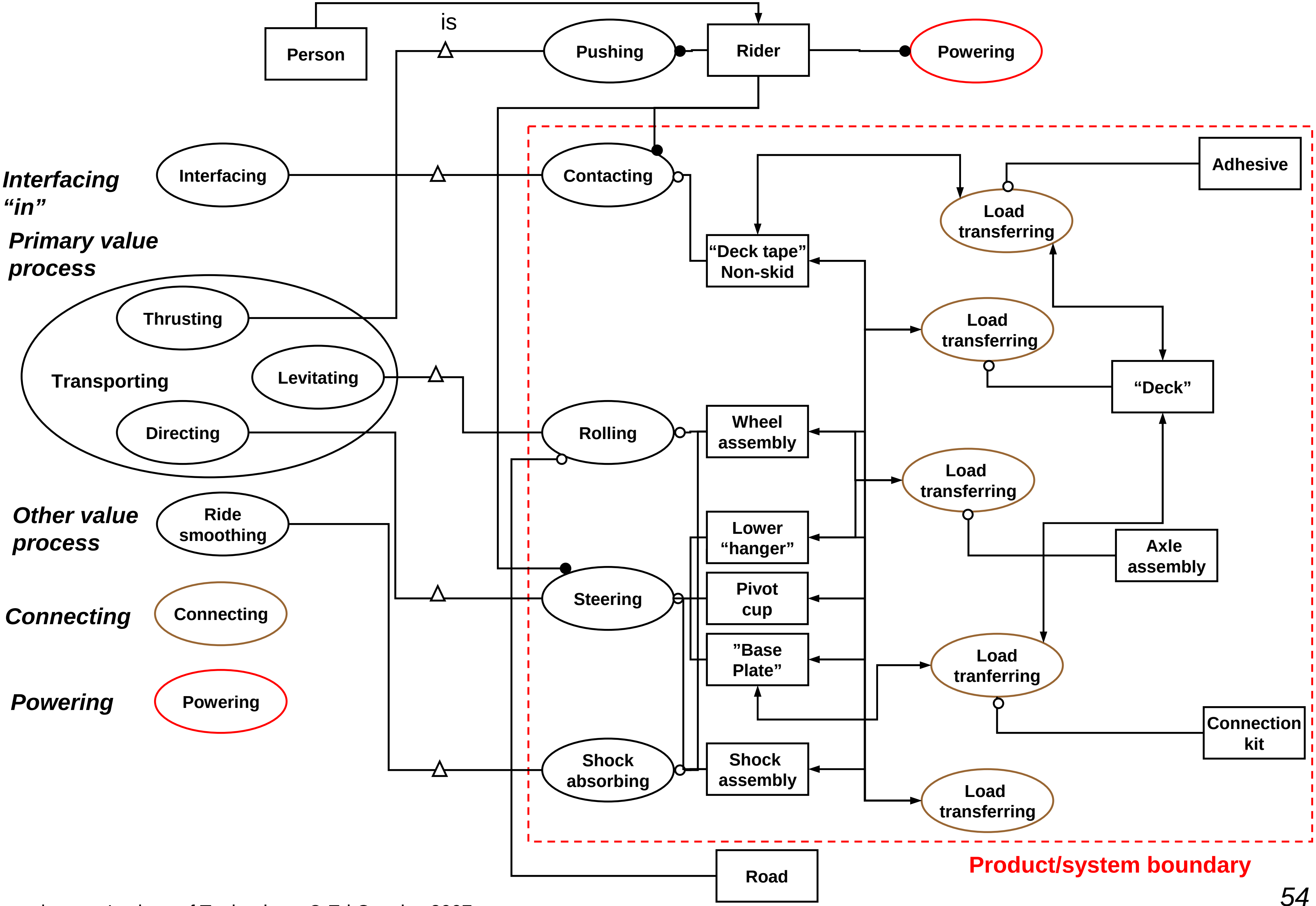
2 Shock assembly

1 Adhesive
2 Rubber pad
8 Assembly nut
2 Top washer
2 Bottom bushing
2 “King pin” bolt
2 Pivot cup
2 Axle
4 Wheel spacer
4 Wheel

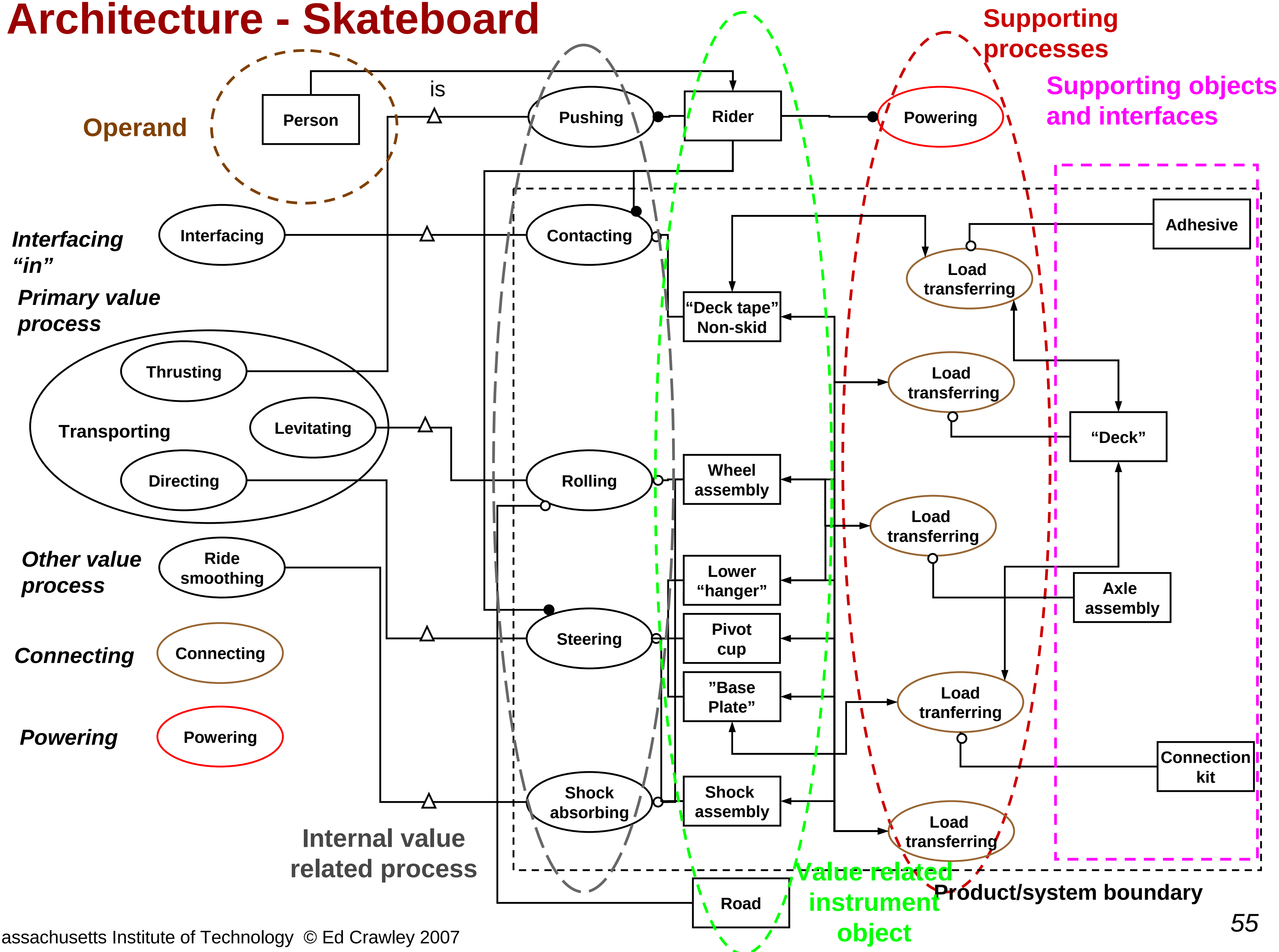
1 “Deck tape” Non-skid
1 “Deck”
8 Assembly bolt
2 “Base Plate”
2 Top bushing
2 Bottom washer
2 “King pin” nut
2 Lower “hanger”
4 Axle nut
8 Bearing
8 Wheel washer

NB: these aggregations are ad hoc and not “correct”

Architecture - Skateboard

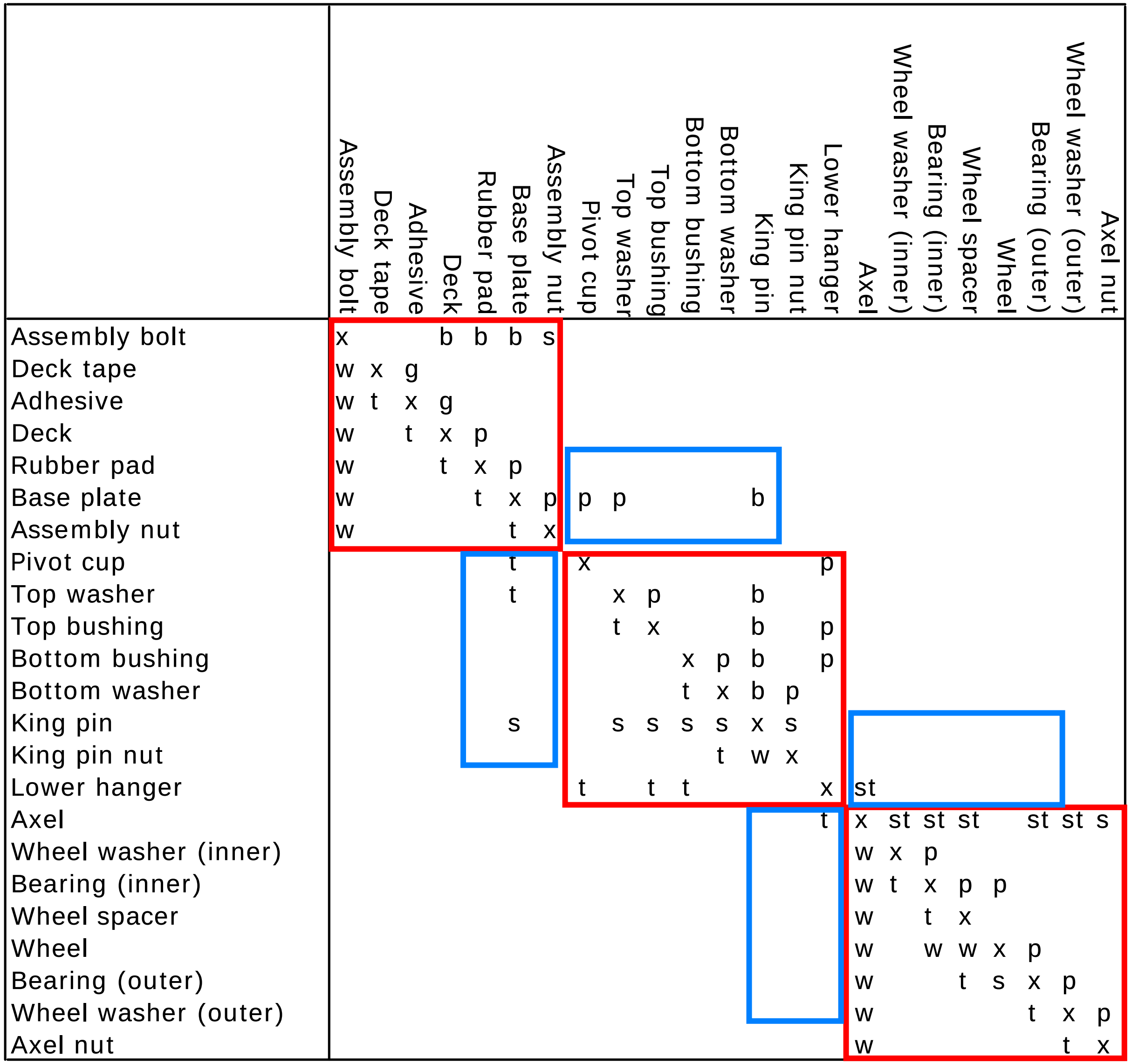


Architecture - Skateboard



Formal Analysis - Skateboard

- Red areas might be modules at level 1
- Blue areas would be interfaces that must be maintained between modules
- Much more about this later!

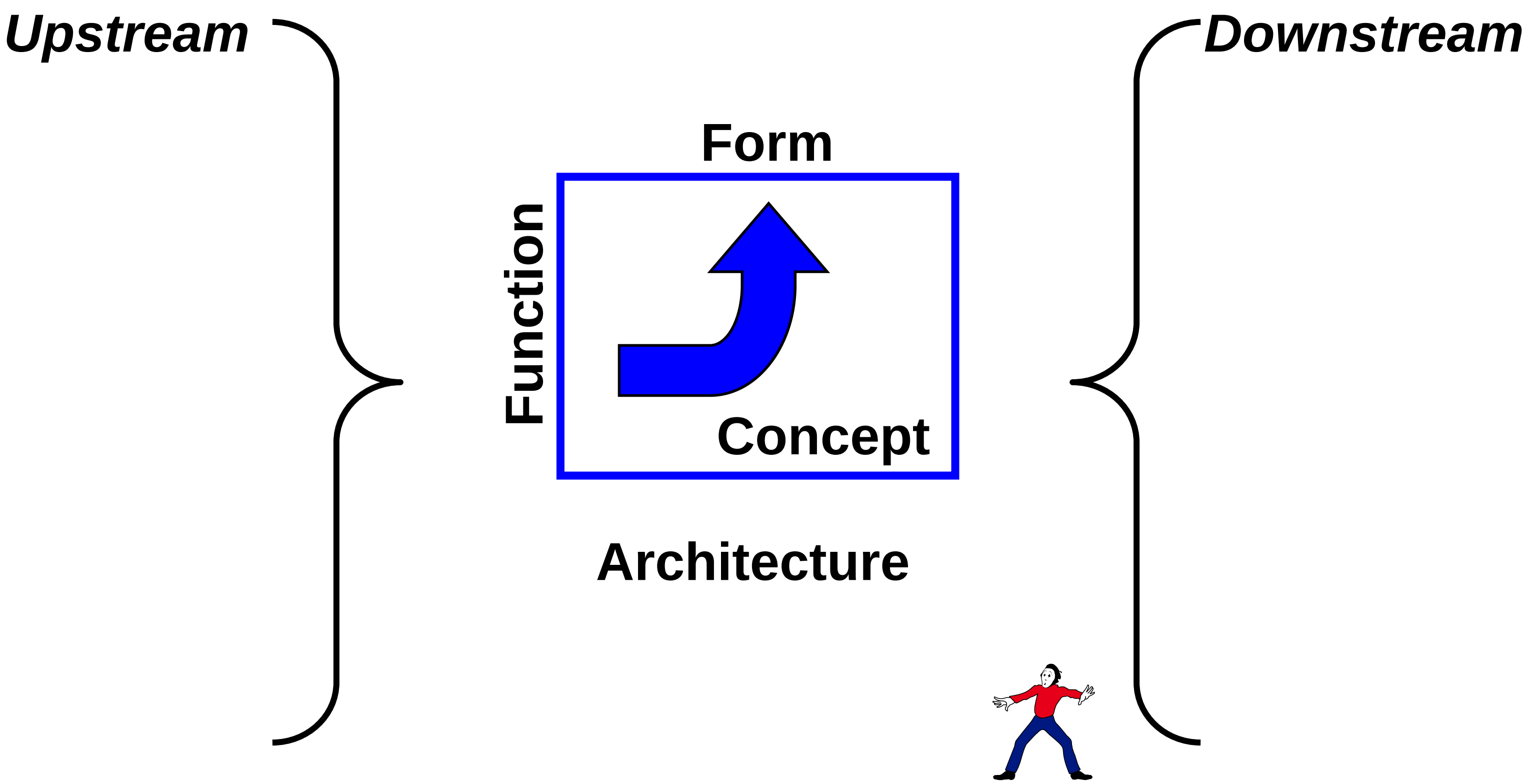


	Assembly bolt	Deck tape	Adhesive	Deck	Rubber pad	Base plate	Assembly nut	Pivot cup	Top washer	Top bushing	Bottom bushing	Bottom washer	King pin	King pin nut	Lower hanger	Axel	Wheel washer (inner)	Bearing (inner)	Wheel spacer	Wheel	Bearing (outer)	Wheel washer (outer)	Axel nut
Assembly bolt	x			b	b	b	s																
Deck tape	w	x	g																				
Adhesive	w	t	x	g																			
Deck	w		t	x	p																		
Rubber pad	w			t	x	p																	
Base plate	w				t	x	p	p	p				b										
Assembly nut	w					t	x																
Pivot cup						t		x							p								
Top washer						t			x	p			b										
Top bushing									t	x			b		p								
Bottom bushing											x	p	b		p								
Bottom washer											t	x	b	p									
King pin						s			s	s	s	s	x	s									
King pin nut												t	w	x									
Lower hanger								t		t	t				x	st							
Axel															t	x	st	st	st		st	st	s
Wheel washer (inner)																w	x	p					
Bearing (inner)																w	t	x	p	p			
Wheel spacer																w		t	x				
Wheel																w		w	w	x	p		
Bearing (outer)																w			t	s	x	p	
Wheel washer (outer)																w					t	x	p
Axel nut																w						t	x

t = touches or tangent
w = within
s = surrounds

g = glued
b = bolted
s = screwwed
p = pressed
st = stacked

Architecture Centric View of the PDP

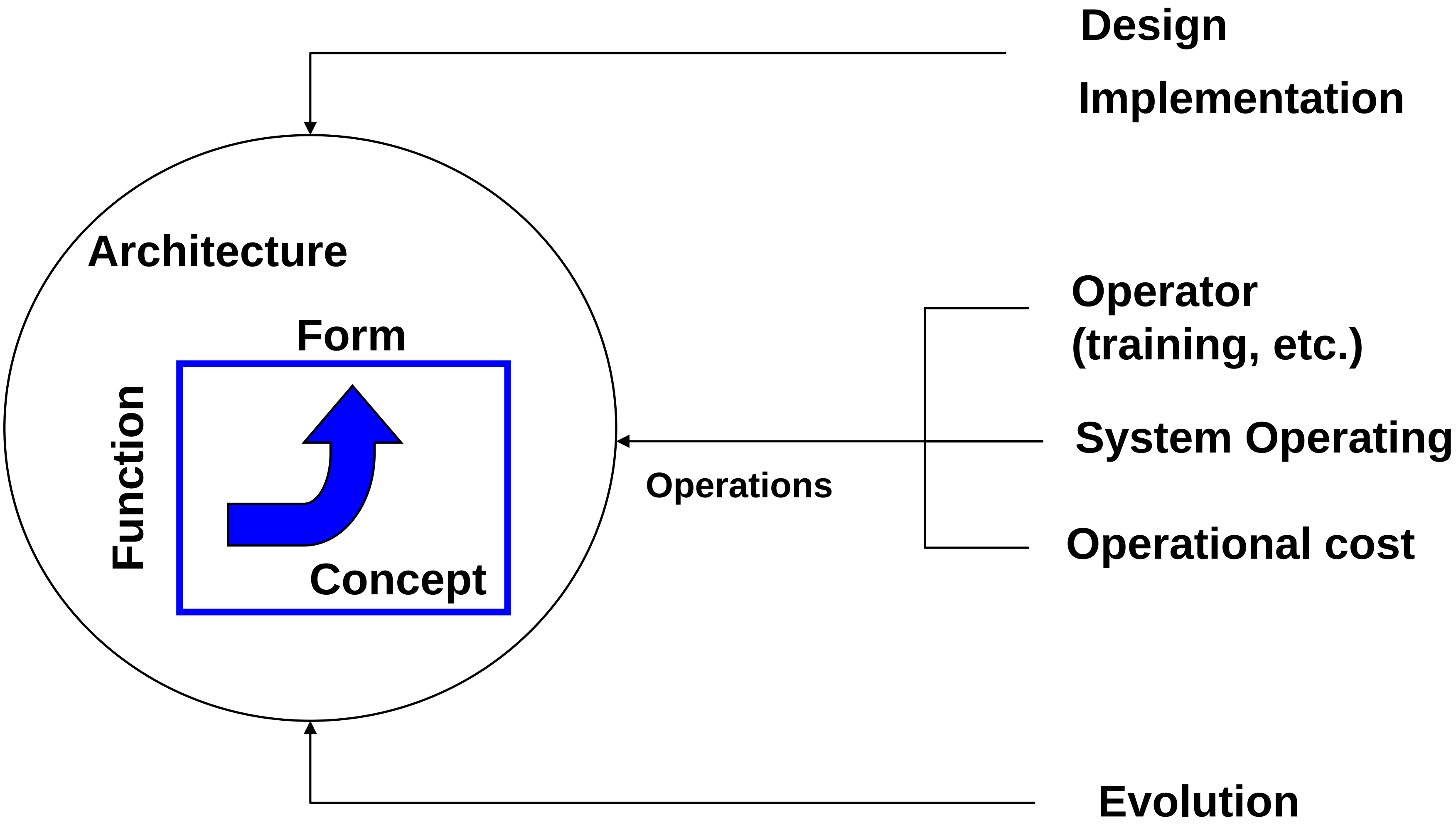


Think of yourself as here today

Principle Downstream Influences

- Operational influences of the delivered system/product :
  - The system operating and associated timing
    - Operational sequence
    - Dynamic behavior and transitions
  - How humans interact with the system as operators
  - The operational cost of the system
- How the system is realized:
  - Design
  - Implementation
- How the system is envisioned to evolve

Framework for Downstream Influences



Design

Implementation

Architecture

Form

Function

Concept

Operations

Operator
(training, etc.)

System Operating

Operational cost

Evolution

Downstream Impacts on Architecture

- During its development, the product will pass through all the downstream influences (e.g. it will be designed, implemented, operated, etc.)
- However, *while* architecting, these influences must also be considered, in order to ensure that the product will be successful (AFX)
- This is sometimes called design for “ilities” (designablilty, implementability, operability, etc)
- Downstream influences give rise to ambiguity at the time of architecting, which should be reduced/resolved to the extent possible in the architecting process
- As the product passes through the downstream steps, the actual complexity of the system grows

Downstream Ambiguity in Architecting

- **At the time of architecting, there is a great deal of ambiguity regarding these downstream influences, because they are "in the future"**
  - **Will manufacturing be ready?**
  - **How much testing do we have to budget/schedule?**
  - **Are the design tools available/calibrated?**
  - **How do we expect this product to evolve?**
  - **In what condition will it be operated?**
  - **Who will train the operator?**

Anticipating the often ambiguous impact of the downstream influences and planning for their impact is a main role of the architect

Downstream Complexity

- As the product passes through the downstream steps, the actual complexity (as measured through the known and documented information on interactions) grows
  - Design information accumulates
  - Implementation plans evolve
  - Operation, training, maintenance, etc. plans develop
  - Evolution plans begin
- Complexity grows approximately combinatorially with elements and participants

Creating abstractions, hierarchy and decomposition to keep the perceived complexity within bounds for all participants in the process is a main role of the architect

Operator

Operator

- Operator is a product attribute
- *Who* will use/execute the system (including training, performance, motivation, etc.)
- Necessary for products with human agents/operators/supervisors
  - – most important for human-in-loop (e.g. bicycle)
  - – important for direct human operation (e.g. lathe, wheelchair, GUI)
  - – for other products, can be considered part of interface/constraints (e.g. human factors design, industrial design)
- Because of the unique issues of human performance and safety, it is useful to keep separate as an additional attribute

Operations Cost

Operating cost

- **Operations Cost is a product attribute**
- ***How much*** **it will cost to operate the system**
- **This is the** ***recurring*** **operational related costs**
  - **Operator and other personnel**
  - **Training**
  - **Maintenance and (nominal) upgrades**
  - **Consumables**
  - **Indirect operating costs (insurance, etc.)**

System Operating - Timing

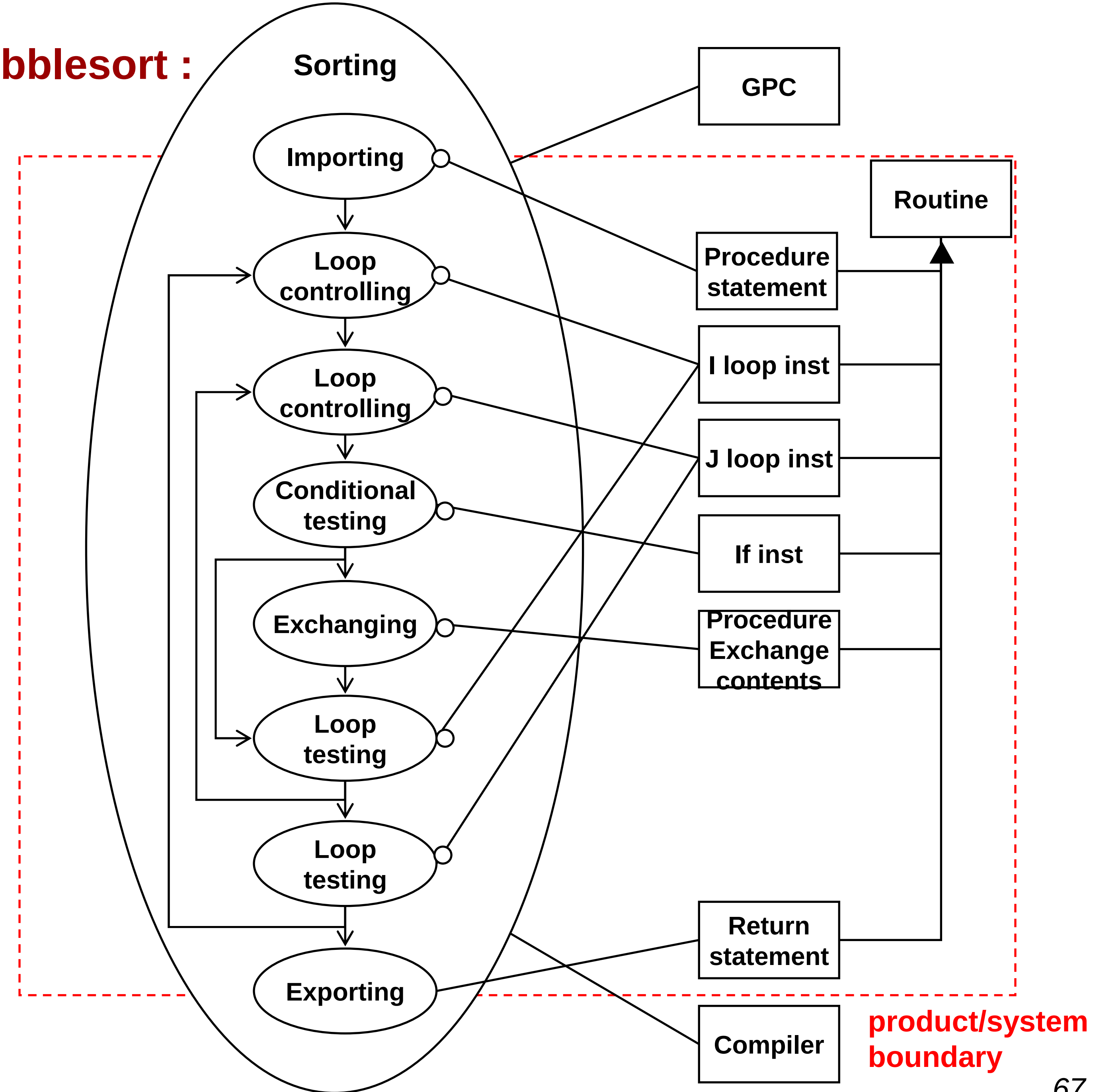
System Operating

- __Timing__ is a product attribute
- *When* the system operates, the *time* sequence of events
- Has two important aspects
  - Operational sequence
  - Dynamic behavior
- __Operations sequence__ is the *total set* of steps or processes that the system undergoes, inclusive of the primary process for which it is intended
- __Dynamic behavior__ is the detailed timing of steps, their sequence, start time, duration, overlap, etc.

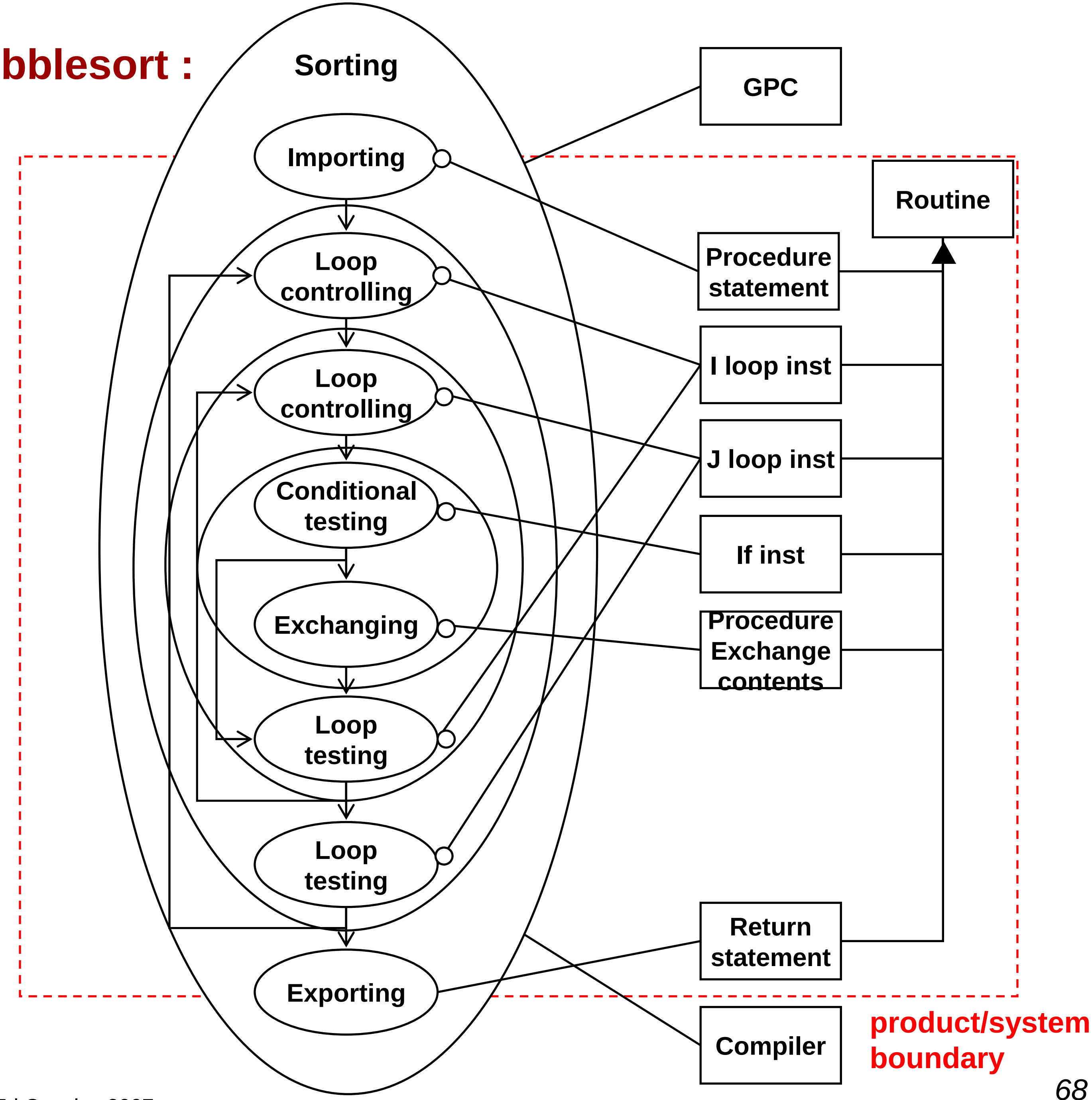
Operating Sequence

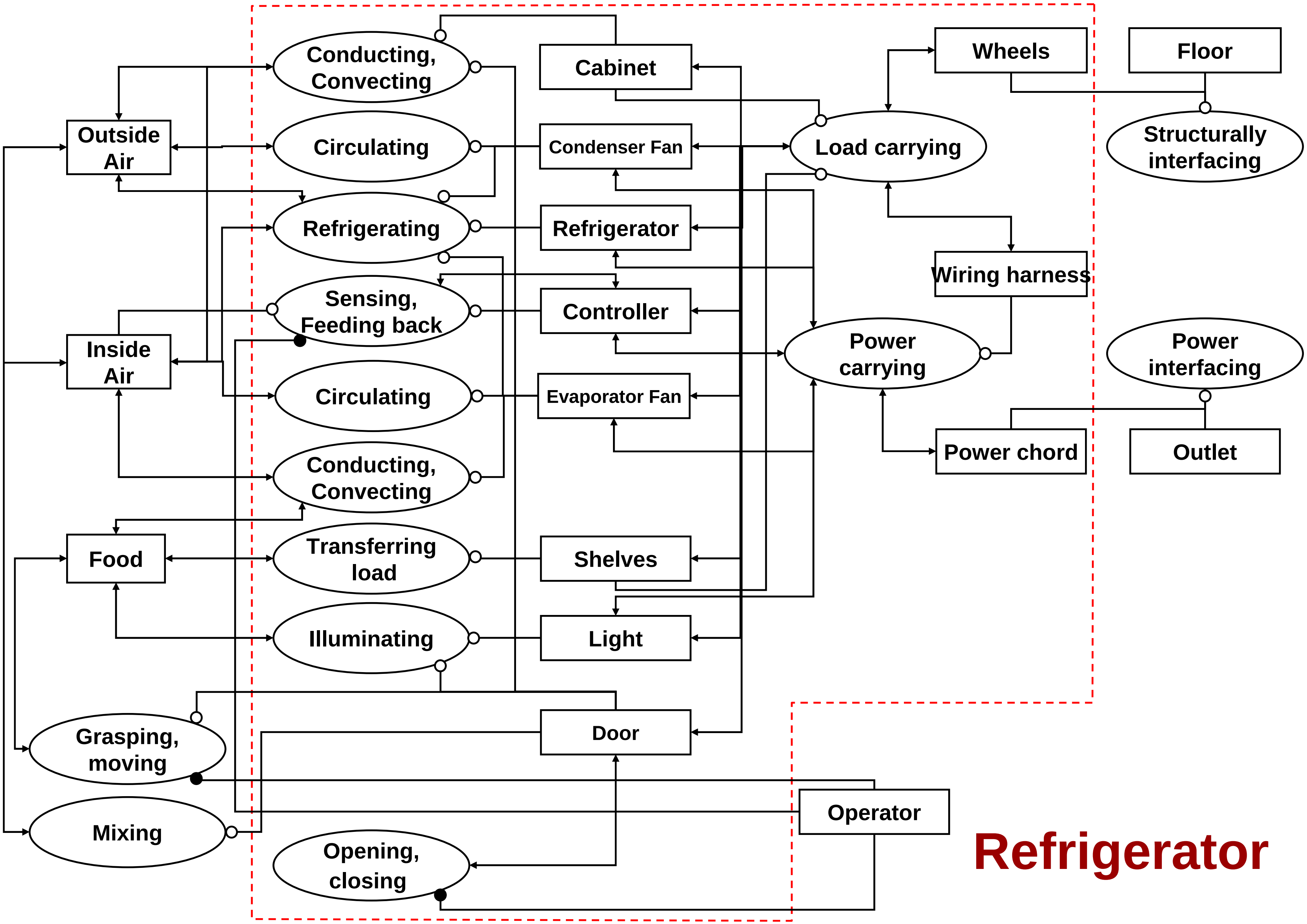
- **Except for static and very simple product/systems, it is important to represent the <u>sequence</u> of events that are executed by the system to deliver its function**
- **There are various ways to represent sequence:**
  - **Invocation arrows on OPM**
  - **Several OPMs or animation**
  - **Sequence lines**
  - **Flow control diagrams**
  - **State transition diagrams**
  - **Algebra of sequence**
- **Focus first on the sequences associated with the delivery of primary value and other value processes**

Software Code Bubblesort : Process - Object Architecture

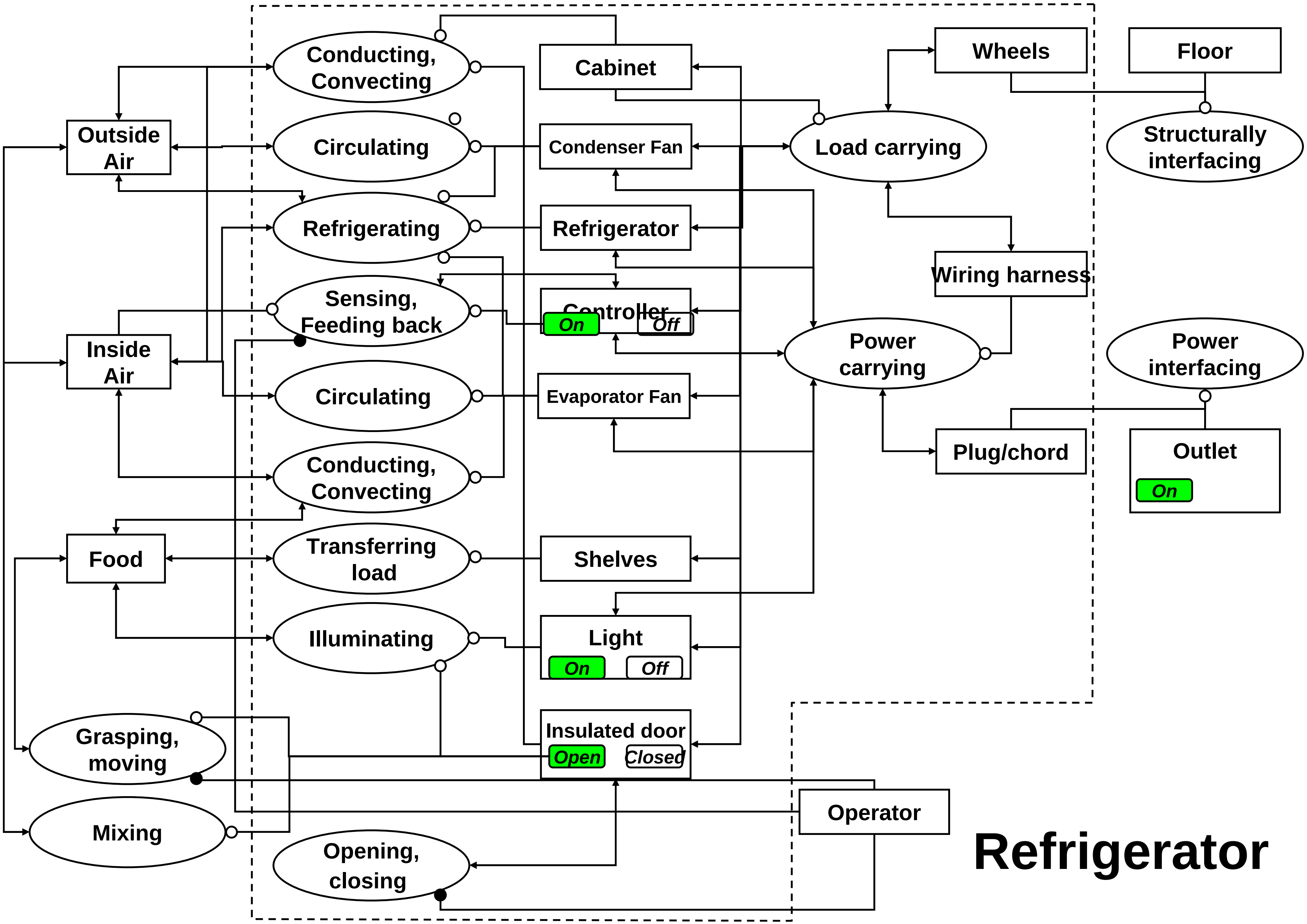


Software Code Bubblesort : Process - Object Architecture

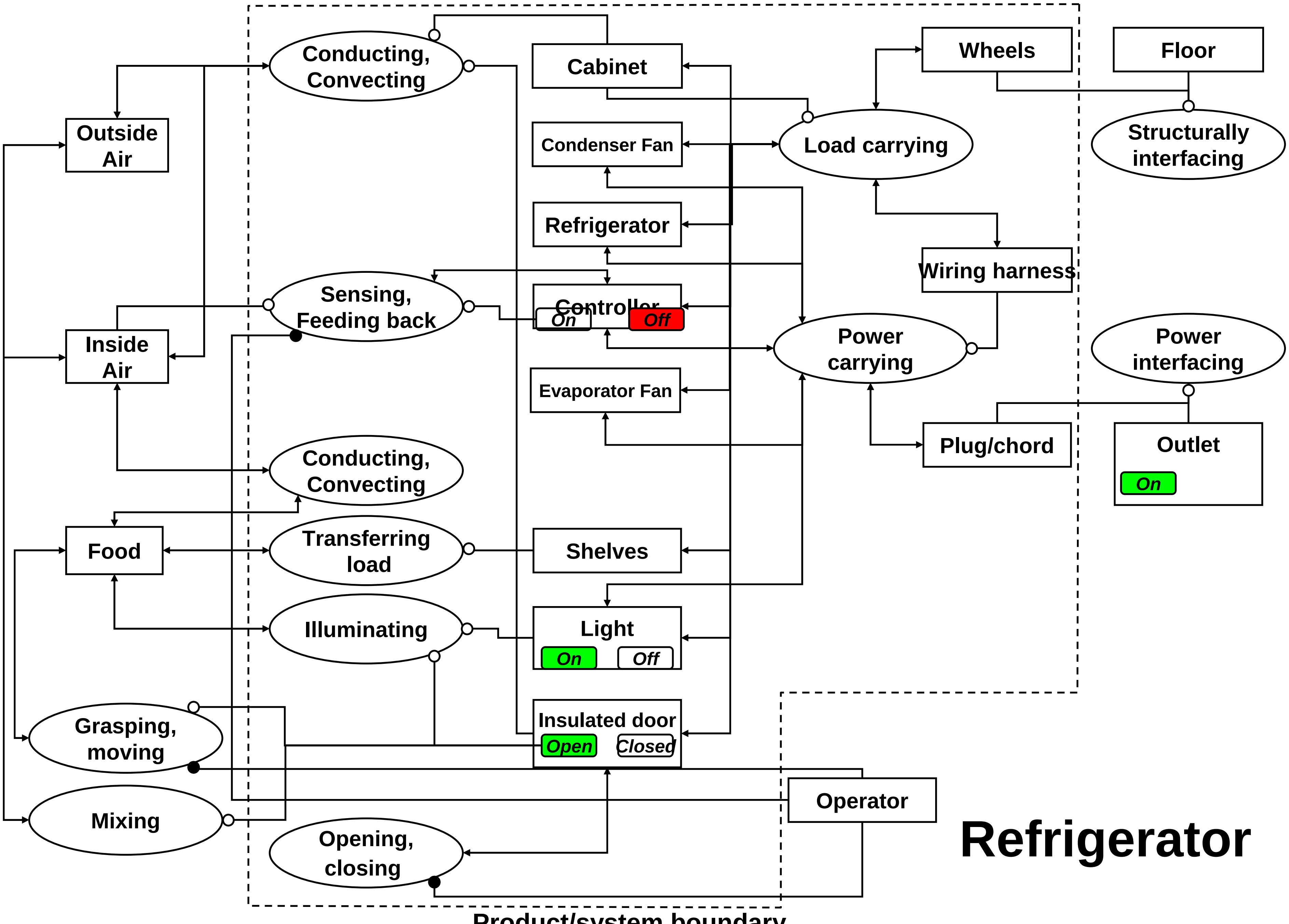




Refrigerator

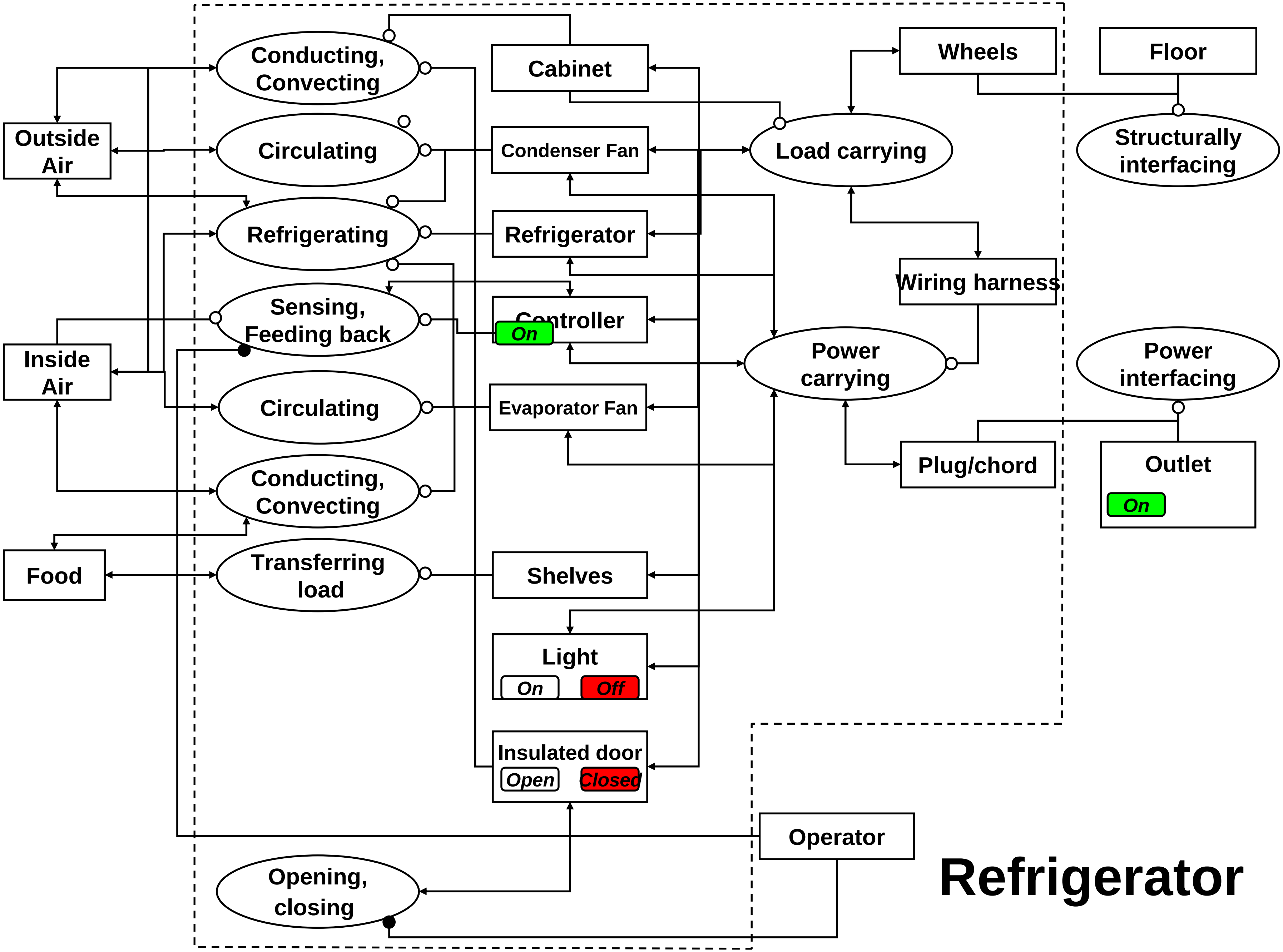


Refrigerator



Refrigerator

Refrigerator

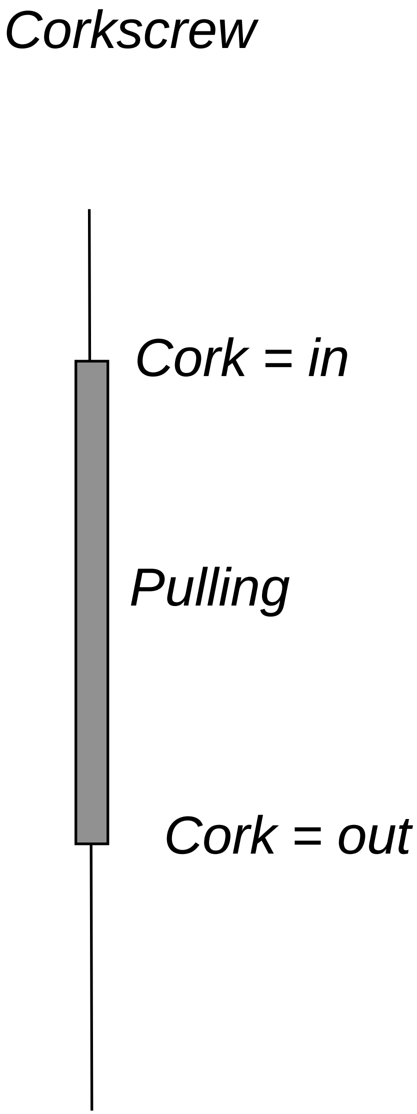


Outside Air
Inside Air
Food
Conducting, Convecting
Circulating
Refrigerating
Sensing, Feeding back
Circulating
Conducting, Convecting
Transferring load
Opening, closing
Cabinet
Condenser Fan
Refrigerator
Controller
On
Evaporator Fan
Shelves
Light
On
Off
Insulated door
Open
Closed
Wheels
Floor
Load carrying
Structurally interfacing
Wiring harness
Power carrying
Power interfacing
Plug/chord
Outlet
On
Operator

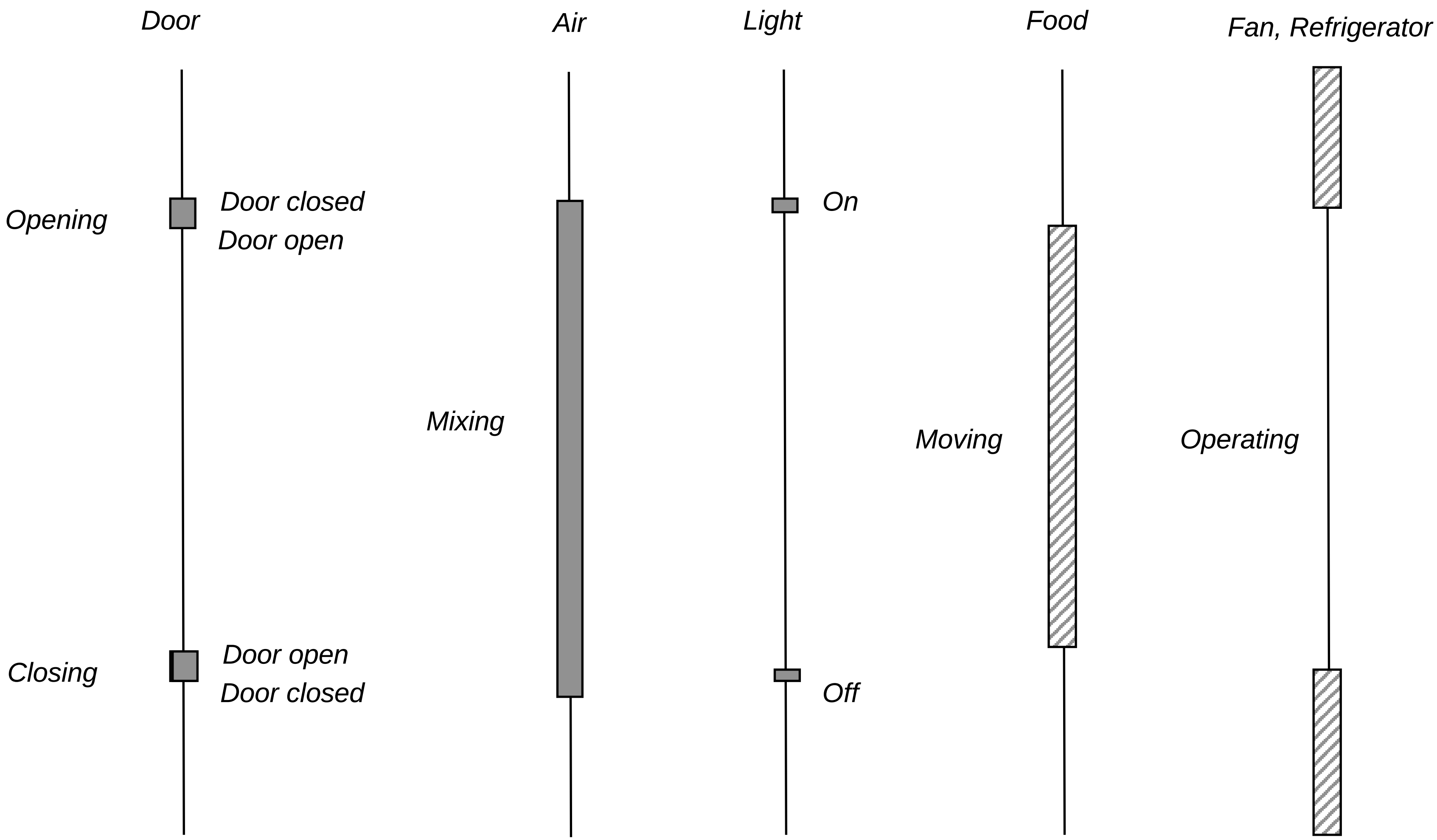
Product/system boundary

“Sequence” Lines

- **Sometimes convenient to represent sequence on vertical or horizontal line, and label events and changes in state**
- **Can do this for one object or several**

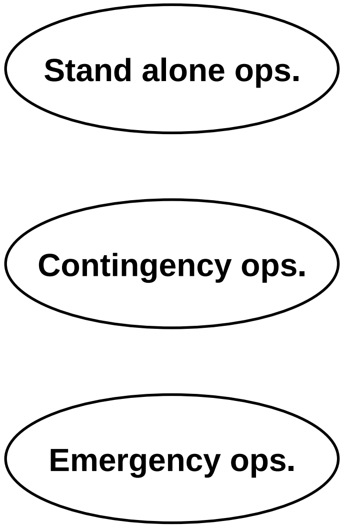


Refrigerator



Operational Sequence Steps

- Executing primary process is a relatively confined view, with no sense of the overall *set of steps* necessary to get it going, take it down, store, maintain and operate in other than the nominal mode
- Stand Alone Operations are operations where the object must operate without connection to its normal system (e.g. backing a trailer by hand, testing a piece of code)
- Contingency Operations are operations other than normal which might reasonably be encountered, and from which recovery is necessary without loss of primary function, personal or property damage (e.g. a spill, noise on a transmission line)
- Emergency Operations are operations other than normal which are outside the contingency window, and when primary function will not be executed, some property damage will be allowed, but personal loss is to be avoided (e.g. a crash, loss of a transmission circuit)



Stand alone ops.

Contingency ops.

Emergency ops.

Commissioning, Maintaining, Decommissioning

- **Commissioning includes:**
  - Waiting in storage, shipping, etc.
  - Getting the system installed, and ready in principle to execute - retrieving, connecting, powering, initializing
  - Loading or positioning, set to execute
- **Decommissioning is the reverse**
- **Maintaining includes:**
  - Inspecting, calibrating
  - Repairing, maintaining, overhauling
  - Updating

Waiting in storage

Retrieving, connecting, powering-up, initializing

Loading, positioning

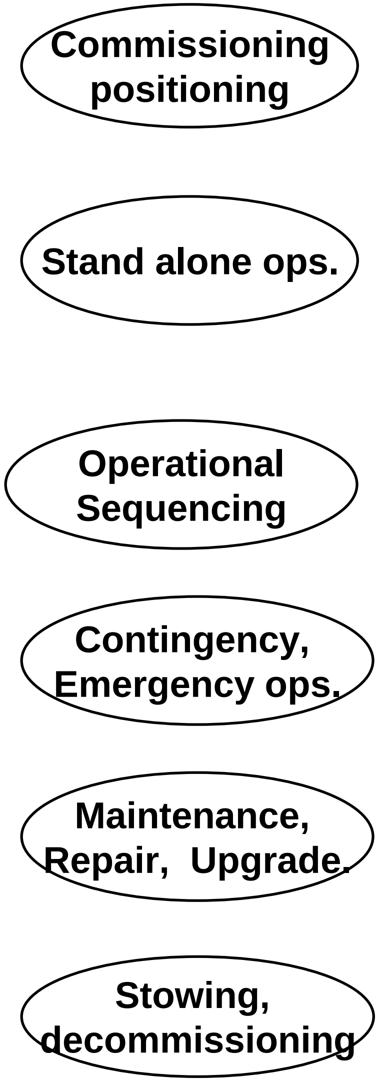
Archiving, unloading

Terminating, disconnecting, depowering, storing

Inspecting, repairing, calibrating, updating, maintaining

Overview Operations

- Commissioning and positioning operations are necessary to get the system from the point of implementation delivery to the point of operation (e.g. transport, storage, get ready, get set)
- Stand alone operations are those that do not involve interactions with other parts of the whole project system usually involved in operations (e.g. check out, trouble shooting)
- Operational sequence is the sequence and if necessary timing of the steps in actual value delivery operation
- Contingency operations are those from which you expect a graceful and continued operation, at perhaps less than maximum value delivery, but no loss of life or property. Emergency operation abandon all value delivery and seek to preserve life and property
- Maintenance is the planned activities, repair, the unplanned activities, upgrade the in place improvement
- Stowing and decommissioning is the set of end of service interval steps necessary to make the system ready again, or retire it



Commissioning positioning

Stand alone ops.

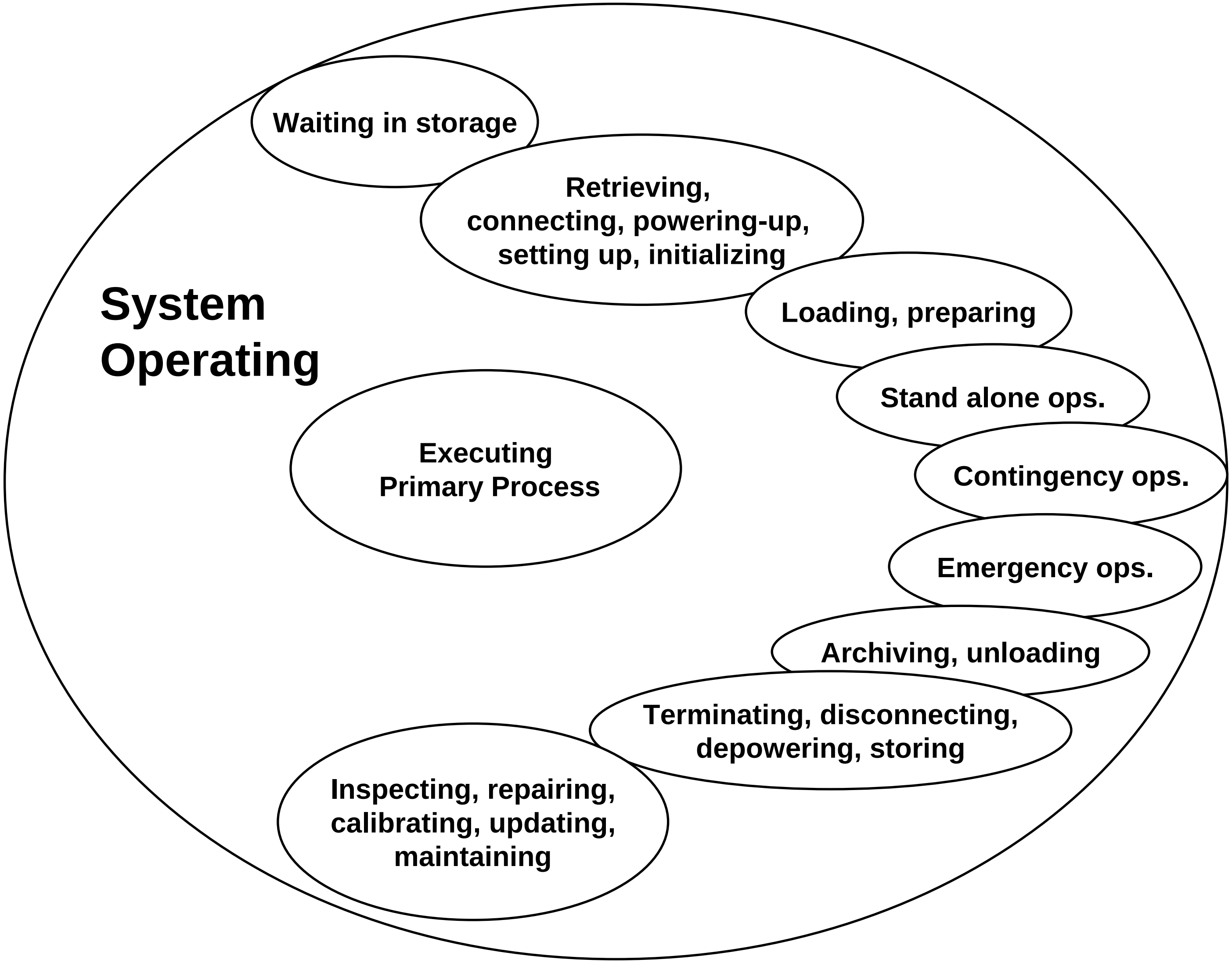
Operational Sequencing

Contingency, Emergency ops.

Maintenance, Repair, Upgrade.

Stowing, decommissioning

Operational Sequence - All Processes



System Operating

Waiting in storage

Retrieving, connecting, powering-up, setting up, initializing

Loading, preparing

Executing Primary Process

Stand alone ops.

Contingency ops.

Emergency ops.

Archiving, unloading

Terminating, disconnecting, depowering, storing

Inspecting, repairing, calibrating, updating, maintaining

Store

Get ready

Get set

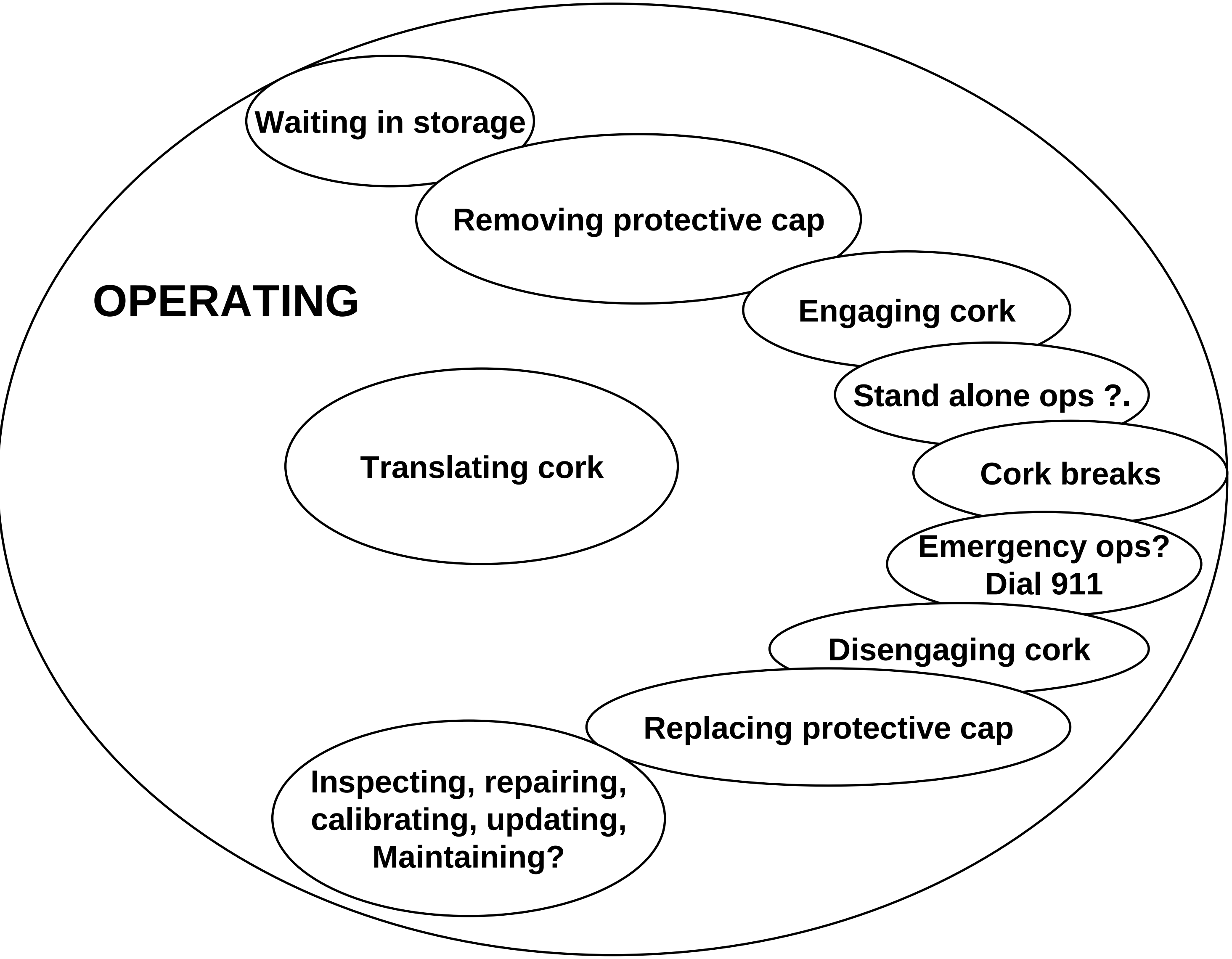
Go
- **Nominal**
- **Stand alone**
- **Contingent**
- **Emergency**

Get unset

Get unready

Fix

Operational Sequence - Cork Translator



OPERATING

Waiting in storage

Removing protective cap

Engaging cork

Stand alone ops ?.

Cork breaks

Emergency ops?
Dial 911

Disengaging cork

Replacing protective cap

Translating cork

Inspecting, repairing, calibrating, updating, Maintaining?

Store

Get ready

Get set

Go { Nominal
Stand alone
Contingent
Emergency

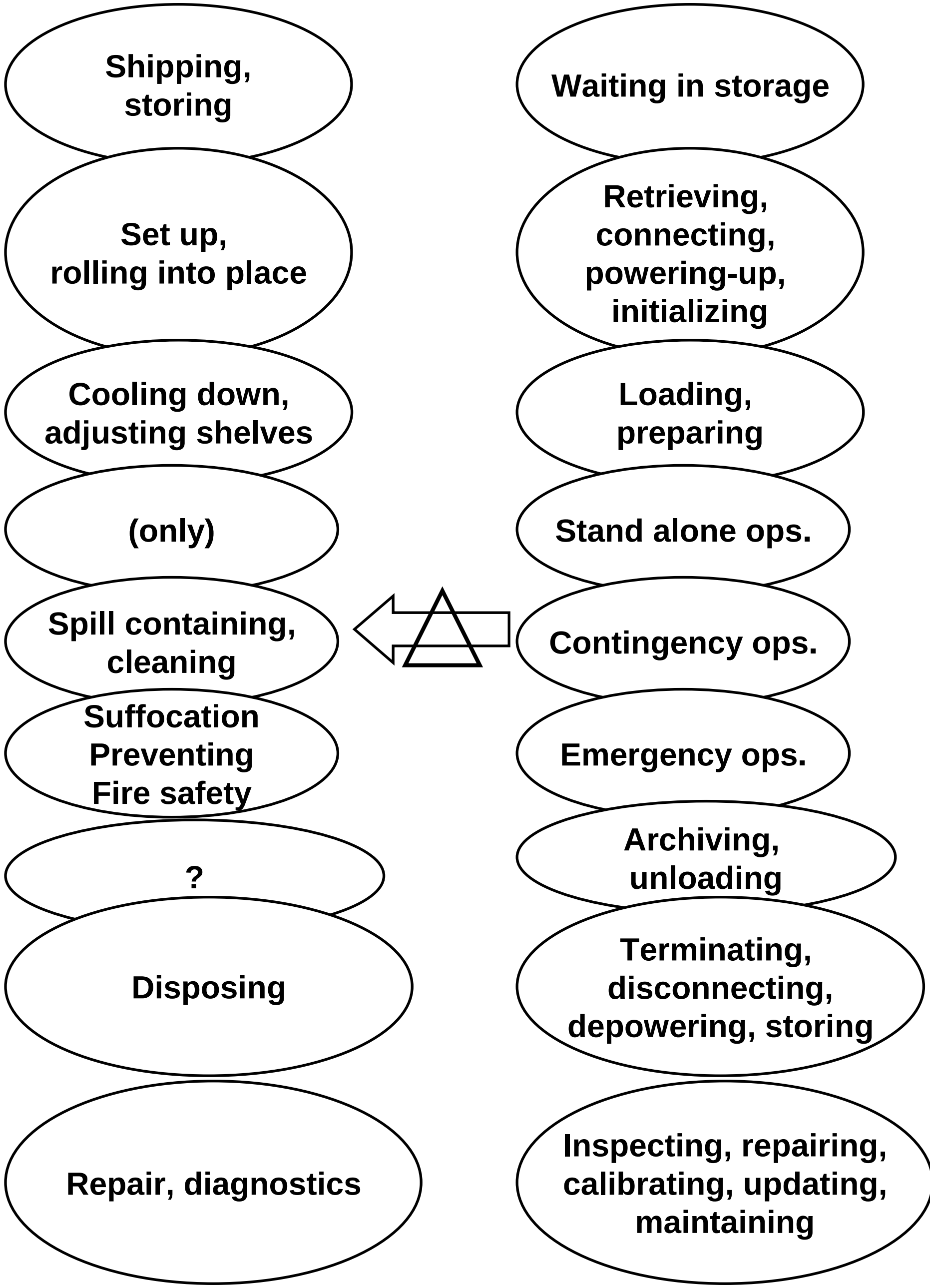
Get unset

Get unready

Fix

Refrigerator - Operational Sequence

What influence with these processes have on the architecture?



Shipping, storing

Set up, rolling into place

Cooling down, adjusting shelves

Opportunity? (only)

Opportunity? Spill containing, cleaning

Suffocation Preventing Fire safety

?

Disposing

Opportunity? Repair, diagnostics

Waiting in storage

Retrieving, connecting, powering-up, initializing

Loading, preparing

Stand alone ops.

Contingency ops.

Emergency ops.

Archiving, unloading

Terminating, disconnecting, depowering, storing

Inspecting, repairing, calibrating, updating, maintaining

Dynamic Behavior - Timing

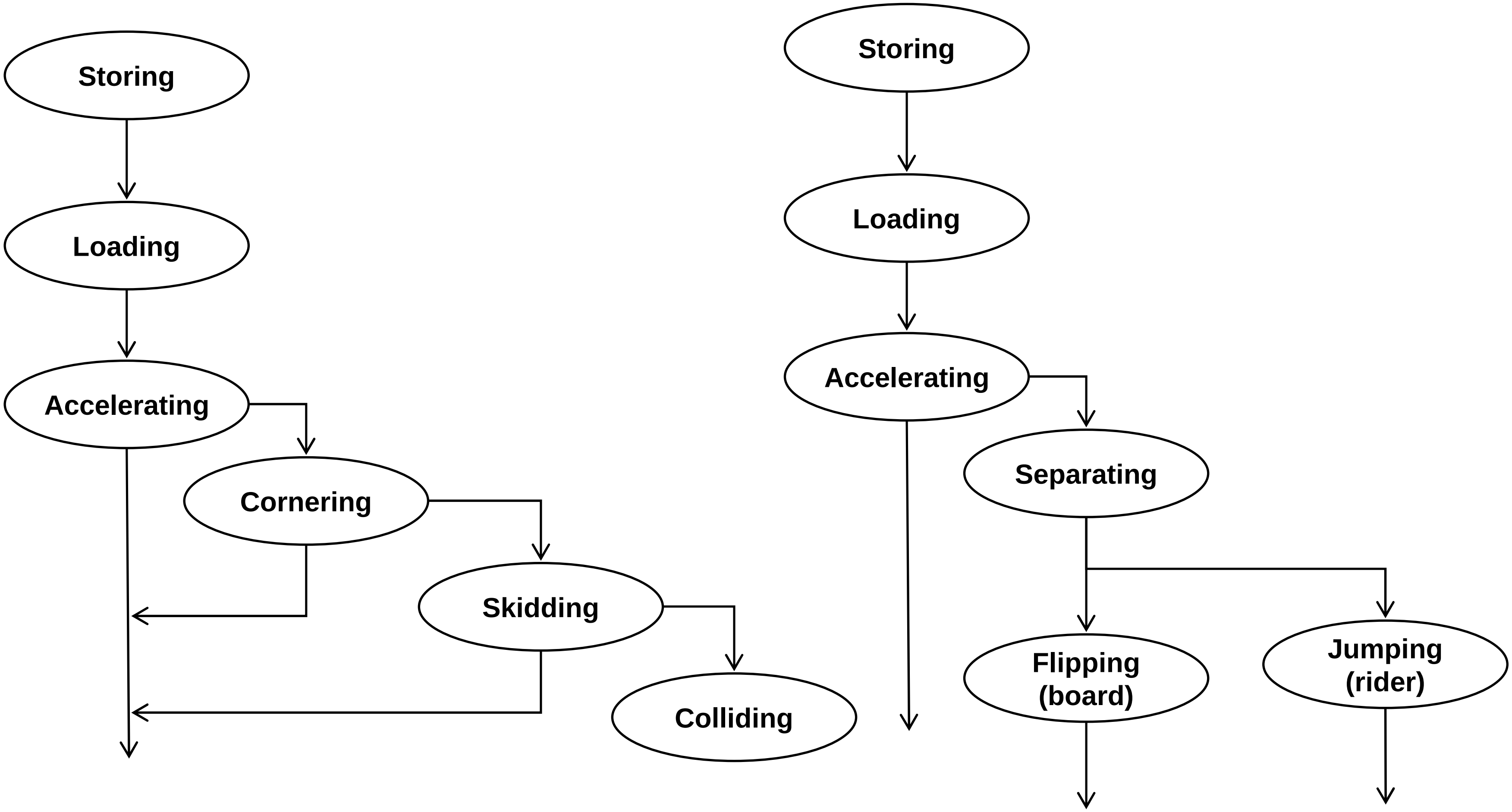
- **Detailed knowledge of the timing of processes and associated state changes is important for “dynamic” systems whose functionality is time dependent or real time**
  - **Start-up transients**
  - **Multiple parallel processes**
  - **Latency**
  - **Timing constraints (e.g. schedule)**
- **Examples in “medium” systems?**

Dynamic Behavior (cont.)

- In both HW and SW, relatively simple if there is a single thread/string of events.
- In hardware, there is often the potential of simultaneous events, which could interact and/or have critical timing (e.g. skidding while driving, terminal guidance)
- In software, even more complex
  - Run time environment
  - Process string/thread management
  - Indetermination of timing because of interaction with other applications or operating systems
  - Interrupts, etc.

State Transitions - Skateboard

Board



What is the Sequence of Operations?

- Is it single string?
- Or multiple string?

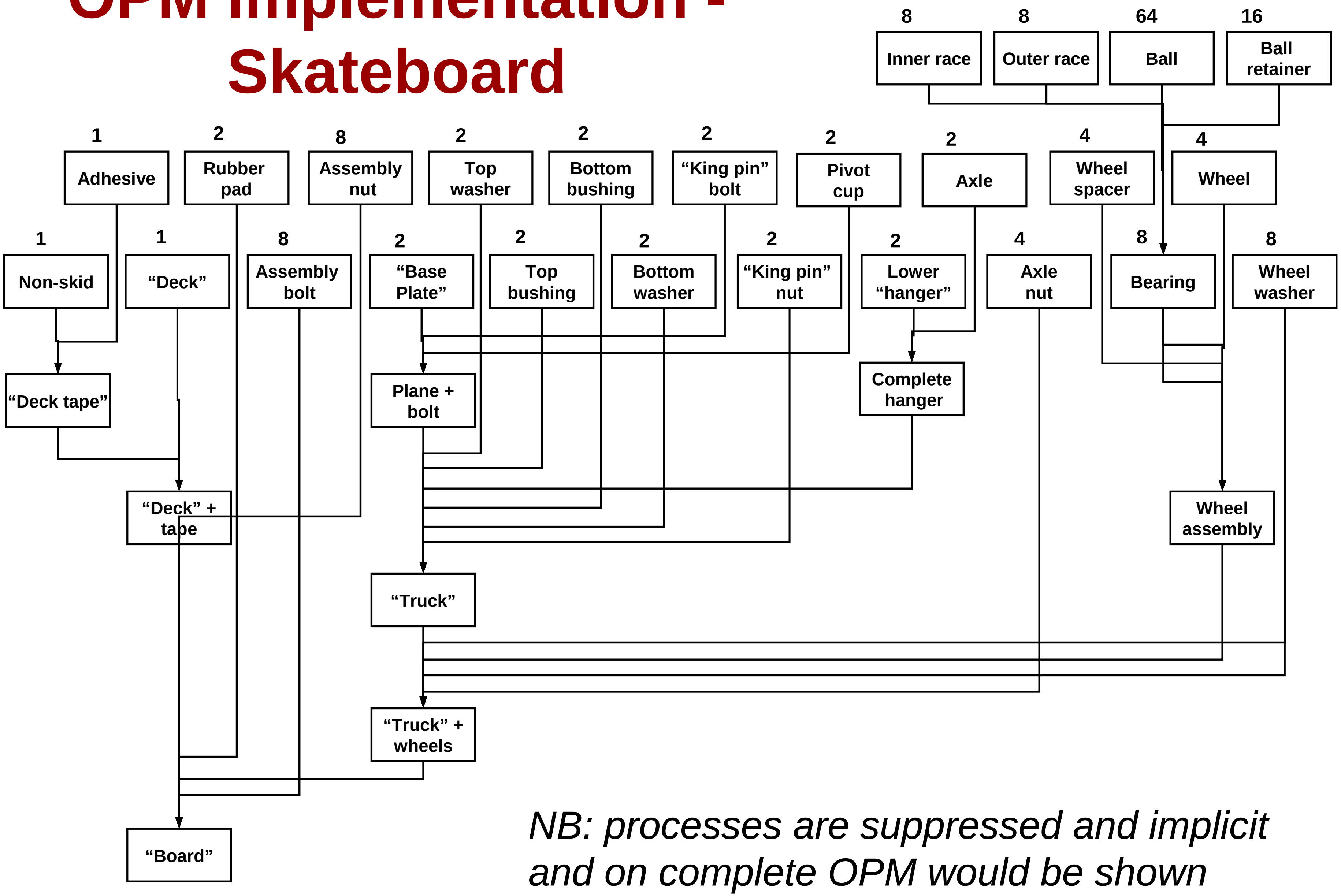
Other Downstream Processes

- The system must be *designed*, so it must be architected in such a way that design can proceed smoothly and efficiently
- The system must be *implemented*, so it must be architected for manufacturability, coding, integration, test, and verification
- The system may *evolve* and be updated, so it must be architected with a view towards these changes - Evolution is really just a *recursive pass* through conception, design and implementation
- Each has its own who, what, where, when, ...

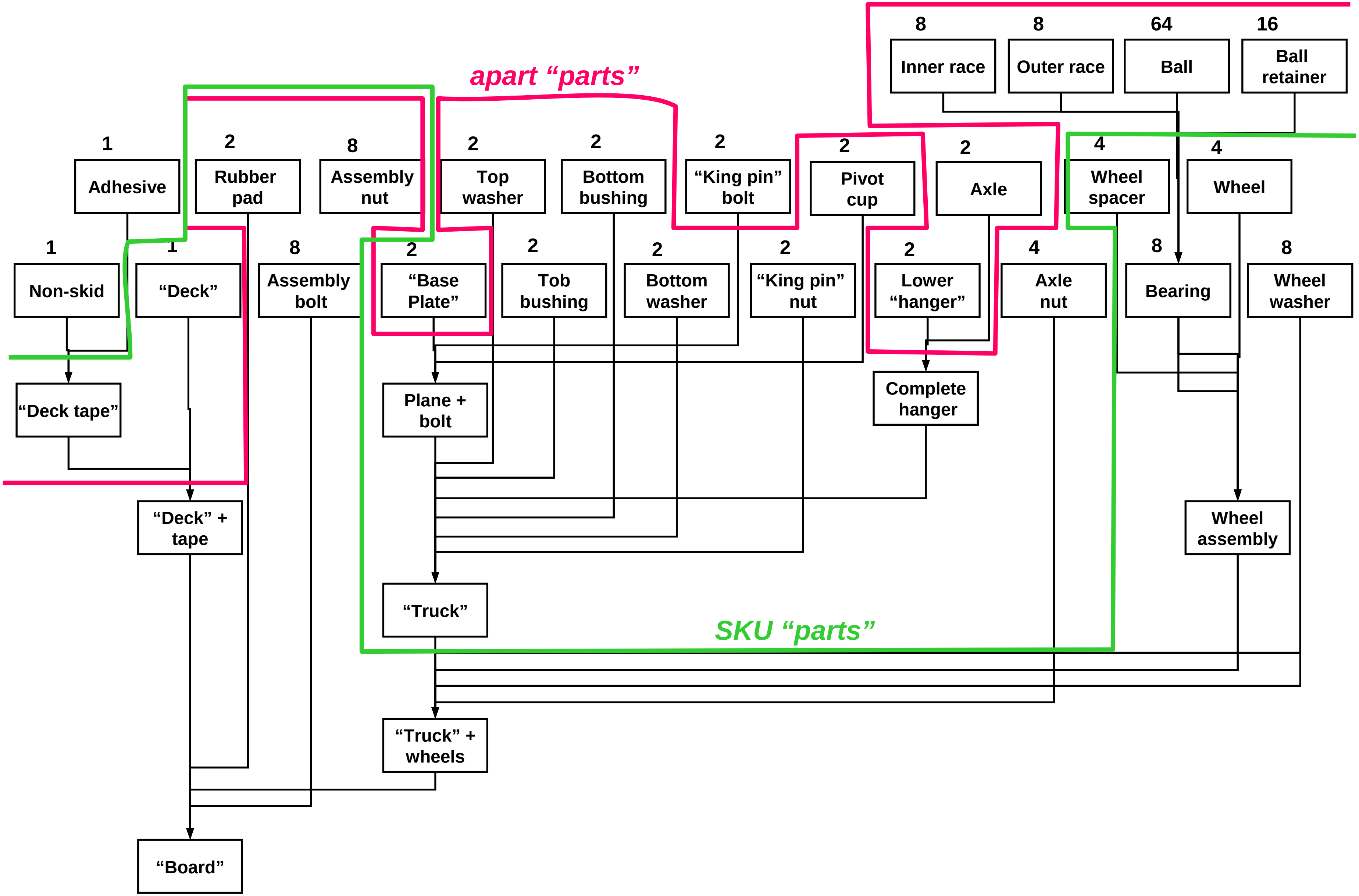
Implementation

- Implementation is the generic term for transitioning the product/system from a design to the deliverable object
  - Creation of elements - manufacturing, crafting, coding
  - Integration - assembly, compilation and system build
  - Testing - verification and validation
  - Preparation for delivery
- Implementation involves objects and processes
- Can be modeled with OPM as well

OPM Implementation - Skateboard



NB: processes are suppressed and implicit and on complete OPM would be shown



apart "parts"

SKU "parts"

8 Inner race

8 Outer race

64 Ball

16 Ball retainer

1 Adhesive

2 Rubber pad

8 Assembly nut

2 Top washer

2 Bottom bushing

2 "King pin" bolt

2 Pivot cup

2 Axle

4 Wheel spacer

4 Wheel

1 Non-skid

1 "Deck"

8 Assembly bolt

2 "Base Plate"

2 Tob bushing

2 Bottom washer

2 "King pin" nut

2 Lower "hanger"

4 Axle nut

8 Bearing

8 Wheel washer

"Deck tape"

Plane + bolt

Complete hanger

"Deck" + tape

Wheel assembly

"Truck"

"Truck" + wheels

"Board"

Concepts and Architectures in Information Systems

- In information and software enabled systems, concepts and their development into architectures are captured in different ways at various levels
- Low level are *algorithms* and their implementation, e.g. bubblesort
- Application domain software are *patterns*, e.g. bridge
- Higher level application domain software is more classical allocation of functionality to modules and routines, and definition of interfaces
- In network software, concepts and architectures are captured in protocols, such as Transport Command Protocol (TCP), and at a higher level, the entire architecture of the XXX seven layer model

OSI Model

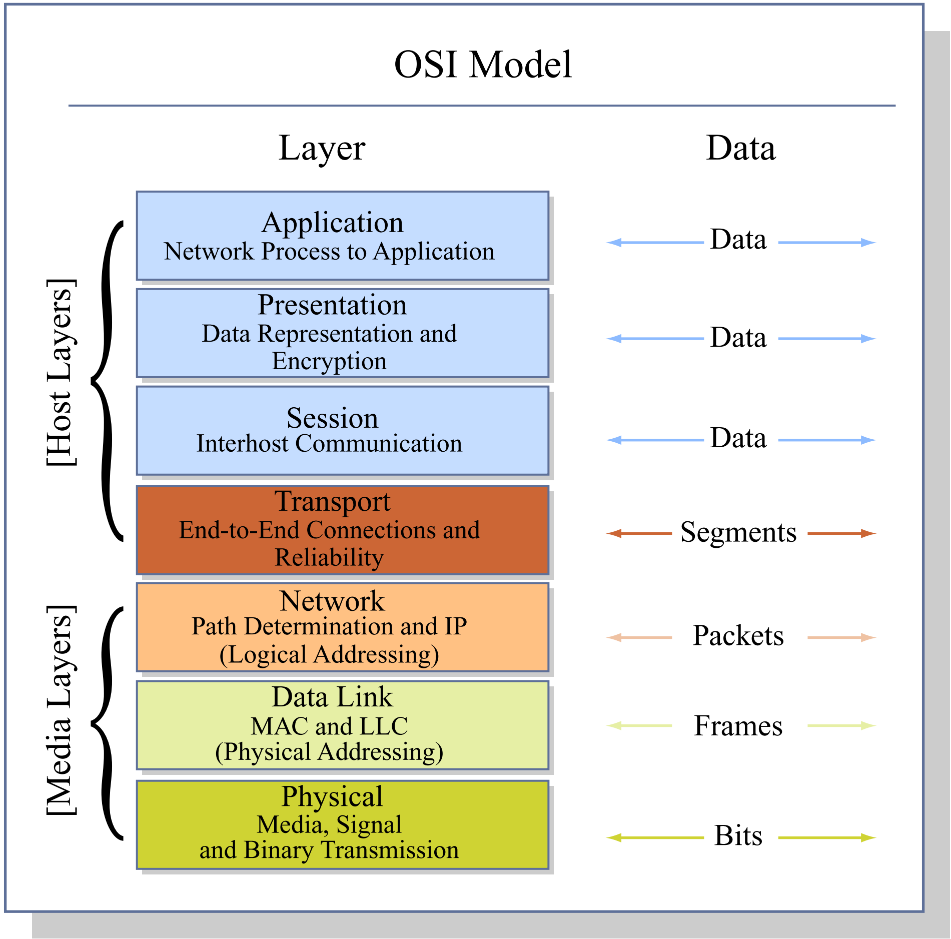

OSI Model
Layer
Data
[Host Layers]
[Media Layers]
Application
Network Process to Application
Presentation
Data Representation and Encryption
Session
Interhost Communication
Transport
End-to-End Connections and Reliability
Network
Path Determination and IP (Logical Addressing)
Data Link
MAC and LLC (Physical Addressing)
Physical
Media, Signal and Binary Transmission
Data
Data
Data
Segments
Packets
Frames
Bits

Figure by MIT OCW.

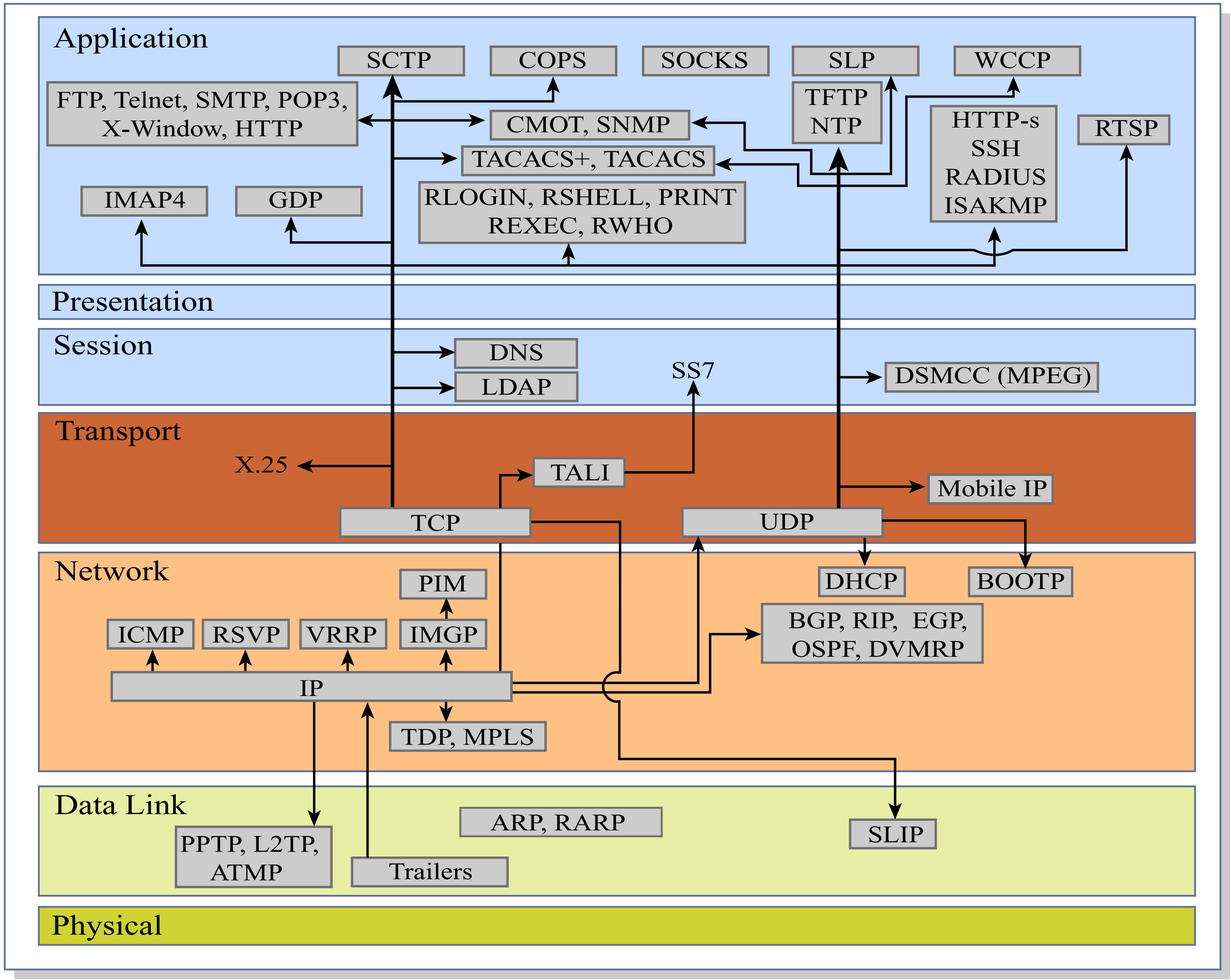
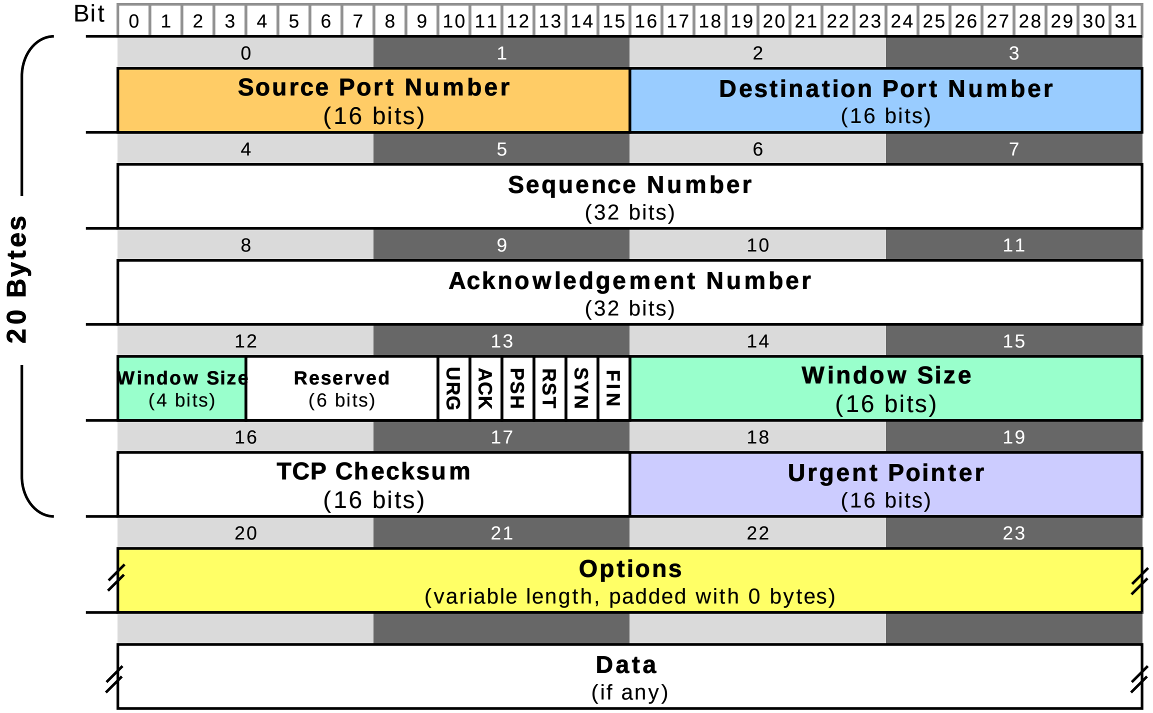


Figure by MIT OCW.

TCP Header

RFC 793 – Transmission Control Protocol



Bit 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

20 Bytes

0 1 2 3

Source Port Number (16 bits)

Destination Port Number (16 bits)

4 5 6 7

Sequence Number (32 bits)

8 9 10 11

Acknowledgement Number (32 bits)

12 13 14 15

Window Size (4 bits)

Reserved (6 bits)

URG ACK PSH RST SYN FIN

Window Size (16 bits)

16 17 18 19

TCP Checksum (16 bits)

Urgent Pointer (16 bits)

20 21 22 23

Options (variable length, padded with 0 bytes)

Data (if any)

Figure by MIT OCW.

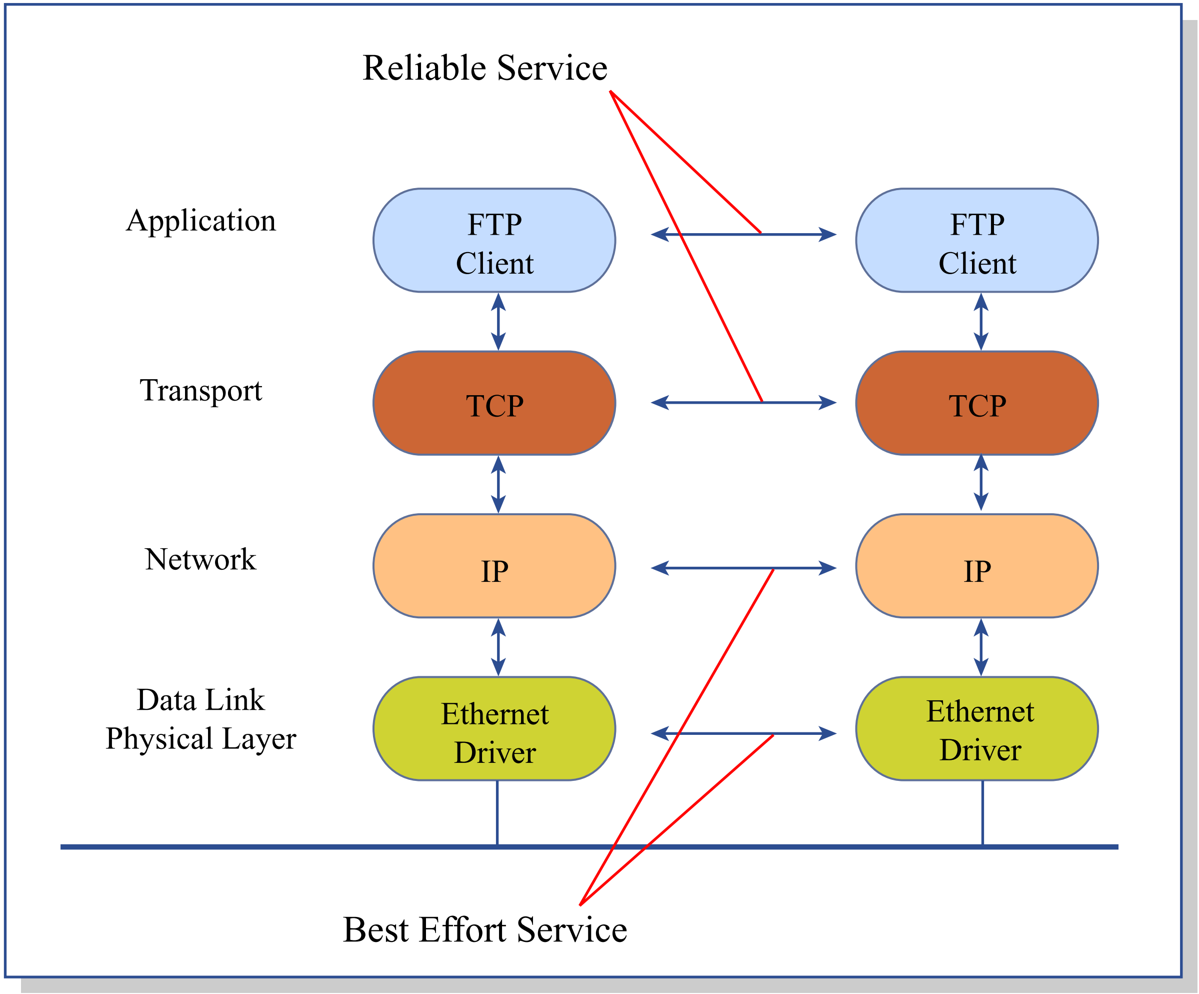


Figure by MIT OCW.

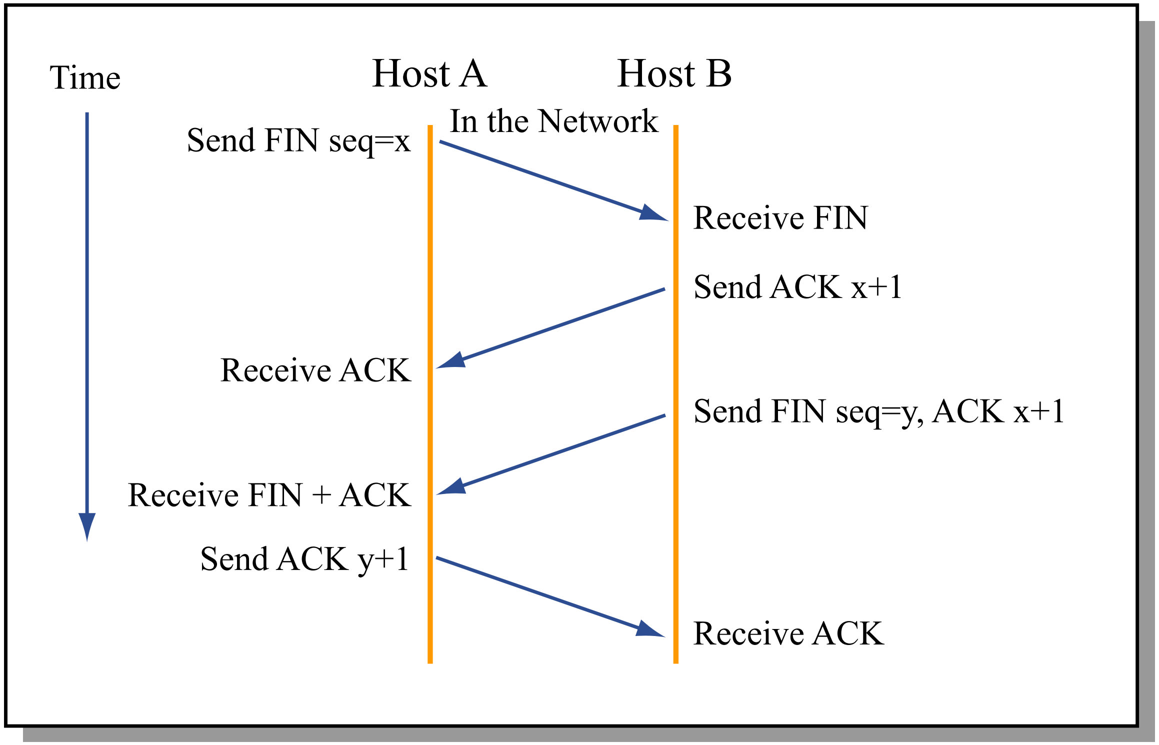


Figure by MIT OCW.

TCP Connection Establishment

How Would You Analyze this “Architecture”?