Model-Based Design Structure Matrix

Dov Dori
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Technion City, Haifa 32000
Israel
dori@ie.technion.ac.il
And Engineering Systems Division
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139, USA
dori@mit.edu

Abstract.

Introduction

Design Structure Matrix (DSM) is an accepted method for enhancing and analyzing design of products and systems. The use of matrices in system modeling can be traced back to Warfield in the 70's and Steward in the 80's. In the 1990s the method received attention and wide spread.

Graphs have been used for system modeling since the early 1970's. Traditionally, the system graph has been constructed by allowing a node to be one of two types of system elements: either an object (or a part, or a component), when the structure or the system was modeled, or a process (or an activity, or a task), when the work of constructing the system was considered, as in a work breakdown structure (WBS) or a critical path method (CPM) graph of a project model, or a specification of the dynamics of the product itself after it has been designed, manufactured, and delivered. Based on this observation, system-describing graphs can be categorized into two types based on the type of element represented by the node in the graph: *object-based system graphs* and *process-based system graphs*.

Recent years have witnessed the emergence of models as a basis for synthesis, communication, development, and analysis of existing and new systems and product. Indeed, model-based systems engineering (MBSE) is a rapidly growing field of research and application. However, relationships between a model and a corresponding DSM of the same system have not been thoroughly investigated. This paper looks into the relationships between a system model and its DSM and features that can be studied by examining these relationships.

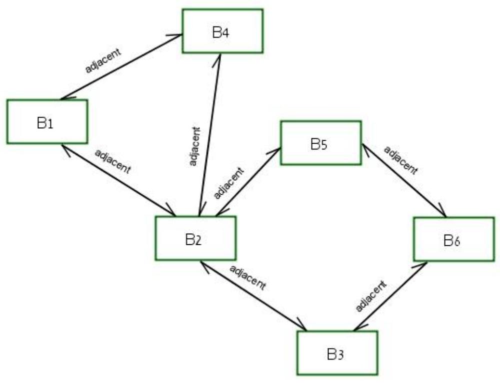
The modeling language we use in this paper is Object-Process Methodology (OPM). OPM takes a fresh look at modeling complex systems, both natural and artificial, where artificial ones might comprise such components as humans, physical objects, hardware, software, regulations, and information. OPM is a formal and intuitive paradigm for systems architecting, engineering, development, lifecycle support, and evolution. It caters to people’s intuition and natural train of thought. As its name suggests, the two basic building blocks in OPM are (stateful) objects—things that exist (at some state), and processes—things that transform objects by creating or destroying them, or by changing their state. In this paper we do not relate to states explicitly. However, any stateful object with *k* states can be modeled as *k* distinct stateless objects. For example, Water with the three states cold, warm, and hot can be modeled as the three objects Cold Water, Warm Water, and Hot Water, with the processes Heating and Cooling transforming any one of these three objects to any other one of these three objects.

Directed vs. Undirected DSM

Joining two nodes in a system graph represents the existence of some relationship between two system elements in the system graph. For an object-based system graph, this link might denote a physical connection or proximity relation between the two connected parts, such as an attachment of the engine to the gearbox in an automobile power train. If object B1 and object B2 are connected, then the object-based system graph would contain an edge between B1 and B2.

Whether the edges in the system graph are directed or not depends on the semantics of the relationship, which determine its symmetry attribute value. If the relationship is symmetric then the edges will be non-directed, while if the relationship is asymmetric then the edges will be directed. To realize why this is so, consider the case of an object-oriented system graph in which the semantics of the edges is adjacency. Adjacency is a symmetric relation, since if B1 is adjacent to B2, then it is also true that B2 is adjacent to B1. Therefore, the edge between B1 and B2 (and any other edge in this graph, which denotes adjacency), is non-directed. A non-directed edge can be interpreted as a pair of directed links between two nodes, pointing in opposite directions. Hence if a graph contains *n* non-directed links it can be considered as containing 2*n* directed links.

Object-Based Systems Graphs and Their DSM



B1 and B4 are adjacent.
B2 and B1 are adjacent.
B2 and B4 are adjacent.
B3 and B2 are adjacent.
B3 and B6 are adjacent.
B5 and B2 are adjacent.
B6 and B5 are adjacent.

Figure 1. An object-based system graph (top) and its automatically-generated OPL paragraph

Figure 1 (top) depicts an example of an object-based system graph with six components labeled B1 through B6, with links labeled "adjacent". The graph was drawn using OPCAT (Dori… 200x). Based on its semantics, adjacency is a symmetric relation, since if B1 is adjacent to B2 then it is also true that B2 is adjacent to B1. Therefore we have used OPM's bidirectional tagged structural relation with a single label, which is used for denoting symmetric structural relations. The tag (label) we have used is simply "adjacent". Figure 1 (bottom) lists the OPL paragraph which is comprised of seven OPL sentences that were generated automatically by OPCAT in response to the graphic user input that created the graph, where each sentence reflects each one of the seven bidirectional structural links labeled "adjacent". For example, the OPL sentence "B2 and B4 are adjacent." was created in response to drawing the bidirectional structural relation tagged "adjacent" between B2 and B4.

	B1	B2	B3	B4	B5	B6
B1	▨	1		1		
B2	1	▨	1	1	1	
B3		1	▨			1
B4	1	1		▨		
B5		1			▨	1
B6			1		1	▨

Figure 2. The Design Structure Matrix that represents the graph and OPL text of Figure 1

Figure 2 is the DSM (Design Structure Matrix) that represents both the graph and the OPL text of Figure 1. The three representations—the object-based system graph, the OPL paragraph, and the DSM—are completely equivalent. Indeed, given any one of them, the other two can be automatically reconstructed. Due to the symmetry of the adjacency relationship, this DSM is symmetric with respect to the diagonal (grey), implying, for example, that if B2 is adjacent to B3, giving rise to "1" in cell (2, 3) of the DSM, then B3 is adjacent to B2, giving rise to "1" in cell (3, 2) of the DSM. Since we have 7 links in the graph, the matrix has 14 1's. In general, the number of non-blank cells in a DSM of a system graph with a symmetric relation is twice the number of the edges in the graph. As noted, each non-directed edge can be thought of as a pair of opposite directed graph, and this further explains the symmetry of the DSM.

In a non-directional (undirected) graph, which depicts a symmetric relationship, a loop of length $n \geq 3$ is created whenever following a path from some node N we return to the same node N after n steps. Examining the graph in Figure 1 (top), we see that it has two loops, one of length 3 and the other of length 4. For humans, it is indeed easy to identify loops by looking at the graph representation, especially if the loops are relatively short and the layout of the nodes is not obscured. For automation, we need to examine the corresponding DSM. Figure 3 shows the Design Structure Matrix of Figure 2 with rows and columns rearranged to reflect the two loops in the DSM of Figure 1. Examining this DSM, we see clearly the two loops. The first, of length 3, is {B1, B4, B2}, and the second, of length 4, is {B2, B3, B5, B6}. Each loop of length n can be followed by tracing a path of length $2n$ of horizontally, vertically, or diagonally adjacent 1's, making sure to change direction each time. For example, the first loop, of length 3, follows the 6 adjacent 1's at the top-left corner. Each connected node in the graph has at least one 1 in its row and column. The number of 1's is the degree of the node, i.e., how many edges are linked to that node. Each node participating in a loop is of degree 2 at least, since it must have one incoming edge and one outgoing edge that participate in the loop. If the node's degree is 1, the node is a leaf and cannot be part of a loop.

A node participating in *k* loops must have a degree of at least 2*k*. Indeed, the DSM in Figure 3 shows that B2, which participates in 2 loops, has a degree 4.

	B1	B4	B2	B3	B5	B6
B1		1	1			
B4	1		1			
B2	1	1		1	1	
B3			1			1
B5			1			1
B6				1	1	

Figure 3. The Design Structure Matrix of Figure 2 with rows and columns rearranged to reflect the two loops (green and orange) in the DSM of Figure 1

Process-Based Systems Graphs and Their DSM

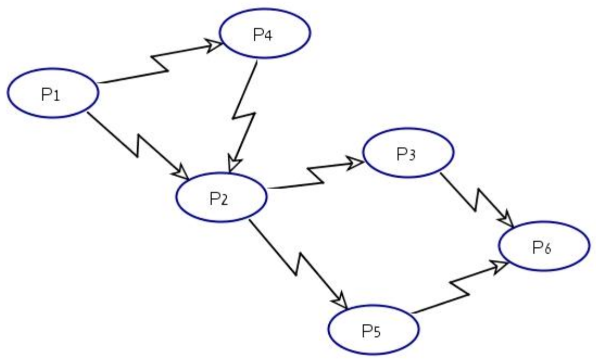
For a process-oriented system graph, a link might denote a temporal or dependency relation between the two connected processes. If process P1 has to be completed before process P2 can start, then P1 and P2 will be linked by an edge in the corresponding process-oriented system graph. The semantics of OPM's invocation link (the lightning shaped arrow) is exactly this. Figure 4 presents a process-based system graph (top) and its automatically-generated OPL paragraph.

Figure 5 is the Design Structure Matrix that represents the graph and OPL text of Figure 4. In a DSM of a digraph, we denote by 1 the matrix entry (A, B) in which there is a direct link from A to B, and by x the matrix entry (B, A) to denote that while B and A are connected, it is not possible to go directly from B to A. For example, since there is a directed edge from P2 to P3, the DSM entry (P2, P3) has a 1 while the DSM entry (P3, P2) has an x.

The difference between the process-based system graph of Figure 4 and the object-based system graph in Figure 1 is that the latter is an undirected graph, while the former is a directed graph (digraph). Hence, while the graphs look similar and the object-based system graph has two cycles, the process-based system graph in Figure 4 has no cycles, because the direction of the edges is such that loops cannot occur. Indeed, examining Figure 5, one cannot find any symmetry. To see what happens when loops do occur in a directed graph, in Figure 6 we have revered the direction of the arrow from P2 to P4, creating a loop {P1, P4, P2}. Figure 7 is the corresponding DSM, and Figure 8 is the same matrix with the rows P1, P4, P2 and columns P1, P4, P2 rearranged to be adjacent in order to visualize the loop.

Examining Figure 8, we can see that what characterizes a loop in a digraph is the alternation of 1's and x's along a closed path (loop) of adjacent non-zero elements.

This closed path is colored green in Figure 8, and indeed following this path, the 1's and X's along it alternate. The orange path is not a loop in the graph, because the 1's and X's along it do not alternate.

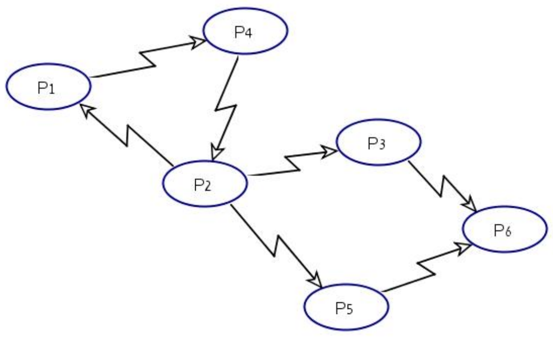


P1 invokes P2 and P4.
P2 invokes P3 and P5.
P3 invokes P6.
P4 invokes P2.
P5 invokes P6.

Figure 4. A process-based system graph (top) and its automatically-generated OPL paragraph

to from	P1	P2	P3	P4	P5	P6
P1	▨	1		1		
P2	x	▨	1	x	1	
P3		x	▨			1
P4	x	1		▨		
P5		x			▨	1
P6			x		x	▨

Figure 5. The Design Structure Matrix that represents the graph and OPL text of Figure 4



P1 invokes P4.
P2 invokes P3, P5, and P1.
P3 invokes P6.
P4 invokes P2.
P5 invokes P6.

Figure 6. The process-based system graph of Figure 5 modified to have one loop (top) and its automatically-generated OPL paragraph

to / from	P1	P2	P3	P4	P5	P6
P1	▨	x		1		
P2	1	▨	1	x	1	
P3		x	▨			1
P4	x	1		▨		
P5		x			▨	1
P6			x		x	▨

Figure 7. The Design Structure Matrix that represents the graph and OPL text of Figure 6

to / from	P1	P4	P2	P3	P5	P6
P1		1	x			
P4	x		1			
P2	1	x		1	1	
P3			x			1
P5			x			1
P6				x	x	

Figure 8. The Design Structure Matrix that represents the graph and OPL text of Figure 6 with rows and columns rearranged to visualize the loop {P1, P4, P2}

Object-Process-Based Systems Graphs and Their DSM

Having studied object- and process-based system graphs and their corresponding DSM's, as well as the characteristics of loops in these types of graphs, we now turn to object-process-based system graphs. In OPM, these are known as Object-Process Diagrams, or OPDs for short. The graphs we have studied until now were also legal OPDs and were indeed drawn by OPCAT (which would not allow constructing illegal OPDs). However, they contained nodes of just one type—either all were objects or all were processes.

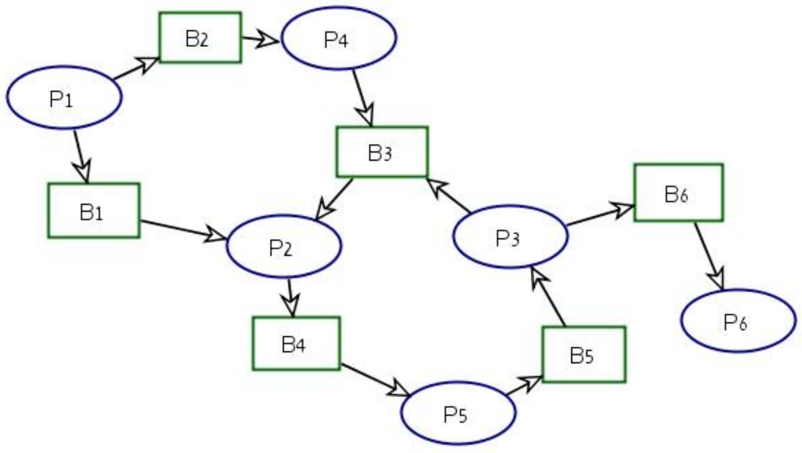
Browning (2001) has suggested four different types of DSM according to the data that the DSM represents: (1) Component-based DSM, for representing relations among components, which are objects in the system, for use in systems engineering and architecting, (2) task-based DSM, for representing relations among tasks or activities, which are processes in the system, for use in project management, (3) parameter-based DSM, which is similar to (2) but at a lower level, and (4) team-based DSM, for interfacing between teams, for use in organizational design and multi-team projects.

The component-based DSM is the DSM of our object-based systems graph, while the task-based DSM is the DSM of our process-based systems graph. Team-based DSM can be extracted from the agent links in an OPM model of a system-delivering project, as explained and demonstrated below.

As we show next, a DSM that combines both component- and task-based DSM can be extracted from an OPM model. In an OPD in general, nodes can be of either type, namely both objects and processes, provided that the procedural links among them (which must be directed) are such that the resulting graph is bi-partite, i.e., any procedural link whose source is an object has a process as its destination and vice versa. Figure 9 is an OPD which demonstrates the fact that it is a bi-partite graph. This OPD uses only two OPM procedural links: consumption link and result link. Both have the same graphic symbol, but a consumption link originates from an object and terminates with a process with the semantics that the process consumes the object, while a result link originates from a process and terminates with an object, with the semantics that the process creates (yields) the object. The OPL in Figure 9 explicates the semantics of these two links. For example, the OPL sentence "P2 consumes B3 and B1." indicates that both B1 and B3 are linked to P2 with a consumption link, and the sentence "P3 yields B3 and B6." indicates that both B3 and B6 are linked to P3 with a result link.

Other procedural links include the enabling links—agent and instrument, and the effect link, which is a bidirectional link expressing state change via the superposition of input state and output state links. In this paper we use the consumption and result links as representatives as links from an object to a process and vice versa, respectively.

The OPD in Figure 9 contains one loop: {P2, B4, P5, B5, P3, B3}. As expected, the type of node in the loop keeps alternating between P (process) and B (object). Moreover, due to the fact that the graph is bi-partite, the number of nodes of the two types must be equal (3 objects and three processes in our case) and the length of the loop is therefore even.



P1 yields B2 and B1.
P2 consumes B3 and B1.
P2 yields B4.
P3 consumes B5.
P3 yields B3 and B6.
P4 consumes B2.
P4 yields B3.
P5 consumes B4.
P5 yields B5.
P6 consumes B6.

Figure 9. An OPD describing an Object-Process-Based Systems Graph (top) and its automatically-generated OPL paragraph

to / From	P1	B1	P2	B2	P3	B3	P4	B4	P5	B5	P6	B6
P1	▨	R		R								
B1		▨	C									
P2			▨					R				
B2				▨			C					
P3					▨	R						R
B3			C			▨						
P4					R		▨					
B4								▨	C			
P5									▨	R		
B5					C					▨		
P6											▨	
B6											C	▨

Figure 10. The Design Structure Matrix that represents the graph and OPL text of the OPD in Figure 9

Figure 10 is the DSM of the OPD in Figure 9, where C and R respectively denote a consumption link, from object to process, and a result link, from process to object. To see the pattern of the loop {P2, B4, P5, B5, P3, B3} in the OPD, in Figure 11 we have rearranged the rows an columns of the DSM in Figure 10 to have P2, B4, P5, B5, and P3 arranged in a RCRCR sequence, and obtained an interesting pattern of alternating R's and C's right above the DSM diagonal, which was originally grey, with the DSM cell (B3, P2) completing the loop as the apex of an isosceles triangle whose base is the five-cell RCRCR string. Moreover, the grey cells of the displaced DSM rows and columns, which were originally along the diagonal, are now arranged in a row beneath this "triangle".

to From	P1	B1	P2	B4	P5	B5	P3	B3	B2	P4	P6	B6
P1	▨	R							R			
B1		▨	C									
P2			▨	R								
B4					C							
P5						R						
B5							C					
P3								R				R
B3			C									
B2				▨	▨	▨	▨	▨	▨	C		
P4							R			▨		
P6											▨	
B6											C	▨

Figure 11. The Design Structure Matrix that represents the graph and OPL text of Figure 9 with rows and columns rearranged to visualize the loop {P2, B4, P5, B5, P3, B3}

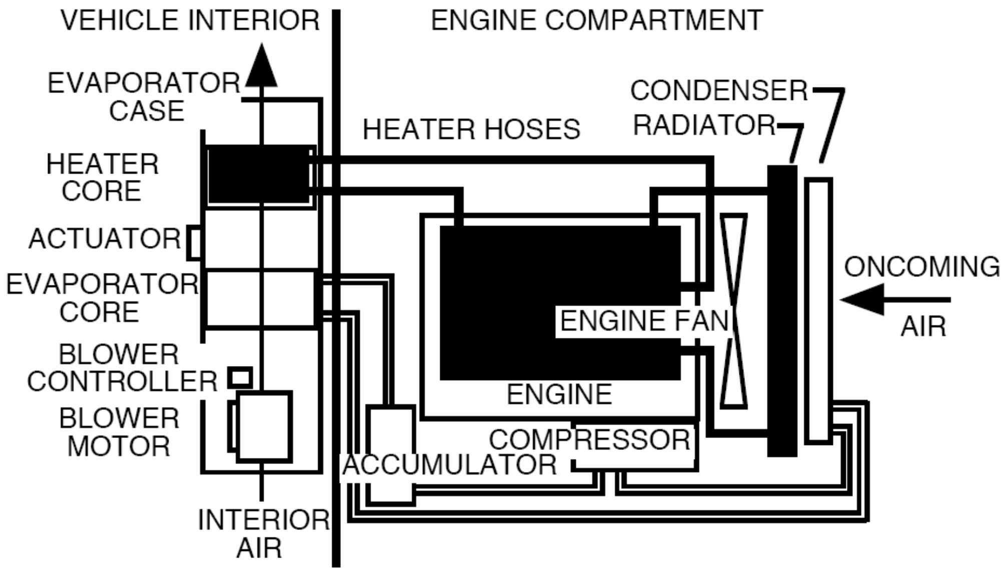
Component-Based DSM

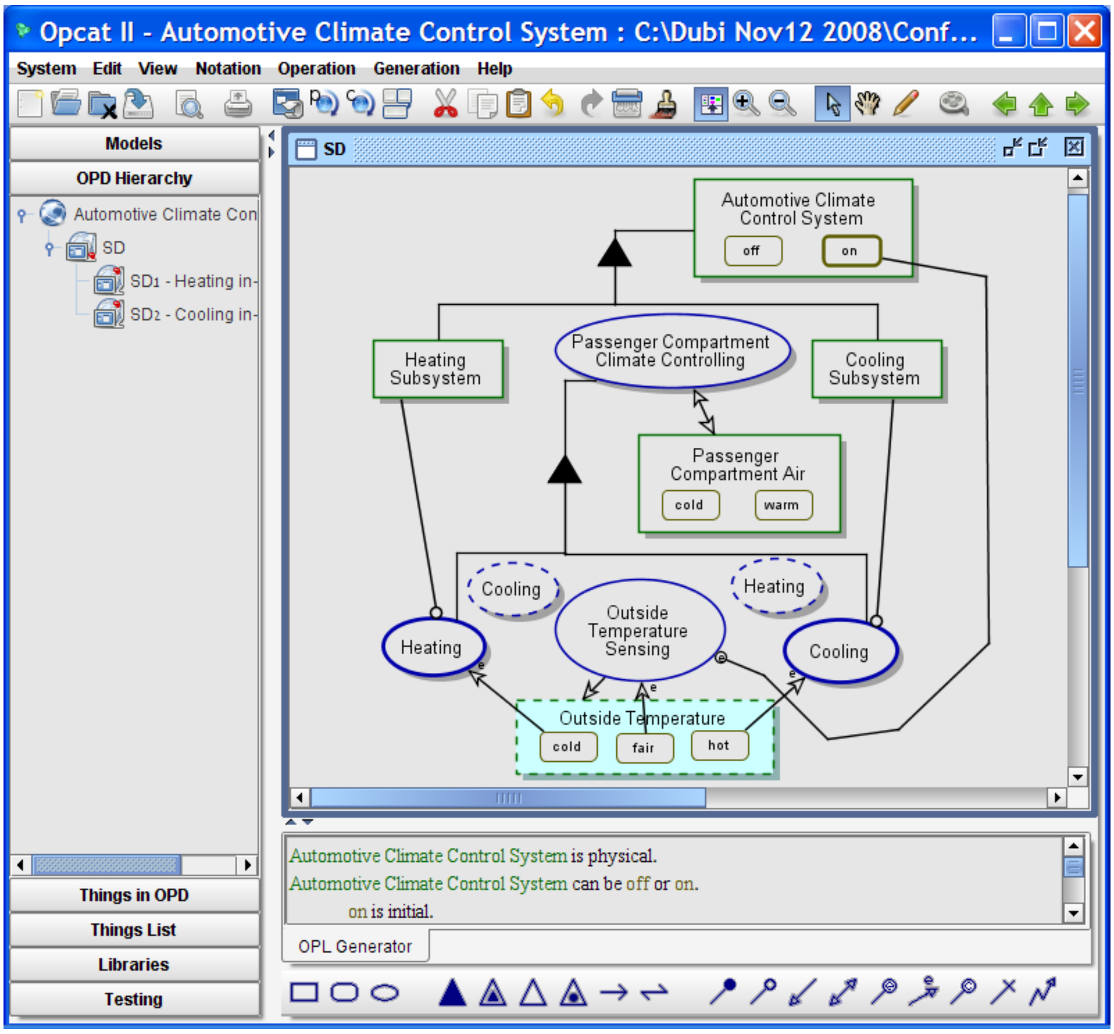
A component-based DSM documents interactions between elements in a complex system architecture. Different types of interactions can be displayed in the DSM. Types of interactions will vary from project to project.
Some representative interaction types are shown in the table below (Pimmler and Eppinger, 1994).

Spatial	needs for adjacency or orientation between two elements
Energy	needs for energy transfer/exchange between two elements
Information	needs for data or signal exchange between two elements
Material	needs for material exchange between two elements

The following example is taken from (Pimmler and Eppinger, 1994). An automotive climate control system performs two basic functions: passenger compartment heating and cooling. Heating is achieved by circulating hot engine coolant via the heater hoses through the heater core (a heat exchanger). Passenger compartment air is forced across the heater core to transfer energy. Engine coolant is also circulated through the radiator

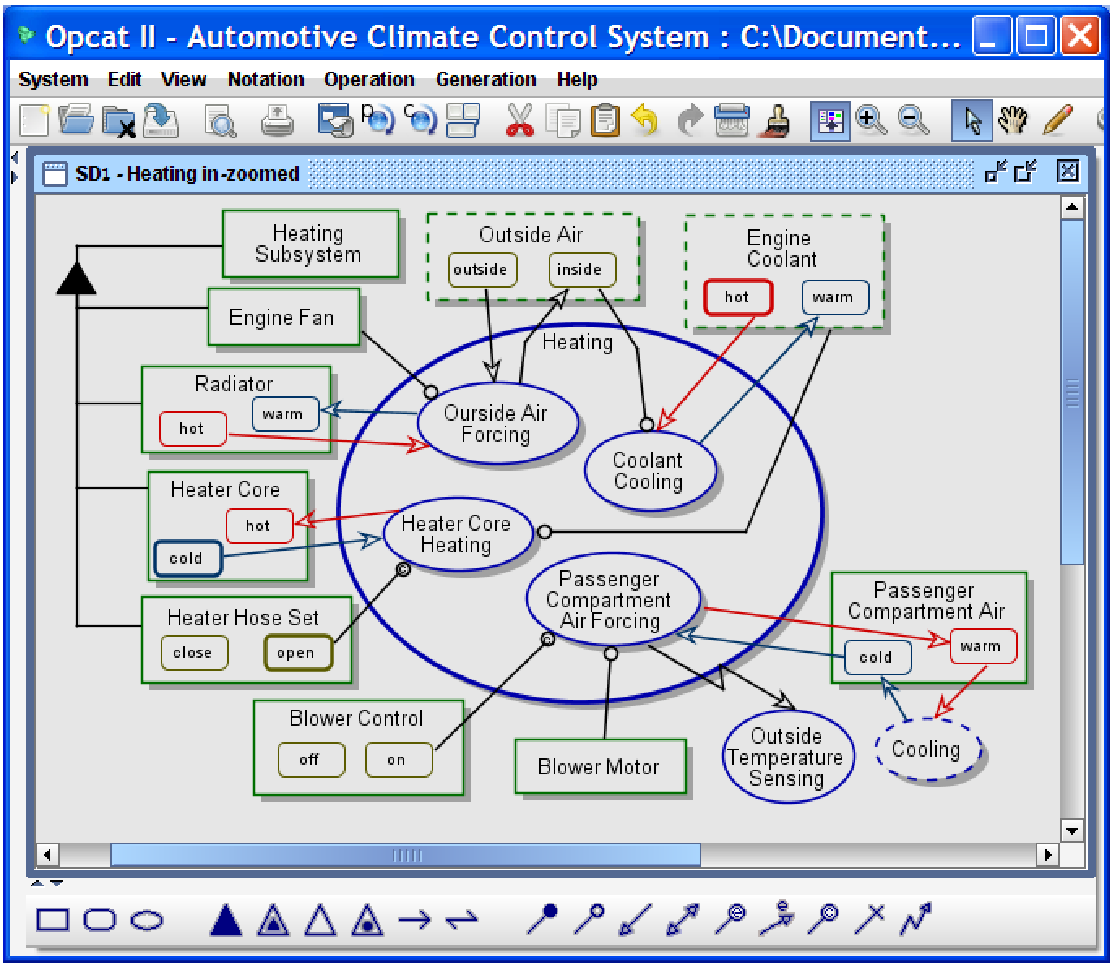
(another heat exchanger) to dissipate additional energy to the outside air. Passenger compartment cooling is achieved using a refrigeration loop comprised of five main components: compressor, condenser, evaporator, expansion valve (not shown), and accumulator. The compressor receives low pressure refrigerant gas and delivers high pressure gas to the condenser (also a heat exchanger), where it condenses, dissipating energy to the outside air. The high pressure liquid then flows to the evaporator through an expansion valve (not shown), which maintains the pressure difference in the loop.
As the refrigerant expands, heat energy is absorbed through the evaporator, which is another heat exchanger over which warm passenger compartment air is passed. The refrigerant, now a mixture of liquid and gas, then moves to the accumulator, which traps the liquid droplets, allowing only gas to enter the compressor.



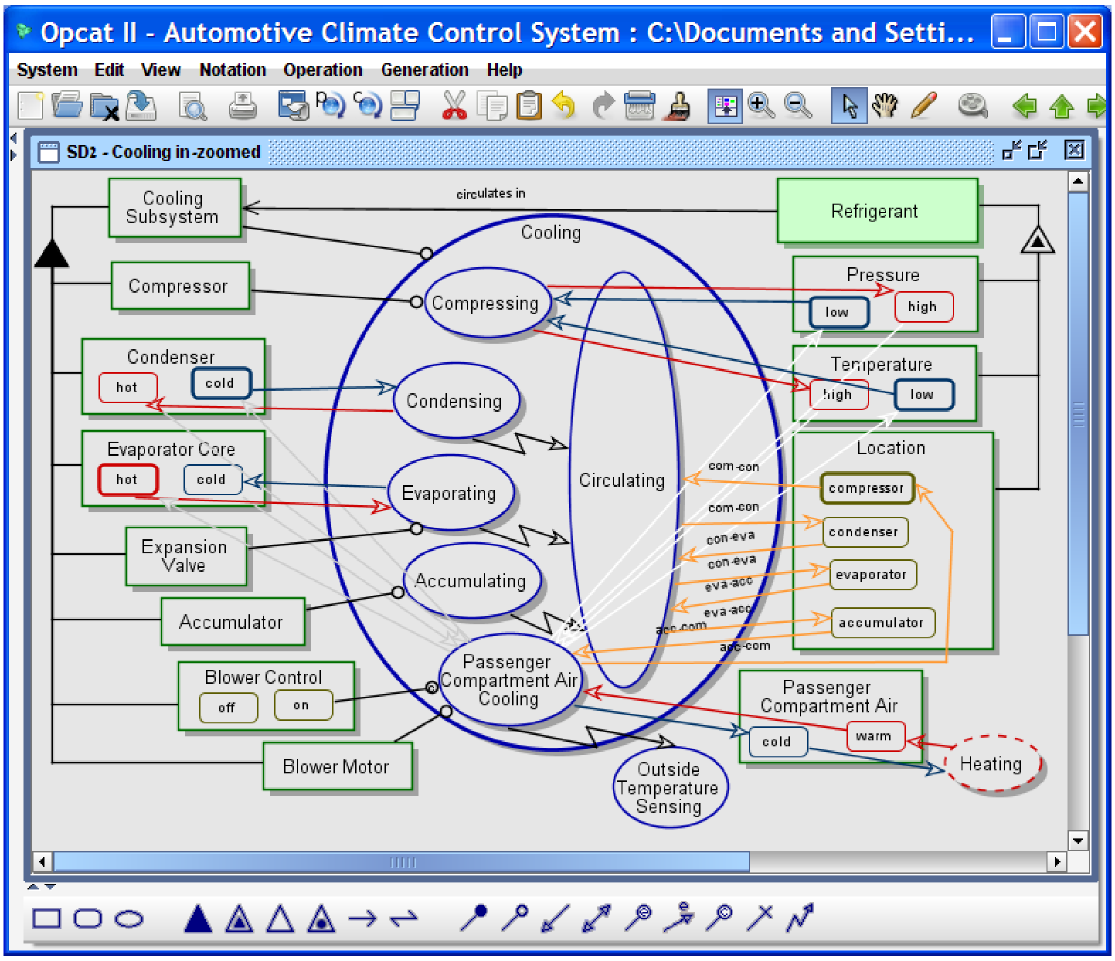


Opcat II - Automotive Climate Control System : C:\Dubi Nov12 2008\Conf...
System Edit View Notation Operation Generation Help
Models
OPD Hierarchy
Automotive Climate Con
SD
SD1 - Heating in-
SD2 - Cooling in-
Things in OPD
Things List
Libraries
Testing
SD
Automotive Climate Control System
off
on
Heating Subsystem
Passenger Compartment Climate Controlling
Cooling Subsystem
Passenger Compartment Air
cold
warm
Cooling
Heating
Outside Temperature Sensing
Heating
Cooling
Outside Temperature
cold
fair
hot
OPL Generator

Automotive Climate Control System is physical.
Automotive Climate Control System can be off or on.
on is initial.



Opcat II - Automotive Climate Control System : C:\Document...
System Edit View Notation Operation Generation Help
SD1 - Heating in-zoomed
Heating Subsystem
Engine Fan
Radiator
hot
warm
Heater Core
hot
cold
Heater Hose Set
close
open
Blower Control
off
on
Blower Motor
Outside Air
outside
inside
Engine Coolant
hot
warm
Heating
Ourside Air Forcing
Coolant Cooling
Heater Core Heating
Passenger Compartment Air Forcing
Passenger Compartment Air
cold
warm
Outside Temperature Sensing
Cooling



		A	B	C	D	E	F	G	H	I	J	K	L
Radiator	A	A											
Engine Fan	B		B										
Heater Core	C			C									
Heater Hose Set	D				D								
Condenser	E					E							
Compressor	F						F						
Evaporator Core	G							G					
Evaporator Valve	H								H				
Accumulator	I									I			
Sensor	J										J		
Blower Control	K											K	
Blower Motor	L												L

As an example, consider the material interaction between components for an automobile Climate Control System.

		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**N**	**M**	**O**	**P**

Radiator	**A**	A	X														
Engine fan	**B**	X	B														
Heater Core	**C**			C													X
Heater Hoses	**D**				D												
Condenser	**E**		X			E	X		X								
Compressor	**F**					X	F		X	X							
Evaporator Case	**G**							G									X
Evaporator Core	**H**					X	X		H	X							X
Accumulator	**I**						X		X	I							
Refrigeration Controls	**J**										J						
Air Controls	**K**											K					
Sensors	**L**												L				
Command Distribution	**M**													M			
Actuators	**N**														N		
Blower Controls	**O**															O	X
Blower Motor	**P**			X				X	X							X	P

Clustering the "X" marks along the diagonal of the DSM resulted in the creation of three "chunks" for the Climate Control System. The "chunks" are (Pimmler and Eppinger, 1994):

1. Front End Air Chunk.
2. Refrigerant Chunk.
3. Interior Air Chunk.

		D	**J**	**K**	**L**	**M**	**N**	**A**	**B**	**E**	**F**	**I**	**H**	**C**	**P**	**O**	**G**
Radiator	**D**	D															
Engine fan	**J**		J														
Heater Core	**K**			K													
Heater Hoses	**L**				L												
Condenser	**M**					M											
Compressor	**N**						N										
Evaporator Case	**A**							A	X								
Evaporator Core	**B**							X	B	X							
Accumulator	**E**								X	E	X		X				
Refrigeration Controls	**F**									X	F	X	X				
Air Controls	**I**										X	I	X				

Sensors	**H**									X	X	X	H		X		
Command Distribution	**C**													C	X		
Actuators	**P**												X	X	P	X	X
Blower Controls	**O**														X	O	
Blower Motor	**G**														X		G

[Back to Tutorial Index]

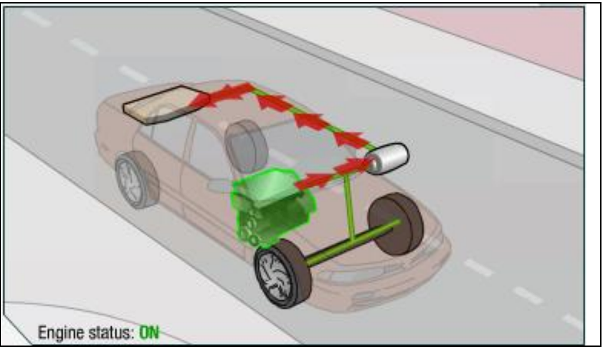
OVERVIEW

Hybrid-electric vehicles combine the benefits of gasoline engines and electric motors to provide improved fuel economy.

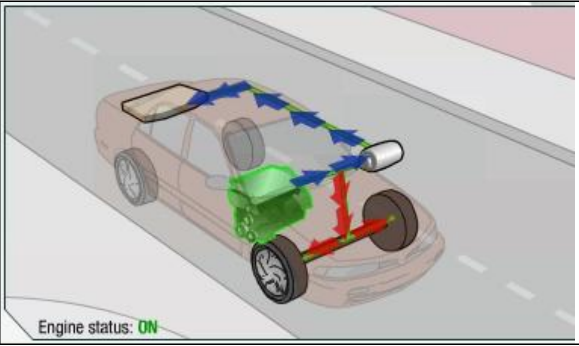
The engine provides most of the vehicle's power, and the electric motor provides additional power when needed, such as for accelerating and passing. This allows a smaller, more-efficient engine to be used.

The electric power for the motor is generated from regenerative braking and from the gasoline engine, so hybrids don't have to be "plugged in" to an electrical outlet to recharge.

Staring

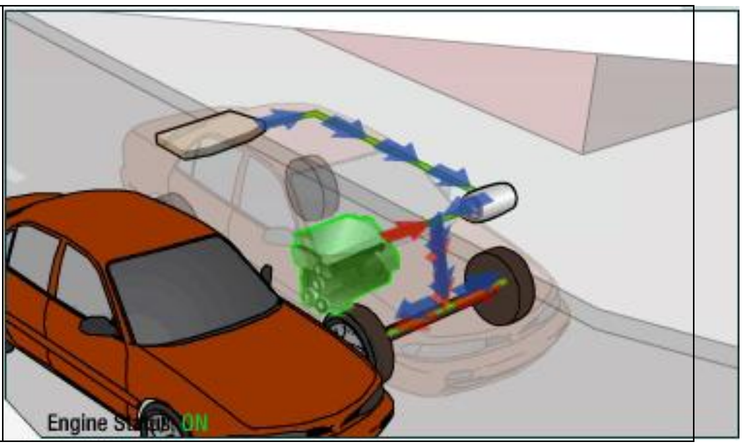
When the vehicle is started, the gasoline engine "warms up." If necessary, the electric motor acts as a generator, converting energy from the engine into electricity and storing it in the battery.	

Cruising

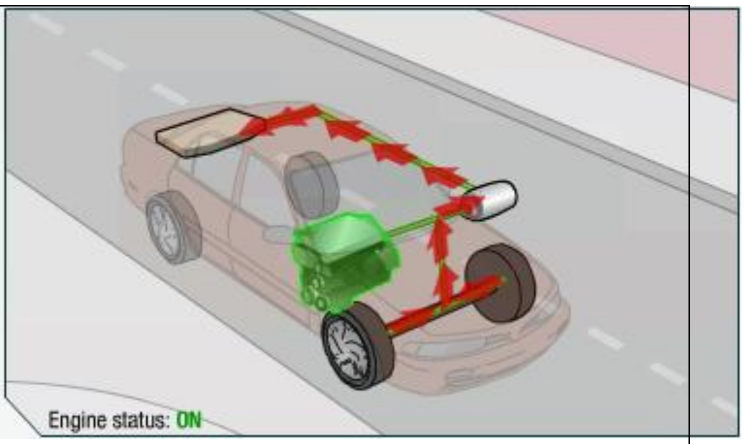
When the vehicle is started, the gasoline engine "warms up." If necessary, the electric motor acts as a generator, converting energy from the engine into electricity and storing it in the battery. The gasoline engine powers the	

vehicle at cruising speeds and, if needed, provides power to the battery for later use.	

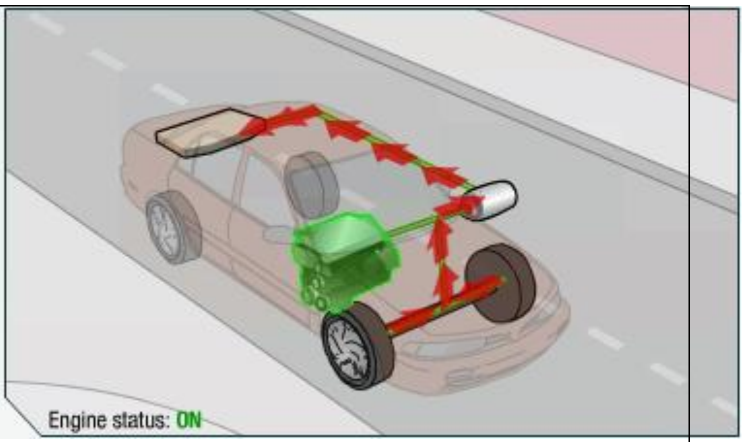
Passing

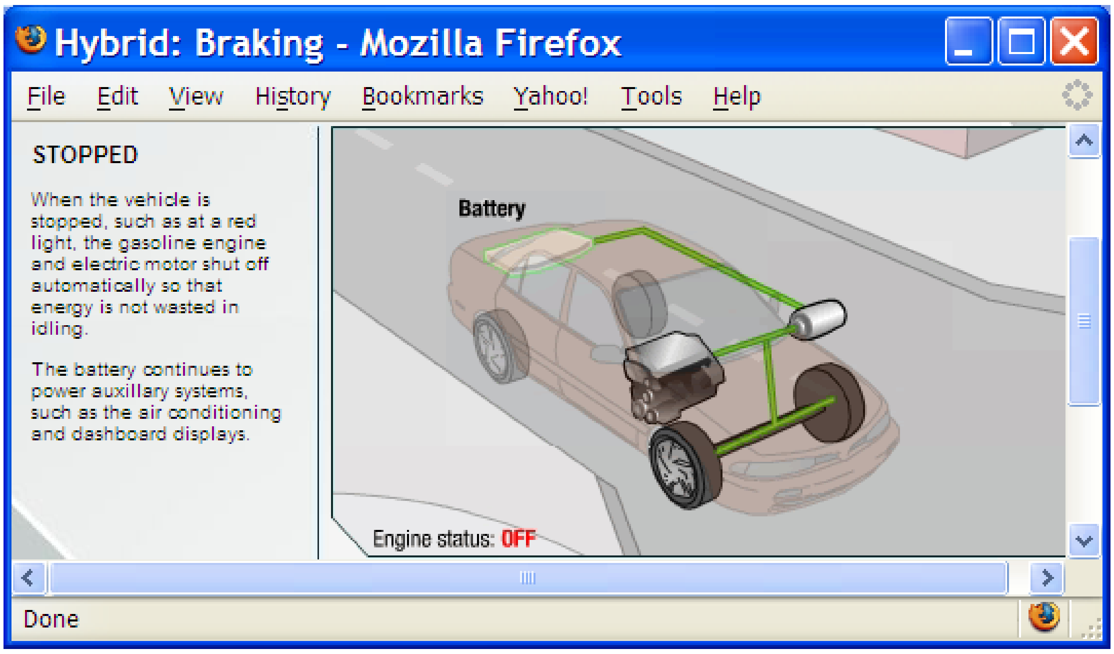
During heavy accelerating or when additional power is needed, the gasoline engine and electric motor are both used to propel the vehicle. Additional power from the battery is used to power the electric motor as needed.	

Braking

Regenerative braking converts otherwise wasted energy from braking into electricity and stores it in the battery. In regenerative braking, the electric motor is reversed so that, instead of using electricity to turn the wheels, the rotating wheels turn the motor and create electricity. Using energy from the wheels to turn the motor slows the vehicle down. If additional stopping power is needed, conventional friction brakes (e.g., disc brakes) are also applied automatically.	

Stopping

When the vehicle is stopped, such as at a red light, the gasoline engine and electric motor shut off automatically so that energy is not wasted in idling. The battery continues to power auxiliary systems, such as the air conditioning and dashboard displays. If additional stopping power is needed, conventional friction brakes (e.g., disc brakes) are also applied automatically.	



STOPPED

When the vehicle is stopped, such as at a red light, the gasoline engine and electric motor shut off automatically so that energy is not wasted in idling.

The battery continues to power auxiliary systems, such as the air conditioning and dashboard displays.

References

Dori, D. 2002. *Object-Process Methodology: A holistic systems paradigm*. Berlin: Springer.

Dori, D., I. Reinhartz-Berger, and A. Sturm. 2003. Developing complex systems with Object-Process Methodology using OPCAT. *Lecture Notes in Computer Science* 2813:570-572. Berlin/Heidelberg: Springer.

Browning, T. Applying the Design Structure Matrix to System Decomposition and Integration problems: A Review and New Directions. IEEE Transactions on Engineering management, Vol. 48, No. 3, August 2001.

Pimmler, T.U. and Eppinger, S.D. Integration Analysis of Product Decompositions. ASME Design Theory and Methodology Conference, Minneapolis, MN, 1994.

Biography

Dov Dori is Visiting Professor at MIT's Engineering Systems Division (ESD). Between 2001 and 2008 he was Head of Technion's Area of Information Systems Engineering at the Faculty of Industrial Engineering and Management, and Research Affiliate at MIT. Between 1999 and 2001 he was Visiting Faculty at MIT's Sloan and ESD. Professor Dori received his B.Sc. in Industrial Engineering and Management from the Technion in 1975, M.Sc. in Operations Research from Tel Aviv University in 1981, and Ph.D. in Computer Science from Weizmann Institute of Science, Israel, in 1988. Between 1978 and 1984 he was Chief Industrial Engineer of the MERKAVA Tank Production Plant. His research interests include Model-Based Systems Engineering, Systems Development and Lifecycle Methodologies, and Information Systems Engineering. Between 1999 and 2001 he was Associate Editor of IEEE T-PAMI and is currently Associate Editor of Systems Engineering. He is author/co-editor of four books and author of over 150 publications. Prof. Dori is Fellow of the International Association for Pattern Recognition (IAPR), Senior Member of IEEE and ACM, and member of INCOSE.