ANATOMY OF SYSTEM NOTATIONS

CODEX DESIGN + VISUAL THINKING

V.0.9.0 BY PETER STOYKO

graphical notations

In the study of systems, a graphical notation is a set of visual conventions for diagramming systems in a standard way. As the old joke goes: what's great about standards is there's so many to choose from. When it comes to system notations, that is an extreme understatement and therein lies a dilemma.

Elaborate systems have many moving parts, usually too many to juggle in the mind at once. Notations offer a shorthand for keeping track of it all while considering the larger whole. However, each notation has its own assumptions of what a system is and how it works, which can be an awkward fit if the viewer brings different theoretical understandings. Most system notations have been devised for narrow uses by highly specialized experts. As a result, a large number of notations have emerged but only a few have any general appeal.

Dilemmas are opportunities in disguise. Only one or two notations were devised with any thought given to graphic- and information-design technique. Many are downright amateurish in their treatment of visual cognition. There is much room for improvement. Yet notations are used by isolated disciplines with little sharing of lessons learned. Efficacy is under-studied, even as shortcomings become obvious. All these dilemmas point to the need to take stock of the various notations to look carefully at how they differ and why.

The *Anatomy of System Notations* is an inventory of the graphic devices used in system notations. The ultimate goal is the creation of a new notation that: (a.) can express the full range of elements and dynamics of systems; (b.) can make sophisticated system analysis available to everyone, including across disciplines; and, (c.) works with emerging forms of interactive media. The *Anatomy* moves us closer to that goal on three counts. First, the shortcomings of notations can be catalogued. These are listed separately from this poster as a set of *Problem Cards*. Second, the *Visual Vocabulary of Systems* is a codex that captures all the elements and dynamics of systems within the theoretical literatures of various disciplines. By comparing the *Anatomy* to the *Visual Vocabulary*, it is possible to see what subjects are hard (or impossible) to express using existing notations. Third, lessons can be drawn from innovative techniques found within existing notations.

Notations are not the only way to visualize systems. Information graphics, gigamaps, and synthesis maps mix various types of graphics to explain systems through bricolage. Popular genres of video games are about system building and maintenance. Visual programming languages use graphical notations to create software. The list goes on. Much can be learned from these alternatives, which are gaining large audiences. These methods also make selective use of notations and would benefit from notation improvement. Notations, in turn, would benefit from cross-fertilization from these alternatives. All that by taking stock of what is available.

The larger concern is that notations have not kept pace with the complexity of systems and their entanglement in our lives. What sort of agency do we have if we do not grasp the full gamut of systems (natural and human-made) that push and pull us in all directions? What vulnerabilities and harms are we exposed to? A well-designed graphical notation can be part of a visual repertoire for making our highly systematized world easier to interpret.

basic forms

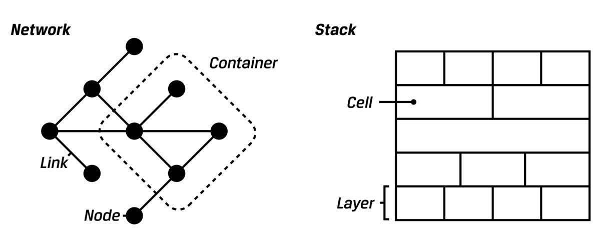
What counts as a graphical notation for system diagramming? For starters, it has to be a *set of graphic devices* (textual elements, symbols, and visuals) that can be assembled to meaningfully describe the moving parts of a system. The set can emerge as widely recognized conventions or be stipulated as a formal specification. Some formal specifications become standards governed by a professional association. Crucially, any system is made up of moving parts that interact to affect overall behavior. Thus, the set of graphical devices cannot merely describe static structures. Moreover, the devices have to be a stable, finite set. That rules out visualization methods that use whatever graphic devices seem appropriate for the case at hand; there has to be a standard basis with which to compare features of different systems.

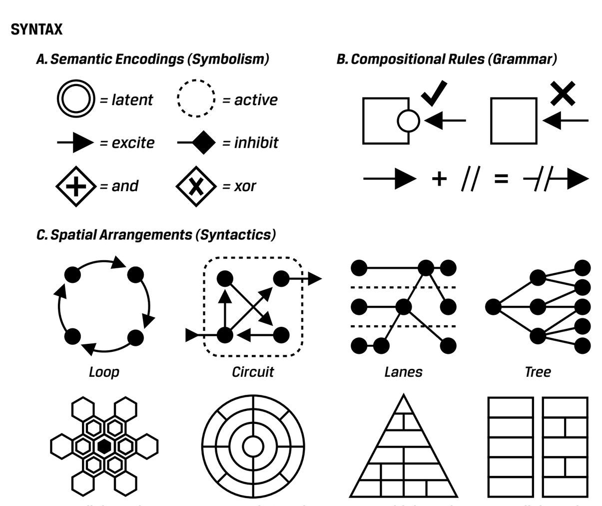
The result of a diagram is a *model*, a simplification of the system being described; the map does not equal the territory, so to speak. That does not mean the description cannot be rich, with all the most salient details included. Even with notations embedded within visual programming languages, where the diagram elements perform functions, not everything about the system is shown; there remains a great deal of activity working behind the scenes, abstracted away from the viewer's consideration.

Most notations are variations on two themes: the *network* and the *stack*, as shown below. Network diagrams (or hypergraphs) are arrangements of nodes and links. Stack diagrams can be thought of as nodes within nodes, as proximate or abutting borders between cells that may express hierarchy. System stacks are so-called because there is a loose order, from low-level sub-systems to higher order ones. Or, in the case of radial arrangements, from central cells to peripheral ones.

What sets one notation apart from any other? They differ along at least three dimensions. First, most diagram items are symbolic stand-ins for parts of actual systems. For example, an arrow with a solid line may indicate "excitory" relation, whereas a dashed line may be an "inhibitory" one. A square node may refer to "sellers" and a round one may be "buyers." There is a limit to how many *semantic encodings* a viewer can be expected to remember. Second, each notation has its own *compositional rules* which stipulate how the pieces are supposed to fit together visually. For example, some notations specify that links can only connect to nodes through "ports" (designated spots on the node). Third, a notation is made easy to interpret by meaningful spatial arrangements of items (*syntactics*). For example, a series of nodes and links may form a loop or a circuit that is recognizable as such. Some technical notations are complicated on one or more of these dimensions. Perhaps they have a large library of symbols that have to be learned. Or they have elaborate rules for diagram layouts.

The *Anatomy of System Notations* looks for commonalities across the various notations and groups them as types of graphical device. Multiple examples of each one are shown to show the range of implementations.





01 NODE

Every network-based diagram has lines (links) connecting objects called *nodes* (vertices, points, or elements). Nodes can represent actors, objects, processes, or places. They take the form of enclosed shapes, symbols, compact illustrations, or stand-alone text labels (with or without some form of bracketing).

02 LINK

A *link* (or edge) is any connection between nodes. If connecting more than two nodes, the link may be called a *hyper-link* (or hyper-edge). A link may abut or encompass a node.

03 LINK END

Links usually have symbolic ends encoded to indicate relation type. Arrowheads are the most common ends but many other symbol encodings are available. Doubled-up end-symbols (rows 6-7) multiply the number of possible encodings. M-IDEFØ combines up to four elements to show weak (—+▸) to strong influence (—++▸). Even simple arrows can have several semantic meanings across diagrams, such as causality, temporal sequence, ordered relation, interaction, state change, conditional relation, and spatial movement (Kurata & Egenhofer, 2005). End symbols can be placed at both ends of a link. However, some notations encode link beginnings differently (last two rows). That can make beginnings and ends of links clearer. The nature of the departure can also be encoded. For example, BPMN uses a slash (—/—) for the default option when more than one out-bound link is connected to a node.

04 LINK-END MODIFIER

Symbols can be placed next to a link end to qualify its meaning. For example, with IDEFØ, brackets around a beginning (or end) indicate a shift from (or to) a lower level of scale ("tunneled arrows"). In NCD notation, a color-coded triangle points to the beginning symbol it modifies (indicating a particular regulatory influence on the relation). A circle next to an arrow head signifies an inevitable effect in Southbeach notation. AML notation uses coded circles to modify the link with constraints (bottom row), potentially combined with doubled-up link-end symbols.

05 LINK LINE-STYLE

The line style of links can be coded to indicate a type or quality of relation. Dotted and dashed lines are common differentiators, as are color-coded lines. Links may be drawn thickly to provide space for an overlaid label. Sankey diagrams use line thickness to indicate magnitude of flow or influence.

06 LINK MODIFIER

A symbol placed along a link can qualify the nature of the relation. For example, iStar notation overlays a letter "D" to represent "dependency." Southbeach notation adds symbols for delayed effect and accellerated effect. G-OWL uses of letters in a line gap to modify the link encoding. One benefit of a modifier is that it can be applied to various line styles, instead of stipulating line styles for every possible combination of qualities.

07 NODE SHAPE

Notations that use multiple shapes for nodes encode each one with specific meanings, such as node function. Flowcharts are an obvious example, with different shapes indicating the type of operation to be performed on a case proceeding through the system. Shapes can also be drawn with a third dimension (last row) to make them stand out, at the risk of adding visual noise. A risk with multiple shapes is that certain ones look larger than others even with the same dimensions. Optical adjustments can be made to ensure certain nodes do not gain unintended emphasis.

08 RELATIONAL NODE

Nodes can indicate directionality or organization with shapes, often serving a role as link in stack diagrams.

09 NODE MODIFIER

Marks can be added to shapes to modify a node, such as change its function or state. As with link modifiers, use of node modifiers can be applied to different shaped nodes, reducing the need for a large number of shape encodings. This comes at the cost of less interior space to add labels or other markings. For that reason, badges and node-margin marks tend to offer greater flexibility.

10 NODE LINE-STYLE

Nodes and node status can be differentiated with lines styles. For example, BPMN (top row) has line styles that apply to a set of secondary nodes to indicate a status. Lines can also be color coded or offset with an underlying shadow.

11 NODE FILL

Nodes can contain a color or pattern fill to differentiate or encode a particular meaning.

12 NODE HALO

Instead of differentiating a node with a line style or fill, an enclosed space surrounding the node can be encoded using line style or fill, as with Wardley maps. A halo can also be used to group multiple objects as a single node.

13 NODE SYMBOL

A stand-alone symbol can be the node. A symbol can also stand out at the center of an enclosed shape to differentiate a node (third row). Moreover, a symbolic shape may be used for a node, as is common with computer science notations and the cloud symbol (☁). Stand-alone symbols are handy for secondary and tertiary nodes given their compactness. Engineering notations make extensive use of symbol nodes, often with subtle variations among symbols to indicate variety of types. Confusingly, some of these symbols look more like link modifiers (bottom row).

14 NODE HIERARCHY

Nodes are often differentiated with size and shape into designate tiers of functions. For example, a secondary node (transit node) may indicate a process or transformation between primary nodes, as with Petri Nets. Accordingly, tertiary nodes are positioned between secondary ones (as with ORM2) or represent minor points along links.

15 NODE COMPARTMENT

Nodes can be subdivided into sections separated by lines, with each compartment storing a particular type of information, usually a text string. Most commonly, a node will have a "marquee" compartment for the label and a "gutter" compartment for minor technical distinctions. With ORM2 (bottom row), secondary transit nodes may be divided for bi- (or tri-) directional links. Because links are expected to be readable both backwards and forwards, separate compartments represent different meanings based on the direction of movement.

16 CONJOINED NODES (COMPLEX)

Nodes that can otherwise stand apart are conjoined to indicated combination. Stacks tend to be conjoined nodes, although they can be arrangements of nearby nodes. Biological notations tend to call conjoined nodes a complex. With some notations, the boundary between nodes suggests a fitting together, as with SIGN notation where arcs indicate "docked" node edges (bottom left). Joined edges that follow the flow of links may indicate a sequence, as is common to Graham Charts (second row).

17 PORT

A port is a designated space on (or abutting) the outer edge of a node where links arrive or depart, often with information encoded in (or near) the space. For example, with many visual programming languages, ports indicate values or operations transmitted through links, as indicated by a port label. A variety of shapes and symbols are used to designate ports. Ports often imply a condition (such as compatibility) or a subroutine being performed to process an arrival or departure. Some notations take the idea to an extreme by covering a node with ports and port-label boxes (BIP) or ports with letter abbreviations (FRAM). Ports also apply to boundaries (see 27) but often with very different functionality.

18 PORT GROUP (INTERFACE)

A port group (also called an interface) are designated places on or near a node edge that contains one or more ports. Often this takes the form of an enclosure that fully (or partially) envelopes a set of ports. AADL is somewhat unique insofar as it does not conform to those patterns but, instead, has a symbol (◐) signifying multiple ports or acting as a connector to mixed port types (second row).

19 LINK INTERIOR

When lines branch from a node to create an acute angle, an arc line can be drawn to encode the branch with meaning. For example, with FODA-FD, an arc line with an angle fill indicates "OR" inclusiveness, whereas a simple arc line indicates "XOR" (mutual exclusion). That notation also allows for common shorthands used for conditional logic: "0..1" at least one chosen; "1" exactly one chosen; "0..*" an arbitrary number chosen; "1..*" at least one chosen; and such.

20 LINK JUNCTION (GATEWAY)

A link junction (or gateway) is a node through which links can branch or merge according to specified branch-logic shorthand. Simple branches (21) and merges (22) do not specify rules but may attribute a quality to the interaction. Each row to the left shows a different set of examples, the e3 notation uses lines and dots to indicate AND, OR and cardinality dependency (top row). BPMN uses symbol-encoded diamonds for similar purposes. DBS-VN (bottom row) uses symbols and brackets to imply the nature of the merger taking place.

21 BRANCH (FORK)

A link can split into multiple links. A perpendicular branch can signal a side-effect or secondary relation. Some notations add a waypoint symbol to indicate the nature of the split, such as SBGN using different symbols to indicate dissociation (◎) and truncation (⊠). Railroad-S notation uses nodeless branching throughout to indicate optionality. Graham chart uses a bracketing technique (bottom row, left example).

22 MERGE (JOIN)

A link can join another link, often to signal a contributory or reinforcing influence. SBGN uses line ends to signify promotion (-○), inhibition (-|), or association (-●) of the merge transition (second row). And mEPN and eMIM notations do likewise with a zig-zag line end (third row). Multiple notations use a bar-like line (bottom row) to provide a large number of in-coming links a tidy target to point to.

23 NODE JUNCTION

A node junction specifies how (a.) a link branches when more than one departs from a node or (b.) how multiple links merge when arriving at a node. It is not a separate node but an attachment to a node. Usually Boolean operators (AND, OR, XOR) or other forms of conditional logic are signified. More common methods for doing this are link junctions, link interiors, and link-end labels.

24 DETOUR

While connecting one node to another, a link may rebound off of a third node in a different causal chain, as indicated by an abrupt link detour. For example, Graham charts have a V-shaped line-segment to show how causal relation will have a side-effect on another process (top row). The reverse is possible, as one flow rebounds off another to pick up an additional quality or taint, as with some variants of MBWD notation (bottom row).

25 CONTAINER

A container encircles one or more nodes (and links) to set them apart for some purpose. For example, CLD notations use containers to show that a simply group like items together. In that way, it can be an underlying domain or environmental context. Containment may demarcate tangible boundaries or modular sub-systems, as with nested nodes (28). Thus, container boundaries can have ports or interfaces and allow movement. Containers can also be discrete assemblages of circuits which have no links between them. The stack cells, are containers. Containers can be discrete categories or be overlapping, as with lanes and groups. Some sort of line style or fill color differentiates containers from ordinary nodes.

26 DIVIDER

Lines can organize chains of nodes and links without enclosure. Such dividers can be a tangible boundary or conceptual distinction. A line style usually differentiates a divider from other lines. Dividers often form lanes or columns to represent parallel domains. Dividers can also subdivide containers, such as BPMN's "swim lane" divisions within larger "pools."

27 BOUNDARY PORT

Boundary ports can regulate access to nodes within a container (or on the other side of any sort of boundary). A single port may offer access through a boundary and into a node, or allow some links to pass through others. For example, a boundary port can regulate access from the inside outward, not just from the outside inward (bottom right). As with node ports, these can be organized using interfaces too (bottom left).

28 NESTED NODE

Nested nodes are one or more nodes located inside of other nodes. The nesting can represent a lower order of scale: the internal operations of the node that would otherwise be too small to show on the diagram. That does not preclude nested nodes from interacting with nodes nested inside other nodes. A nested node is distinguished from a container by the ability of the encompassing node to interact as any node would. For example, VSA notation (top row) shows nested nodes interacting across encompassing nodes, while the encompassing nodes act as aggregate agents. UML (middle row) uses nesting to show both sub-systems and the classification of instances. Energese (bottom row), uses encompassing nodes to represent organizations.

29 GROUP NODE

A group node aggregates a collection of mixed items. The items can show changes to internal composition or structure of the larger node. They can also be a shorthand for representing collective behavior. MBWB notation uses a group node to show bound chemical elements (bottom row). Perhaps confusingly, these structures are shown as networks within a larger network diagram, creating a need to differentiate both types of networks visually.

30 DECOMPOSITION BLOCKS

Nodes or containers can be used to represent multiple levels of scale. Decomposition blocks show this using a magnification metaphor, with projection lines indicating that a lower level of magnitude is being represented. With SDL and SADT notations the blocks are shown as flat two-dimensional planes (bottom example). Even though the example shows underlying systems also in 3D (isometric) perspective, this need not be the case; the planes may be shown in 2D but the contents can be a flat 2D diagram. Call out boxes (■) common to information diagrams and comic-book dialogue can also be used as decomposition blocks, although many notations use that particular graphic device for notes (60).

31 (ACYCLICAL) CONDENSATION

Busy networks can obscure big-picture effects. Conversely, summary networks can hide salient underlying events. Some notations summarize causal links between loops and circuits (DAG notation) while retaining cyclical dynamics. For example, the top row shows a super-imposed condensation, while the bottom row offers an acyclical "macro network" below the cyclical one (see Wahl & Runge, 2023).

32 DISCLOSURE TOGGLES

The DSRP notation has nodes with corner triangles to reveal node parts. The triangles work like the triangle toggles on Graphical User Interfaces (GUIs). On a static diagram, a downward triangle (▼) indicates that constituent parts are arranged below. A rightward (▶) triangle indicates constituent parts exist but are not shown. Toggles are used with transit nodes too (bottom row), as well as a variety of spatial arrangements of component parts below the node. This sort of device is ideally suited to interactive diagrams rather than static ones. BPMN also has a toggle (⊞) placed more centrally to indicate "collapsed sub-process."

33 INSET FUNCTION

Interaction between nodes may not be straight-forwardly linear. A graph may be shown inside a node to show curvilinear interactions or even more complicated functions. Energese shows how a path changes depending on what other influence happens to be interacting (top row). The graphs of two different nodes can even be connected with separate lines to show more precisely how the interaction works (third row). Visual programming languages have more flexibility for showing "live" graphs on nodes as interaction proceeds. Some flow-charts will indicate the desired distribution of a quantity across categories (bottom row).

34 INSET TOPOLOGY

Mathematical and computer-science notations may show network archetypes inside nodes, indicating an underlying structure. These miniature diagrams are more like thumbnail sketches than fully-fledged diagrams insofar as they are unlabeled and mostly indicate network type or general circuit structure.

35 LOOP LABEL

A closed loop can have a free-floating label at its center as an icon or text label. For example, CLD notations labels reinforcing and balancing loops. Some icons contain arrowed lines that potentially interfere with the interpretation of nearby links. Nested loops make it harder to associate a free-floating label with a loop.

36 SELF-LINK

Nodes can link to themselves to indicate iteration or a continuation of state. For example, MCD notation shows state transitions. If an object's previous state recurs over the next cycle, it is shown as a self-link, often with a label indicating the likelihood of its occurrence (expressed as a probability).

37 PROXIMATE MARK

A compact mark or symbol is placed close to an item (usually a node) to add a qualifier. For example, some process-map notations assign roles to particular nodes and links, represented by color-coded circles that are placed near the item (sometimes in an overlapping arrangement to conserve space and indicate grouping). ORM2 uses lines placed above or below bi-directional transit nodes to designate uniqueness constraints and a nearby circuit icon (⟲, ⟳, etc.) to indicate a ring constraint. Proximate marks can be placed anywhere near an item but most notations that use them place them above the item in a designated relative position (left, center, or right side).

38 NODE-MARGIN MARK

Small icons or encoded markings can be placed just inside the outer boundaries of rectangular nodes to indicate node type, quality, or state. Designated corners are often reserved for a particular set of marks to set viewer expectations. A corner mark can protrude out of one side of the node's outer boundary or be placed on the boundary line, but cannot be positioned entirely outside (tag, 39) or mostly outside as an overlay (badge, 40).

39 BADGE

A badge partially overlays an item (node or link) to modify it in some way. Badges are often used to indicate status and valence of a link. Part of an underlying node becomes obscured, which precludes margin marks in the same corner or side as the badge. Beyesian flowgraphs use a badge ("E" in a rounded square) to indicate "evidence," either "hard" (solid line) or "soft" (dotted line). Some process-map notations place color-coded shapes along links to indicate which roles are involved in the action. Notations that accommodate unknowns or uncertainty often do so with the help of badges.

40 TAG

A tag underlays or abuts an object (node or link) to modify it in some way. Tags are similar to badges but do not obscure any part of the object they modify. Tags are often more free-floating. Proximate labels on dense diagrams invoke the ambiguity that tags can create. An interesting example is SOD notation's use of tags on links (near where nodes are) to express how visible a particular influence is.

41 PROXIMATE LABEL

Viewers associate items that are placed together in close proximity. Accordingly, labels can be placed near items they refer to. To avoid misattribution in dense diagrams, free-floating labels are best placed in a predictable position relative to items, such as labels directly below nodes or at the mid-point of links.

42 LABEL MARK

Labels can be modified with a mark. Often this takes the form of a small icon used to bullet the label. BPM uses a set of label marks directly above a nested label, as well as label underlining as an encoded signifier.

43 NESTED LABEL

Many notations place labels within nodes, either at the top or center. Top labels may come with additional compartment, bullet mark, or background shape. Nested labels are constrained by the interior space and shape of the node, which often results in abbreviations, acronyms, and squinty text sizes.

44 OVERLAID MARKS

A mark can be placed overtop of the whole of a node in order to designate it in some way. The most common example is the X-ing out of nodes, which happens with YAWL notation.

45 HYBRID NODES

Two distinct node types may be combined in order to indicate qualities of both subtypes. For example, Graham charts rely on nodes made of basic shapes which can be laid overtop of each other to create a new type of node encoding.

46 PILED NODES

Nodes can be arranged in a pile, with multiple versions of the same symbol overlaid with a partial offset. The meanings of these piles can vary significantly across notations. Flowchart style notations (such as Drakon) often use piles as just another form of symbol. Piles can signify a related set. Some notations (such as YAWL) use piles to indicate multiple instances, with a label placed above to add relevant details.

47 COUNTER

A counter indicates the number of items in a node at a particular point in time, with the node representing a procedure or station as part of a larger sequence. The counters move within the system diagram, leaving one node and landing in another. This device is helpful if nodes have finite capacities. The Petri net is a notation that requires counters. Machinations notation uses stacked piles of color-coded counters for resources stored in a node. When the number of counters becomes too large to illustrate, the counters are replaced by numbers.

48 RUNNING OBJECTS

Objects can move along links or be shown moving along-side them. With SC notation, multiple circle-with-arrow symbols run parallel to directionless links to show signals (data) passing between nodes. The second row shows running counters from Machinations notation, with animated Petri Net counters traveling along links in a similar way. SOD notation relies on running signals, including a backward-bending arrow and label to indicate a signal that "balks" before arriving.

49 RUNNING ARROWS

Unlabeled arrows can be placed alongside a link to indicate directionality. That might be done because the link is very long and winding, or because other available link encodings are used for non-directional information. A more space efficient option is to add arrow heads at one or more points along the link. A less obvious application is to indicate secondary flows that might run counter to primary directional indicators.

50 QUEUE

When a notation uses nodes as stations or process stages, there may be an implied wait. However, most notations do not make timing intrinsic to the way a diagram works. Back-like tables are thus inserted to represent a capacity constraint, wait, or threshold. These are "nodes" in a loose sense; more like elaborate combinations of tables, symbols, counters, and nodes. A few notations use ordinary nodes, such as symbolic nodes for delay (⊠) and queue (◎) in Machinations. The convention of showing a queue as a rack-like label has made it somewhat recognizable across notations in particular fields, so much so that symbolic nodes use a simplified version (|||).

51 TIMING CUE

Timing may be important to a system. Accordingly timing durations can be added to nodes and links. However, it may not always be clear when the overall timing starts and stops. Thus, special cues may be added, either to links (first example) or nodes (second example).

52 FLOW LIMITER

Notations for physical systems involving flows or currents can have (a.) valves to restrict the rate of flow, (b.) gates that place conditions on passage, or (c.) switches that turn flow on and off. Some general notations use such devices, as with S&FD relying heavily on valves. These flow limiters are a special class of either node or link modifier depending on whether they act as a control point or only as a gauge respectively. Some have a gauge-like extension to show rate information (bottom left two) which can be a live status display in non-static diagrams.

53 MULTI-THREADED LINK

Links can be shown in parallel between nodes to show aggregate or reinforcing effects, or to reduce link clutter. The spatial density of the multi-threaded link makes it difficult to apply labels to individual threads until they branch out. Thus, each thread is usually color-coded, as with Subway Maps in the Beck/Vignelli diagram style. Multiple link ends may be combined with a more noticeable indicator of how many links end at a node.

54 JOINT

Links can connect continuously without a node per se. Indeed, conduits connecting directly with other conduits is common to many systems. The joint (or binding) can be made explicit, with a point showing a plug and receptacle, as with electrical notations. Or the joint can be a mixed sub-notation. That makes compatibility between links obvious. Link ends can overlap. The joint can also be implicit with a change in line styles.

55 BRIDGE

Connecting links between other links can serve a function, such as allow connectivity between otherwise incompatible conduits. Unit VPS notation uses bridges ("plugs") to control link pathways. NCD notation uses bridges to signify a change of the signal transmitted through links, an "inhibition" or "excitation" of the signal. Some UML notations use bridges ("complex connectors") to connect a link to two ports simultaneously.

56 SEGMENT

Sections of a link may be singled out for a particular purpose. SysML notation (UML) uses brackets to signify "coregion," grouping any links that branch or merge in that span. mEPN groups nodes and links along a longer path with brackets. The link may also be made up of segments of finite length, with ends indicated by notches, as with several engineering notations.

57 WAYPOINT

A place where a link changes course is called a waypoint. This can be a single point or an enclosed zone.

58 LASSO

A lasso groups links into bundles as they flow through an enclosed shape. That may be for purposes of labeling nearby links. With Energese notation, a diamond shape ties together two flows into a transaction, such as money in exchange for materials.

59 CROSS-OVER

A link can "jump" over another link to show that there is no interaction; the lines merely get in each others way and a cross-over connector makes that clear.

60 NOTE

Clarifying annotations can be added to boxes specifically encoded for that purpose. KiPN is an example that relies heavily on notes and has several types, mostly differentiated by line styles.

61 LABEL LINES

A proximate label can be associated with an object using a line. To avoid confusion, the line is differentiated from links, usually by using a kinked (N-type) line.

62 SYMBOLIC LINK SHAPE

Symbolic link shapes are a special case of a link where the way the link is drawn is intended to invoke a meaning. A link shaped like a lightning bolt is an obvious example. An arrow with a wider "stem" (relative to its "neck") is a symbolic shape that often indicates a concentrated flow or culminating effect. Symbolic links that are harder to detect include those with symbolic kinks, which might be mistaken for ordinary way-points.

63 N-CASE SHORTHAND

If there are multiple instances of the same sub-system, then two or three examples can be shown, with the rest replaced by the points-of-ellipsis symbol (⋯). This is common to several flow- and wire notations. This symbol may also be used if part of the diagram has been removed for some other purpose. Points of ellipsis can also be used as a generic indicator of "continuation."

64 JUMP

A jump is a place where a diagram ends but is expected to continue on another page (or space). A jump is also called an "off-page reference" in flowcharts and engineering notations.

65 TERMINUS

Notations may have points beyond which system interactions cease. For example several notations have symbol nodes for entropy, degradation, or discard. A terminus can also take the form of a special type of link end.

referenced notations

AADL	**Architecture Analysis & Design Language** (Feiler et al., 2006)
AML	**Agent Modeling Language** (Cervenka & Trencansky, 2000)
BCD	**Biological Circuit Diagram**
BDN	**(Functional) Block Diagram Notation**, including International Electrotechnical Commission (IEC) standards 61499 (Vyatkin, 2007)
BIP	**Behavior, Interaction, Priority** (BIP) modeling language
BON	**Business Object Notation** (Zamir, 1999)
BPMN	**Business Process Management Notation** (Kossak et al., 2014)
Chain Graph	**Probabilistic Network Chain Graph** (Kjaerulff & Madsen, 2008)
CLD	**Causal Loop Diagram** (Kim, 1992)
CM	**Concept Map**, also called Mind Map
CS	**Construction Symbols** for systems, or Utility Graffiti
DAG	**Directed Acyclic Graph**, or Acyclic Digraph
DFD	**Data Flow Diagram**
DBS-VN	**Declarative Behaviour Specification Visual Notation** (Kühne, 2011)
DRAKON	**DRAKON Visual Language** (Parondzhanov, 1995)
DSRP	**Distinctions / Systems / Relationships / Perspectives** (Cabrera & Cabrera, 2015)
e³value	**e³value notation** (Hotie & Gordijn, 2017)
EEN	**Electrical Engineering Notation**
Energese	**Energy Systems/Circuit Language** (Odum & Odum, 2000)
EPC	**Event-driven Process Chain**
Flowchart	**Flowchart**
Flowgraph	**Flowgraph** (mathematics) or Flow Network
FHEN	**Fluid Hydraulics Engineering Notation**
FMC	**Fundamental Modeling Concepts** notation (Apfelbacher & Rozinat, 2003)
FODA-FD	**Feature-Oriented Domain Analysis Feature Diagrams** (also called Original Feature Trees, or OFTs; Schobbens, 2007)
FRAM	**Functional Resonance Analysis Method** (2019)
G-OWL	**Graphical Ontology Web Language**
Graham Chart	**Graham (Process) Chart** (Graham, 2004)
Graph Theory	**Graph Theory**, a branch of discrete mathematics (Trudeau, 1993)
GTM	**Graphical Topic Maps** (Thomas et al., 2008)
[M-]IDEF[Ø-14]	ICAM **[Integrated Computer Aided Manufacturing] DEFinition** includes several notation specifications (Laamanen, 1994), with IDEFØ based on SADT and M-IDEFØ being a modification (Serifi & Dašić, 2009)
iStar	**iStar or i***
KEGG	**Kyoto Encyclopedia of Genes and Genomes** Pathway/Wiring Diagram Notation (www.kegg.jp)
KiPN	**Knowledge-intensive Processes Notation** (Netto et al., 2019)
Machinations	**Machinations Game System Notation** (Adams & Dormans, 2012)
MEN	**Mechanical Engineering Notation**
MCD	**Markov Chain Diagrams**
MBWD	**Molecular Biology (Biochemical Pathway) Wire Diagram** notation (Michal & Schomburg, 2012).
[e]MIM	**Molecular Interaction Map** notation, with eMIM as the interactive version (Kohn et al., 2006)
mEPN	**modified Edinburgh Pathway Notation**, with mEPN3D as a very different three-dimensional version (Freeman et al., 2010)
NCD	**Neural Circuit Diagram**
ORM2	**Object-Role Modeling** version 2 (Halpin, 2015)
Petri Nets	**Petri Nets** (Petri, 1962)
P&ID	**Piping & Instrumentation Diagram** Notation
PPINOT	**Process Performance Indicators Notation** (del-Río-Ortega et al., 2019)
Process Map	**Process Map** (and roles: Salvati et al., 2023)
Railroad	**Railroad Diagram**
Railroad-S	**Railroad (Syntax) Diagram**
RALph	**Resource Assignment Language Graph** (Cabanillas et al., 2015)
REFAS	**Requirements Engineering For self-Adaptive Software** systems (Muñoz-Fernández et al., 2015)
Reo	**Reo Coordination Language** (Arbab, 2004)
S&FD	**Stock & Flow Diagram/Forrester Diagram** (Forrester, 1961)
Sankey	**Sankey Diagram**
SBGN	**Systems Biology Graphical Notation** (Le Novère et al., 2010)
SDL	**Specification & Description Language**, a variant of flowchart (Insert)
SC	Software Engineering **Structure Chart** (Martin & McClure, 1988)
Southbeach	**Southbeach Notation** (Smith & Burnett, 2011)
SIGN	**System Implementation Graphical Notation** (Rosin, 1977)
Simulink	**Simulink Block Diagram**
SOD	**System Object Diagram** (Booch, 1994)
STD	**State-Transition Diagram**
STDT	**Structured Analysis and Design Technique**
Subway Map	**Subway System Diagram/Map** (Vignelli, 1970)
SystemCSP	**System Communicating Sequential Processes** Notation (Insert)
UML	**Unified Modeling Language**, with Systems Modeling Language (SysML) as mostly a subset
Unit VPS	**Unit Visual Programming System** (Timbó, 2021)
VNDBS	**Visual Notation for Declarative Behavior Specification**
VSM-N	**Viable Systems Model Notation** (Beer, 1995)
Wardley Map	**Wardley [Value Chain] Map** (Wardley, 2018)
YAWL	**Yet Another Workflow Language** (Van der Aalst et al., 2003)

sources

Ernest Adams & Jois Dormans, *Game Mechanics: Advanced Game Design* (Berkeley, CA: New Riders, 2012). Rémy Apfelbacher & Anne Rozinat, *FMC Notation Reference* [Appendix Version] (Kaiserslautern: FMC Group, 2003). Farhad Arbab, "Reo: a channel-based coordination model for component composition," *Mathematical Structures in Computer Science*, vol. 14, no. 3 (2004), pp. 329-366. Stafford Beer, *Brain of the Firm*—Second Edition (Hoboken, NJ: Wiley, 1995). Grady Booch, *Object-Oriented Analysis and Design*—Second Edition (New York, NY: Addison-Wesley, 1994). Cristina Cabanillas et al., "RALph: A Graphical Notation for Resource Assignments in Business Processes, Jelena Zdravkovic et al., eds., *Advanced Information Systems Engineering, Proceedings of the 27th International Conference, CAiSE 2015* (Chem: Springer, 2015), pp. 53-68. Derek Cabrera & Laura Cabrera, *Systems Thinking Made Simple: New Hope for Solving Wicked Problems* (Ithica, NY: Odyssean Press, 2015). Radovan Cervenka & Ivan Trencansky, *The Agent Modeling Language – AML* (Basil: Birkäuser, 2000). Peter H. Feiler et al., "The Architecture Analysis & Design Language (AADL): An Introduction," *Technical Note*, Carnegie Mellon University, Software Engineering Institute, no. 011 (2006). Jay Wright Forrester, *Industrial Dynamics* (Cambridge, MA: Productivity Press, 1961). Tom C. Freeman et al., "The mEPN scheme: an intuitive and flexible graphical system for rendering biological pathways," *BMC Systems Biology*, vol. 4, art. no. 65 (2010), pp. 1-13. Ben B. Graham, *Detail Process Charting* (Hoboken, NJ: John Wiley & Sons, 2004). Adela del-Río-Ortega et al., "Visual PPINOT: A Graphical Notation for Process Performance Indicators," *Business Information Systems Engineering*, vol. 61, no. 2 (2019), pp. 137-161. Terry Halpin, *Object-Role Modeling Fundamentals* (Basking Ridge, NJ: Technics Publications, 2015). Takayuki Hirose & Tetsuo Sawaragi, "Development of FRAM Model Based on Structure of Complex Adaptive Systems to Visualize Safety of Socio-Technical Systems," *IFAC PapersOnLine*, no. 52-19 (2019), pp. 13-18. Felicia Hotie & Jaap Gordijn, "Value-Based Process Model Design," *Business Information Systems Engineering*, vol. 61, no. (2), pp. 163-180. Daniel H. Kim, "Guidelines for Drawing Causal Loop Diagrams," *The System Thinking*, vol. 3, no. 1 (1992), pp. 5-6. Uffe B. Kjaerulff & Anders L. Madsen, *Beyesian Networks and Influence Diagrams: A Guide to Construction and Analysis* (Chem: Springer, 2008). Kurt W. Kohn et al., "Molecular interaction maps of bioregulatory networks: a general rubric for systems biology," *Molecular Biology of the Cell*, vol. 17, no. 1 (2006), pp. 1-13. Felix Kossak et al., *A Rigorous Semantics for BPMN 2.0 Process Diagrams* (Cham: Springer, 2014). Thomas Kühne, "A Visual Notation for Declarative Behaviour Specification," *Electronic Communications of the EASST*, vol. 42 (2011), pp. 1-10. Yohei Kurata and Max J. Egenhofer, "Structure and Semantics of Arrow Diagrams," *Lecture Notes in Computer Computer Science*, no. 3693 (2005), pp. 232-250. Mary Laamanen, "The IDEF standards process modeling standard," in Alex Verrijn-Stuart and T. WIlliam Olle, eds., *Methods and Associated Tools for the Information Systems Life Cycle* (Amsterdam: Elsevier, 1994). James Martin & Carma McClure, *Structured Techniques*—Revised Edition (Englewood Cliffs, NJ: Prentice Hall, 1988), pp. 181-190. Juan C. Muñoz-Fernández, Gabriel Tamura, Raúl Mazo, and Camille Salinesi, "Towards a Requirements Specification Multi-View Framework for Self-Adaptive Systems," *CLEI Electronic Journal*, vol. 18, no. 2 (2015), paper 5. Joanne Manhães Netto et al., "KiPN: A Visual Notation for Knowledge-intensive Processes," *International Journal of Business Process Integration and Management*, vol. 9, no. 3 (2019), pp. 197- 219. Gerhard Michal & Dietmar Schomburg, *Biochemical Pathways*—Second Edition (Hoboken, NJ: John Wiley & Sons, 2012). Nicolas Le Novère et al., "System Biology Graphical Notation: Entity Relationship Language Level 1—Version 1.1," *Nature Proceedings* (2010). V. D. Parondzhanov, "Visual Syntax of the DRAKON Language," *Programming and Computer Software*, vol. 21, no. 3 (1995), pp. 142-153. Howard T. Odum & Elisabeth C. Odum, *Modeling for All Scales: An Introduction to System Simulation* (San Diego, CA: Academic Press, 2000). Robert F. Rosin, "A Graphical Notation for Describing System Implementation," *Software—Practice & Experience*, vol. 7 (1977), pp. 239-250. Salvati et al., "A picture is worth a thousand words: advancing the use of visualization tools in implementation science through process mapping and matrix heat mapping," *Implementation Science Communications*, vol. 4, no. 43 (2023), pp. 1-15. Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps, "Generic Semantics of Feature Diagrams," *Computer Networks*, vol. 51 (2007), pp. 356-479. Howard Smith & Mark Burnett, *The Elements of Southbeach Notation 0.9.6* (London: Southbeach Solutions, 2011). Hendrik Thomas et al., "GTM alpha: Towards a Graphical Notation for Topic Maps," in Lutz Maicher and Lars Marius Gershol, eds., *Subject-centric Computing, Proceedings of the Fourth International Conference on Topic Maps Research and Applications* [TMRA] (Leipzig: Leipziger Beiträge zur Informatik, 2008), pp. 113-128. Veis Serifi & Predrag Dašić, "Functional and Information Modeling of Production using IDEF Methods," *Strojniški vestnik – Journal of Mechanical Engineering*, vol. 55, no. 2 (2009), pp. 131-140. Samuel Timbó, "Unit" [software repository], www.github.com/samuelmtimbo/unit (2021). Richard J. Trudeau, *Introduction to Graph Theory* (New York, NY: Dover, 1993). Van der Aalst, Wil, Arthur ter Hofstede, Bartosz Kiepuszewski, and Alistair P. Barros, "Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages," *Distributed and Parallel Databases*, vol. 14, no. (2003), pp. 5-51. Massimo Vignelli, *System Map: New York Subway Diagram* [poster] (New York, NY: New York City Transit Authority, 1970). Valeriy Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design* (Research Triangle Park, NC: Instrumentation, Systems, and Automation Society, 2007). Jonas Wahl & Jakob Runge, "Foundations of Causal Discovery on Groups of Variables," preprint manuscript (2023). Kim Waldén, "Business Object Notation," in Saba Zamir, ed., *Handbook of Object Technology* (Boca Raton, FL: CRC Press, 1999), pp. 193-204. Simon Wardley, *Wardley Maps* [In progress: www.medium.com/wardleymaps] (2018).

SYSTEMVIZ

Author. Peter Stoyko is an interdisciplinary social scientist and information designer who studies systems, culture, foresight, and governance.

Learn more about the SystemViz project and download materials at:

www.systemviz.com