Systems Design: SysML vs. Flowthing Modeling

Sabah Al-Fedaghi
Kuwait University
sabah@alfedaghi.com

Abstract

In model-based systems engineering, a system is depicted graphically and textually at various levels of granularity and complexity. For this purpose, Systems Modeling Language (SysML) is designed to support development stages in systems engineering applications, including specification, analysis, design, and validation. In comparison with UML, SysML is said to be easier to learn and apply, and it makes it possible to generate specifications in a single language for heterogeneous teams working on realization of blocks in system hardware and software. This paper studies the utilization of SysML in an integrated systems and software development process based on the Rational Integrated Systems/Embedded Software Development Process known as Harmony. This process follows the classic development scheme that starts with requirements and functional analysis and continues to the phases of design synthesis, software design, and implementation, through to final system acceptance. This paper focuses on requirements and functional analysis, offering an alternative new model, called the Flowthing Model (FM), that captures the dynamics of requirements in the system. The claim is that FM representation provides a viable alternative to the use case/activity diagram procedure at the base of the design foundation in SysML. In this paper, the same study case is used to compare the two representations side by side, allowing the reader to validate the claim that FM provides an easy and unified conceptual map of the system without a multiplicity of graphs. This result is a critical outcome, because changing the foundational description would change the entire development process.

Keywords: conceptual model, SysML, model-based systems engineering, requirements specification, design analysis

1. Introduction

Systems engineering is an interdisciplinary field that develops complex systems through software and hardware solutions, focusing on the best ways to design and manage projects over their life cycles. It aims at ensuring that “customer and stakeholder’s needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system’s entire life cycle” [1-2].

Development of complex systems relies on modeling and simulation methodologies (e.g., data flow diagrams, functional flow block diagrams) to develop functional specifications, data flow descriptions, and system structure definitions [3]. The resultant specifications produce a large number of documents, including a multiplicity of diagrams and representations that lead to confusion and difficulty in managing processes, as well as inconsistent usage [3]. A more methodological approach is the scheme known as model-based systems engineering, in which the system is specified at various levels of granularity (operational, functional, and technical), reducing its complexity, since each model and diagram provides an abstracted view and definition of the system [3]. “The greatest benefit of a model-driven process is improved
communication between engineering disciplines, between technical and non-technical parties, using different levels of abstraction, and avoids information overload” [4].

A model-based approach requires use of modeling concepts and tools to produce a coherent system model. Systems Modeling Language (SysML) was created to realize this approach [4-7]. It is based on Unified Modeling Language (UML) but addresses systems engineering needs and is more suitable “to analyze, specify, design, and verify complex systems, … to enhance systems quality, improve the ability to exchange systems engineering information amongst tools, and help bridge the semantic gap between systems, software, and other engineering disciplines” [8]. SysML was developed by the Object Management Group (OMG) [9-10], International Council on Systems Engineering (INCOSE), and AP233 (early requirement analysis) of International Organization for Standardization (ISO), using UML’s profile mechanism that represents a subset of UML 2 with extensions. It is designed to support development stages, including the specification, analysis, design, and validation of systems engineering applications. “In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models” [9].

SysML involves modeling of blocks instead of classes in UML, and is said to be easier to learn and apply, adding two new diagrams (requirement and parametric diagrams) to the seven included from UML, for a total of nine diagram types (UML diagrams include activity diagram, sequence diagram, state diagram, use case diagram, block definition diagram [replaces UML2 class diagram], internal block diagram [replaces UML2 composite structure diagram], and package diagram) with flexible and more expressive semantics than UML.

SysML makes it possible to generate specifications in a single language for heterogeneous teams, dealing with the realisation of the system hardware and software blocks. Knowledge is thereby captured through models stored in a single repository, enhancing communication throughout all teams. In the long term, blocks can be reused as their specifications and models enable suitability assessment for subsequent projects. [3]

This paper focuses on the early phases of applying SysML in a sample software development process: the Rational Integrated Systems/Embedded Software Development Process Harmony [11]. As claimed in [11], SysML achieves only marginal success as a modeling tool in this development process, mainly because of the Harmony process itself more than with problems with SysML, but also because of a multiplicity of fragmented representations in UML that are exported to SysML, including narratives, diagrams, and notions. SysML lacks a nucleus around which different phases of the development process can evolve, analogous to the role of a blueprint for a complex project such as a high-rise building, where it would be the core around which the building’s framework, an electrical system, water system, and interior walls are built by their various specialists.

One way to substantiate such a claim is to produce a sample viable core and contrast it with SysML. The paper proposes such an alternative tool, called the Flowthing Model (FM) [12–16], and demonstrates its viability in the requirements and design phases. To provide a background for this thesis, and for the sake of completeness, FM will be briefly described in Section 2. The example developed there, of a water distiller, is a new contribution. Section 3 reviews the SysML design process in the context of
Harmony. Section 4 recasts the study case of a security system given by Hoffmann [17] in terms of FM in order to compare the two methodologies side by side.

2. Flowthing Model

The Flowthing Model (FM) is a uniform method for representing “things that flow,” called flowthings. Flow in FM refers to the exclusive (i.e., being in one and only one) transformation among six states (also called stages): transfer, process, creation, release, arrival, and acceptance, as shown in Figure 1. We will use Receive as a combined stage of Arrive and Accept whenever arriving flowthings are always accepted.

![Figure 1. Flowsystem](image)

The fundamental elements of FM are as follows:

**Flowthing**: A thing that has the capability of being created, released, transferred, arrived, accepted, and processed while flowing within and between “units” called spheres.

A flow system (referred to as flowsystem) is a system with six stages and transformations (edges) between them. The flow notion in FM corresponds to “atomic flow” in SysML, in which only a single type of input or output is specified. Strangely, flows in SysML can be discrete (flowthings), streaming, or control, but how does “control” flow? It is not a flowthing. In FM, flows can be controlled by the progress (sequence) of the stream of events (create, release, transfer within the next sphere, receive, …), or by triggering that can initiate a new flow. For example, in a computer program, control does not flow through instruction; rather, instruction flows through the control sphere.

**Spheres and subspheres**: These are the environments of the flowthing, such as, e.g., a company, a computer, and a person. Spheres are analogous to blocks in SysML. So-called ports form the Transfer stage of a flowsystem. Note that a sphere may not have the function of Transfer (hence, ports) explicitly. A sphere can include the sphere of a flowsystem that includes the Transfer stage. Thus a flowsystem can be considered a block in SysML where all ports are collected in the Transfer stage. Additionally, this stage can include other roles such as queuing and checking of input and output.

**Triggering**: Triggering is a transformation (denoted by a dashed arrow) from one flow to another, e.g., a flow of electricity triggers a flow of air.

In addition to the fundamental characteristics of flow in FM, the following types of possible operations exist in different stages:

1. **Copying**: Copy is an operation such that flowthing \( f \rightarrow f \). That is, it is possible to copy \( f \) to produce another flowthing \( f \) in a system \( S \). In this case, \( S \) is said to be \( S \) with copying feature, or, for short, Copy \( S \). For example, any informational flowsystem can be Copy \( S \), while physical flowsystems are non-copying \( S \). Note that in Copy \( S \), stored \( f \) may have its copy in a non-stored state. It is possible that copying is allowed in certain stages and not in others.
2. **Erasure:** Erasure is an operation such that \( f \Rightarrow e \), where \( e \) denotes the empty flowthing. That is, it is possible to erase a flowthing in \( S \). In this case, \( S \) is said to be \( S \) with erasure feature, or, for short, Erasure \( S \). Erasure can be used for a single instance, all instances in a stage, or all instances in \( S \).

3. **Canceling:** Anti-flowthing \( f^- \) (\( f \) with superscript \( - \)) is a flowthing such that \( (f^- + f) \Rightarrow e \), where \( e \) denotes the empty flowthing, and \( + \) denotes the presence of \( f^- \) and \( f \).

   It is possible for anti-flowthing \( f^- \) to be declared in a stage or a flowsystem. If flowthing \( f \) triggers the flow of flowthing \( g \), then anti-flowthing \( f^- \) triggers anti-flowthing \( g^- \).

   An example of the use of these FM features is erasure of a flow, as in the case of a customer who orders a product, then cancels the order, an action that might require the cancellation of several flows in different spheres triggered by the original order.

   Formally, FM can be specified as \( FM = \{S_i, \{F_{ij}\}, T\}, \{(F_{ij}, F_{ij})\}, 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq l \leq t\} \)

   That is, FM is a set of spheres \( S_1, \ldots, S_n \), each with its own flowsystems \( F_{ij}, \ldots, F_{im} \). \( T \) is a type of flowthing \( T_1, \ldots, T_t \). Also, \( F \) is a graph with vertices \( V \) that is a (possibly proper) subset \{Arrive*, Accept*, Process*, Create*, Release*, Transfer*\}, where the asterisks indicate secondary stages. For example, \{Copy, Store, and Destroy\} can represent these secondary stages.

**Example:** Friedenthal *et al.*, [5] describe the application of SysML to the design of a water distiller:

The packages in this model are primarily organized based on the types of artifacts developed using the selected process, including requirements use cases, behavioral and structural models. The Engineering Analysis package includes the constraint blocks and parametric models used to analyze performance.

Flows in and out of the Distiller have been depicted using item flows typed by appropriate item types (\( H_2O, Heat \)) contained in Item Types package [5].

The initial decomposition of the distiller divides it into Heat Water, Boil Water, and Condense Water. The flow through a water distiller is shown in the activity diagram of Figure 2.

**Fig. 2. Activity Diagram Describing Flow through a Water Distiller**
Each action ends before the next one begins; when Heat Water is complete, the action Condense Steam and Drain Residue are initialized. When these actions are complete, the Distill Water activity is complete, and a patch of pure water is available.[5]

Figure 3 shows the FM representation corresponding to the activities depicted in Figure 2. In the Figure, the sphere of Heat Water has two flowthings that pass through it: Cold dirty liquid and Heat (circles 1 and 2 in the figure). Receiving heat (3) triggers (5) processing (heating) of water (6) to cause (generate/create) Hot dirty water (7). The hot dirty water flows to the sphere Boil water (8), where it is heated (9), triggered (10) by receiving (11) additional heat. Also in the Boil water sphere, Steam is created (12) that flows to Condense water (13), where it is processed/condensed (14) to trigger the generation of Pure water (15) and Waste heat (16), which both flow to the outside. Also, processing of hot dirty water in Boil water (9) triggers the creation of Residue (17) that flows to the outside.

The FM representation is a conceptual picture of the distiller in which the coordinated flows of flowthings drive the progression, like parts of a dynamo that transfer activation from one piece to another. Synchronization, logical operations, and constraints can be superimposed on this core conceptual picture. By contrast, the activity diagram mixes categorically different flowthings, scatters spheres (port of the act vs. port of the block), and hides (at this level) the internal processes of blocks because of a lack of generic stages, thus delegating the description to another diagram, causing fragmentation of specifications.

The remainder of this paper will concentrate on applying SysML in a sample software development process: Harmony [11], which moves from a requirements narrative to a diagrammatic conceptual representation. In Harmony, the narrative is given in piecemeal fashion to construct a conceptual design based on multiple diagrams. The FM approach constructs a conceptual representation such as Fig. 3 from input submitted in different forms, in the style of an architectural blueprint that forms the base for any subsequent processes.
3. SysML Design Process

How to apply SysML in an integrated systems and software development process? Hoffmann [17] undertakes such a project based on Rational Harmony for Systems Engineering [11]. The Harmony process facilitates the transition from systems engineering to software engineering by using SysML for system representation and specification based on a scenario-driven and iterative development process, and promotes reuse of test scenarios throughout system development [4]. It follows the classic “V” development diagram that starts with the focus of this paper: requirements, then functional analysis, and continues through the phases of design synthesis, software design, and implementation, then moves up the right-hand side of the “V”, the integration phases, from unit test to final system acceptance [17].

Stakeholder requirements are translated into system requirements that define what the system must do (functional requirements) in the form of “shall” statements linked to stakeholder requirements for the purpose of traceability. The Requirement Diagram in SysML visualizes the functional and nonfunctional requirements and relates them to each other and to design elements, test cases, and stereotypes for requirement traceability [4].

The next major step is the definition of use cases.

A use case describes a specific operational aspect of the system... It specifies the behavior as perceived by the actors (user) and the message flow between the actors and the use case. An actor may be a person, another system or a piece of hardware external to the system under development... [17]

A functional analysis is based on use cases; each system-level use case is defined in an Internal Block Diagram that includes instances of empty and unlinked SysML blocks.

The next step in the development process is to capture the behavior of the use case block by means of diagrams, including an activity diagram.

The activity diagram – referred to as Use Case Black-Box Activity Diagram – describes the overall functional flow (storyboard) of the use case. It groups functional requirements in actions – in Harmony for Systems Engineering the equivalent of operations – and shows, how these actions/operations are linked to each other. [17]

This paper focuses on this portion of the process, offering an alternative FM representation that moves beyond requirements analysis to capture the flows and interactions in the system under consideration.

4. Study Case

Requirements analysis starts with importing stakeholder requirements specifications, using different communication methods. Figure 4 shows stakeholder requirements specification for a security system study case [17]. To illustrate basic characteristics of FM, a study case representation will be developed from the initial description, then the same FM diagram will be carried further by adding details, the way one would draw a map using layers.

Beginning with the system summary (SS1 shown in Figure 4) as input, Figure 5 shows the initial FM representation.
Security System – Stakeholder Requirements Specification

System Overview
SS1: System Summary
A security system is to be developed that controls entry and exit to a building through a single point of entry. Identification of personnel will be made by two independent checks. Each person will be photographed upon entry and their time in the building monitored.

Nominal System Specification

SS1: Security Checks
Secure areas are to be protected by two independent security checks, one based upon an employee ID and one based upon biometric data. Access to secure areas will be unavailable until the users ID is confirmed. The time between the two independent security checks will not exceed a configurable period. The user is allowed three attempts at biometric and/or card identification before access is completely disabled.

Identification Requirements

SS11: Security Card
Access will be denied to any user unless he has a valid Security Card. Security cards only contain the employee name and ID. Out of date security cards cause a denial of access.

SS12: Biometric Scan
Access will be denied to any user unless their biometric data is recognized. The biometric data is to be stored in the system database and not on the security card.

SS13: Exit requirements
The system will only process one user at a time, giving them sufficient time to enter and exit the area before automatically securing itself.

Personnel Monitoring

MA1: Image Capture
An image is to be taken of any person, at the first attempt, when trying to access a secure area for logging time and employee ID.

MA2: Time monitoring
The time a user spends in a secure area is to be recorded. An alarm will notify administration/security if a person stays longer than 10 hours in the secure area.

Emergency Requirements

E1: Emergency Exit
In the event of an emergency the administrator can invoke a "Free Exit Mode". All security checks for exiting the area are to be disabled until the administrator returns the system to normal working.

E2: Security Lockdown
The administrator can invoke a security lockdown mode - in this event the system will lock all access points until the administrator returns the system to normal working.

Figure 4. Partial view of Stakeholder Requirements Specification (from [17]). Some Minor Steps such as Reporting to Administration have been Omitted

Figure 5. FM Representation Corresponding to System Summary SS1

Figure 5 includes the Physical entrance and Building spheres. As a person comes through the entrance (1), arrival (2) triggers security checks:

- Triggering (3) causes creation of a photograph that is processed (4) to determine whether to trigger rejection (5 and 12), or acceptance (6) and subsequent ID checking.
- With ID checking triggered (7), the person is asked to enter his/her ID (8). Processing of the ID (9) triggers either acceptance (10) or rejection (11, 12).
Triggering, in these cases, can be in the form of signals sent to a routine that controls \textit{Arrival} and checks whether the two security checks are valid. As stated, many notions can be superimposed on the basic FM representation such as diagonal signs that denote logical operations OR and AND. If the person satisfies both security constraints (13), he/she progresses to the acceptance stage (14). This triggers registering of the time (15) of entering the secured area (16) where he/she is processed (17), \textit{e.g.}, interviewed. The person then proceeds back to the entrance (18), where exit time is registered (19), then outside (12).

The process of drawing Figure 5 is not difficult, starting with the given SS1 System Summary:

A security system is to be developed that controls \textit{entry and exit} to a building through \textit{a single point of entry}. Identification of personnel will be made by \textit{two independent checks}. Each person will be \textit{photographed} upon entry and their \textit{time in the building} monitored. (italic added)

One can roughly sketch the flows in the FM representation (Figure 6) and then develop the model completely.

![Figure 6. Rough Sketch of FM Representation that Corresponds to the System Summary SS1](image)

Returning to the use case analysis by Hoffmann [17], the following is a sample of items in the system specification.

SS112: Biometric Scan: Access will be denied to any user unless their biometric data is recognized. The biometric data is to be stored in the system database and not on the security card.

SS12: Access Priority and Time: The system will only process one user at a time, giving them sufficient time to enter and exit the area before automatically securing itself.

SS13: Exit requirements: The user is not allowed to exit until the security card has been successfully authorized.

MA2: Time monitoring: The time a user spends in a secure area is to be recorded. An alarm will notify administration/security if a person stays longer than 10 hours in the secure area.

E1: Emergency Exit: In the event of an emergency the administrator can invoke a "Free Exit Mode". All security checks for exiting the area are to be disabled until the administrator returns the system to normal working.

E2: Security Lockdown: The administrator can invoke a security lockdown mode - in this event the system will lock all access points until the administrator returns the system to normal working.

Figure 7 shows an expanded FM representation that incorporates these requirements. The original circled numbers shown in Figure 5 are not shown in the new figure.

First, consider the requirement,
SS12: **Access Priority and Time:** The system will only process one user at a time, giving them sufficient time to enter and exit the area before automatically securing itself.  

This requirement is represented at circle 20 in Figure 7, where persons are queued to enter the Arrival area, which includes the constraint: *one at a time from queue* (21). As indicated earlier, constraints can be superimposed on the underlying FM representation. Note that the exit of a person triggers permitting the next person in queue to proceed to Arrival. While SysML uses the separate notion of compartment to describe block characteristics such as constraints, FM integrates them into the flow.

![Figure 7. FM Representation Expanded According to New Requirements](image)

Next consider:

**SS112: Biometric Scan:** Access will be denied to any user unless their biometric data is recognized. The biometric data is to be stored in the system database and not on the security card.

Circle 22 in Figure 7 indicates that biometric data is retrieved from a database to be compared (processed) with the photo just taken. Then consider,
**MA2: Time monitoring:** The time a user spends in a secure area is to be recorded. An alarm will notify administration/security if a person stays longer than 10 hours in the secure area.

At circle 23, a person who leaves the protected area to return to the entrance triggers (24) registering his/her time of exiting; the data flows to be processed (25) and compared (diamond shape) with the previously registered time of entering the protected area. If the time is greater than 10 hours, this triggers the alarm (26). Further:

**SS13: Exit requirements:** The user is not allowed to exit until the security card has been successfully authorized.

The person who is exiting submits his/her ID card (27); ID information is then processed (28) to trigger the person’s release (29). Interestingly, the requirement does not say what to do if the ID information is rejected at this point!

**E1: Emergency Exit:** In the event of an emergency the administrator can invoke a "Free Exit Mode". All security checks for exiting the area are to be disabled until the administrator returns the system to normal working.

**E2: Security Lockdown:** The administrator can invoke a security lockdown mode - in this event the system will lock all access points until the administrator returns the system to normal working.

Free Exit Mode is a state of the system, and a *state* is a flowthing; thus, in an emergency, the *Administrator* (box at the bottom) can invoke (create – circle 30) a Free Exit state. This state triggers:

- Unconstrained transfer through the entrance (31), *e.g.*, open gate.
- Disabling of operations at the entrance, such as recording of exit time (32).
- Disabling of ID checking (33).
- Unconstrained transfer to the outside (34).

Note that the flowsystem of Free Exit Mode is a single stage that can create or de-create (35) this state. De-creating would trigger nullification of the above four actions.

Similar situations are modeled for Lockdown Mode (top box in the figure).

Next, the following sets of new requirements will be incorporated into the FM representation, as shown in Figure 8. This time only the entrance sphere is shown in the figure. Consider the requirements:

**SS11-8 Three Attempts On Employee ID Entry:** Upon entry the user shall be allowed three attempts on card identification.

**SS11-8 Three Attempts On Biometric Data Entry:** Upon entry the user shall be allowed three biometric data entries.
The processing in Photo and ID includes checking for the three-time constraints (36 and 37, respectively) in addition to checking the photograph and ID of the arriving person. Note that a joint bar (transition node in Petri nets terminology) is used to indicate that both conditions of “validity/not out of date” and “three times” are required to trigger acceptance. Again, such synchronization notions are imposed on the core FM representation.

Consider the following,

**SS11-5: Time Between Two Independent Checks:** The time between the two independent security checks shall not exceed a configurable period.

This is modeled in the *Check Time* sphere (38), where taking a photograph or entering an ID triggers (39 and 40, respectively) recording (creating) the time of the event. If the difference exceeds the configurable period (41), the person is rejected (42). Otherwise, flow of the biometric data from the database is triggered (43) to continue its processing. Note that this is a type of synchronization; the biometric checking proceeds (is triggered) only if the difference in time does not exceed the configurable period.

Continuing further, the following set of new requirements can be incorporated into the FM representation, as shown in Figure 9:

**SS11-10 Denied Entry Notification:** Any denied access attempt shall be logged and account details sent to the administrator

**SS11-12 Alarm – Entry** On a denied entry an alarm signal shall be raised.

**SS11-13 Alarm – Exit** On a denied exit an alarm signal shall be raised.

These requirements are modeled in Figure 9 as triggering set 44–48 and using the wide bar notation to indicate that when any triggering occurs, it will trigger the following:
- setting the alarm (49), and
- releasing the person to the outside (50).
Figure 9. Another FM Representation Expanded According to New Requirements

Note that for simplicity’s sake, the method of counting of the number of times for biometric checking (Photo) is represented by the curved dashed arrow (to left of circle 51) to indicate a change in the database. This can be represented, in a less compacted representation, by having Process in Photo trigger a flowthing (an independent box), called “register number of times”, that retrieves the value of number of times, adds one to it, then stores it back in the database. The point here is that flowsystems can be generated to any level of detail, just as with the blueprint of a building in which, say, some interior partitions are marked but not completely represented in the drawing. Fig. 9 presents a compact and convenient level of description; a more complete representation can be developed in real cases.

5. Discussion

In this paper, the same study case is applied to compare the two representations, SysML and FM, side by side. This comparison has critical consequences, because changing the foundation description would cause alterations in the entire development process. FM can be taken as a pivot (see Figure 10) for the development process life cycle, where its representation is constructed in ordered layers of detail to deal with complexity.
Figure 10. Proposed Role of FM Representation in Development Life Cycle

To illustrate the idea behind different layers of the FM representation shown in Figure 10, consider a specification of the user interface (in the overview, in the middle of Figure 10) for the portion that models the time monitoring requirement:

MA2: Time monitoring: The time a user spends in a secure area is to be recorded. An alarm will notify administration/security if a person stays longer than 10 hours in the secure area.

This requirement is represented first in Figure 7 and then magnified in Figure 11 over the basic FM representation in the background. The modeler would ask at a certain phase of the design how to present creation of Alarm at the user interface. Thus, in the user interface layer, the creation of an alarm is presented to the security administrator as the screen shown at the top left (circle 1). The screen is designed in terms of the FM model where the alarm is received, processed, released, and transferred. The security administrator, upon receiving the alarm, can inspect the details (2) or review the constraint that applies to this alarm (3). The alarm can also be processed, when the security administrator can activate searching the building (4), lock down the entrance (5), or turn it off (6), or he/she can release and transfer the alarm to someone else (7).
Suppose that the security manager selects *Search the building* (4); the SEARCH BUILDING screen (8) would appear. Assuming that areas of the building are monitored by cameras, selecting the *Transfer area* (9) would show a camera view of the entrance area of the secured building. Selecting the *Receive area* (10) would show a camera view of the reception area, *Process area* (11), a camera view of the business area, and *Release area*, a camera view of the area leading to the exit. Note the uniformity in the application of FM, where stages can be applied all the way from the requirements level to the user interface level.

Of course such a new venture needs a great deal of development. One objective of this paper is to raise interest among researchers in such an approach. Currently, in further research, a project is under way to build a small-scale, Harmony-like system based on FM.

### 6. Conclusion

This paper has focused on the utilization of SysML in the Rational Integrated Systems/Embedded Software Development Process *Harmony*, especially the requirements and functional analysis phases. It offers an alternative new model, FM, that captures requirements behavior in the system. The claim is that, as a modeling tool in this development process, SysML achieves only marginal success because of its multiple and fragmented representations. SysML lacks a nucleus around which different phases of the development process can evolve. In addition, one of the alleged advantages of SysML over UML, in terms...
of better, more efficient communication between technical and nontechnical parties, is debatable. It seems that FM has an edge in this context.

References
