28th annual INCOSE international symposium
Washington, DC, USA
July 7 - 12, 2018

Using Board Games as Subject Matter for Developing Expertise in Model-Based Systems Engineering

Matthew Corrado
Georgia Institute of Technology
mattcorrado98@gmail.com

Kathryn Trase
NASA Glenn Research Center
216–433–8067
Kathryn.trase@nasa.gov

Copyright © 2018 by Matthew Corrado and Kathryn Trase. Published and used by INCOSE with permission.

Abstract. As more organizations transition from traditional document-centric systems engineering to a model-based approach, many are challenged to train their staff in new languages, tools, and methodologies, while managing the expectations of stakeholders and their expected model outcomes. In particular, challenges associated with learning a new modeling language and developing skills in the 'art' of modeling present organizations with formidable obstacles to realizing this transition. This paper hypothesizes that systems engineers may more readily learn how to correctly model with SysML, and develop intuition about the art of modeling and using patterns, if their learning references a commonly and thoroughly-understood subject, such as a board game. This paper presents a case for the use of board games as subject matter for new modelers. It demonstrates the concept with a sample model of Hasbro's popular board game, Monopoly, and discusses the limitations of this approach and potential adaptations that may broaden the applicability of the learned skills to projects. Finally, results from a small feasibility assessment and concepts for more formal study to evaluate the hypothesis are presented.

Introduction

As organizations continue to transition their systems engineering tools and methodology to more model-based methods, they face challenges in training staff in new languages, cultivating intuition in the 'art' of modeling, addressing stakeholder needs via the models, and integrating the model into current project management practices. In particular, challenges associated with learning a new modeling language and developing skills in the art of modeling present organizations with formidable obstacles to realizing this transition.

Training in the Systems Modeling Language (SysML) often references over-simplified sample models of subject matters, of which learners may only have a cursory understanding. For example, learners may know generally how an automobile (Friedenthal, Moore, & Steiner 2015) or spacecraft (Deligatti 2014) operates, but are not typically experts in all aspects of those systems' designs, or they may have different levels of understanding relative to their colleagues (whether fellow modelers or not). Pilot projects, such as that described by Vipavets, et al., often face challenges in proceeding to model at lower levels of detail than what was covered in initial training courses due to the limited depth of concept coverage.

SysML is the current de-facto language standard for a model-based approach, but has been identified as difficult to learn (Andersson et al. 2010) and difficult to specialize for a particular domain, especially by systems engineers who may not be trained in such topics as knowledge

management and ontology development. Some (Voirin et al. 2015) have embarked on an approach to develop custom, domain-specific languages (DSLs) to get around the hurdle of having to learn new terminology and syntax, and make adoption more user-friendly. However, if SysML is to be utilized until the development of community-accepted DSLs, then a learning approach that focuses specifically on the language and the subtleties of the 'art of modeling' is necessary.

While some modeling practitioners have cited the importance of modeling with a purpose and considering how the end stakeholder will use the information in the model (Hallqvist & Larsson 2016), there is a requisite learning period where new practitioners must simply be allowed to explore the language and associated techniques. Only after having grasped advanced aspects of the language and developing some confidence in successful modeling patterns can the new practitioner be able to think critically about a specific stakeholder and how to implement their concerns within the model content and structure.

Introductory aspects of SysML can be taught with top-level overviews of subject matters, but acquiring more advanced skill in applying the language and associated modeling techniques requires learners to have greater depth of understanding of the educational subject matter. By selecting a subject matter that learners understand thoroughly to the lowest level, training efforts can offer learning experiences that are more practically applicable to real project scenarios, including: how to model at different levels of abstraction while maintaining traceability between levels, or how to organize both the model and patterns to best achieve stakeholder objectives.

This paper makes the case for using board games as a commonly and thoroughly-understood subject matter for teaching the more subtle aspects of SysML and the art of modeling, followed by some example modeling of the game Monopoly by Hasbro, and finally, proposals for future investigations to validate the hypothesis of accelerated and deeper learning.

Premise: MBSE Learning Can Be Enhanced by Modeling Board Games

Board games may be suitable subject matter for new modeling practitioners to utilize when first learning to apply a modeling language and concepts. Most games are sufficiently multi-faceted in that several modeling language constructs must be utilized to capture the full nature of the game. These facets thus provide learners with opportunities to both apply several language constructs, as well as determine how to provide traceability between constructs (e.g., from requirements to behavior, or state machines to activities). While games may not necessarily be considered "complex" by the systems engineering community's technical definition, because they do not consist of several parts that combine and transmit data in sophisticated ways (Ferreira 2001), one could consider games to be sufficiently complicated and thus warrant methodical description, and potentially analysis, via modeling.

By using a game as the modeled subject matter, students could focus their learning only on modeling principles and techniques, without simultaneously trying to learn the subject (as may be the case when an automobile engineer attempts to learn SysML using a model of a spacecraft, for example). Peer tutoring and remote learning may also be more effective when familiar games are utilized as model subjects. Presuming two people have played the same game, it would be easier for the learners to intuit more subtle aspects of modeling rules and conventions by reviewing models created by someone else. Sample models of games, even if the learner has not generated the model him- or herself, may also prove more useful as an example or reference, given a

reasonably similar level of understanding of the subject matter. Learners would more readily be able to identify when a description of the game (as modeled) does not align with their understanding of its rules or behavior, and either discern a language or pattern subtlety, or identify a modeling error or cheater.

In addition to simply learning modeling languages and constructs, games may also provide an avenue toward learning more advanced modeling and simulation techniques. Game play could be modeled in the system model, and more complex probabilistic or mathematical models developed in external tools could be integrated with the system model to enable a higher-fidelity simulation. The learner benefits from already having a sense of the expected simulation outcome (does the simulation output match their experience in playing the game?), and can focus attention on learning the tools and integrated simulation techniques.

Modeling games can also provide an opportunity to develop skills that would be utilized in applying modeling techniques within a project environment. For example, skills in developing and generating documents from the model could be practiced: in this analogy, the rules of the game are to the players as the system requirements document is to the project. The learner can again focus on learning the mechanics of preparing content for document generation or stakeholder review, rather than also trying to learn the subject matter.

With a greater confidence in applying the language correctly and some intuition about the appropriate way to model a topic, the learner may be better equipped to solicit input from stakeholders regarding both the scope of the model and intended use of the information stored within the model. However, this paper does not attempt to provide guidance on how the systems engineer should decide which design aspects are the appropriate topics to model, in order to address stakeholder needs.

Characteristics of Games Suitable for Modeling

The wide variety of games and their diversity of game play mechanisms provides plenty of opportunities to apply different pillars of SysML, or emphasize some over others, and test modeling patterns in different scenarios. While games with a single mechanism, such as "roll-and-move" (found in Chutes and Ladders or Candyland, e.g.), may be too simple in concept to be instructive on the details of SysML or provide coverage of most of the language's pillars, games with several mechanisms quickly become sufficiently complex to be candidates for learning modeling techniques.

The gaming mechanisms themselves may be an interesting case study for learning how to develop patterns (in this case, the game mechanisms) and adapt known patterns to a specific instance (i.e., to a particular set of games that each use that pattern but perhaps in a variant form). Table 1 lists a few game mechanisms that may be extensible to real project scenarios and lists a few associated modeling constructs.

Other characteristics of games suitable for modeling include, as a subset or in combination with each other:

- Multiple players, game pieces or markers, and conditions for winning or losing
- Multiple types of interactions between players
- Variable player roles, levels of difficulty, rules, or initial game set-up or conditions

Table 1: Sample game mechanisms of interest for learning SysML and modeling techniques (Adapted from Board Game Geek (n.d.))

Mechanism	Description (adapted from Board Game Geek (n.d.))	Sample Modeling Constructs
Trading	Players interchange game pieces, tokens, currency, etc. between each other.	Interface definition and constraints, sequences
Set Collection	Players work to achieve a specified combination of items or conditions. Collecting or achieving a sets typically results in award of points or game resources, or wins the game.	Constraints, states, objective functions
Variable Player Powers	Different role capabilities or features are assigned to different players within the same game, such that each player might play to variations of the main rules.	Variants, constraints, requirements, states, use cases, allocation
Deck/Pool Building	Players start the game with a given deck or set of cards or pieces, and add and/or exchange those pieces over the course of the game, or multiple games. Some games allow additional items to be "purchased" and subsequently added to the set of cards. These new items typically enhance the abilities of the player or rules of the game in future sessions.	Architecture, context or domain definition, variants, constraints, use cases, allocation
Chit-Pull System	A player draws a random counter to indicate which group of units may be moved during that turn. In some cases, constraints may be placed on which type of units, the quantity of units, or possible behaviors of those units, that may be executed.	Activities, sequences, states, allocation

Monopoly Example

A thorough discussion of the intricacies of a SysML model of Monopoly necessitates a discussion of the qualities of Monopoly that make it a good candidate to model. First, the popular and widely understood board game exhibits most, if not all, of the characteristics described above. In essence, Monopoly is highly complex *and* widely understood, both of which need to be true of a game that is to be used as a learning and teaching tool. In fact, Monopoly is perceived as so simple that Hasbro recommends it for anyone who is at least eight years old, and most people are typically exposed to it at that age (Jingles 2013). The game's ubiquity affords it an advantage over similar games: Monopoly only seems simple because a large number of people have a pre-existing familiarity with it, as it is arguably one of the most popular board games of all time (Kismet 2017). Without this familiarity, Monopoly would reveal itself as a complex system with moving parts, competing objectives, interfaces, and more. Given the right amount of time, one could create a SysML model that describes Monopoly using all of the pillars and diagrams of SysML. However, for the purposes of this demonstration, many, but not all, of the possible modeling constructs are used.

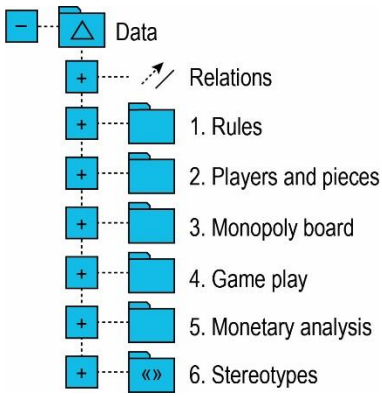

Figure 1. Monopoly model containment tree

The structure of the SysML model of Monopoly, referred to hereafter as "the Monopoly model," is intended to analogize that of a model of a real system. In other words, each section, or package, of the model represents a different concept or pillar that one would likely represent in a model of an actual system. For instance, most SysML models capture requirements, block definition representing the structure or architecture of the system, activity diagrams to describe the behaviors of the system, and more. Refer to Figure 1, a view of the containment tree of the Monopoly model. The Monopoly model was developed so that different aspects of the game of Monopoly represent different pillars of SysML. These relationships are discussed in detail below. Samples of Requirements, Block Definition, Activity, and Parametric elements corresponding to concepts in the game are provided, and discussion of potential future work for Use Case, Internal Block, Sequence, and State Machine elements follows.

Requirements

In the case of a game, the requirements (for play, game set-up, winning or losing, etc.) are found in the rulebook. Rules can be modeled in the same fashion as technical requirements, as demonstrated in Figure 2, which is an example requirements diagram that describes how the initial money provided to each player shall be divided. In this case, Rules 1.1 through 1.7 are derived from Rule 1, as indicated via the <<deriveReqt>> stereotype on the relationships. It would be instructive to the learner, and typical of a real project discussion, to consider which SysML requirement relationships should be used in the project and under which circumstances. In this case, one may consider applying the <<Refine>> relationship as an alternative and compare the options with project stakeholders to understand why one relationship type is preferred over another. Even requirements numbering could be evaluated by the learner to assess whether one numbering approach works better than another, and the applicable circumstances to which each approach applies.

Figure 2 also illustrates how different cross-cutting modeling techniques can relate various SysML pillars to each other. Figure 2 relates the act of providing the starting money to each player (an activity) to the corresponding requirement being satisfied: a common modeling pattern used to demonstrate all requirements have a specified function, and all functions relate to a specified requirement. Similarly, Figure 2 describes a block that defines the architectural element that satisfies the specified requirement; another modeling pattern. In a real engineering model, every requirement should be satisfied by some other element in the model. In the Monopoly model, it is

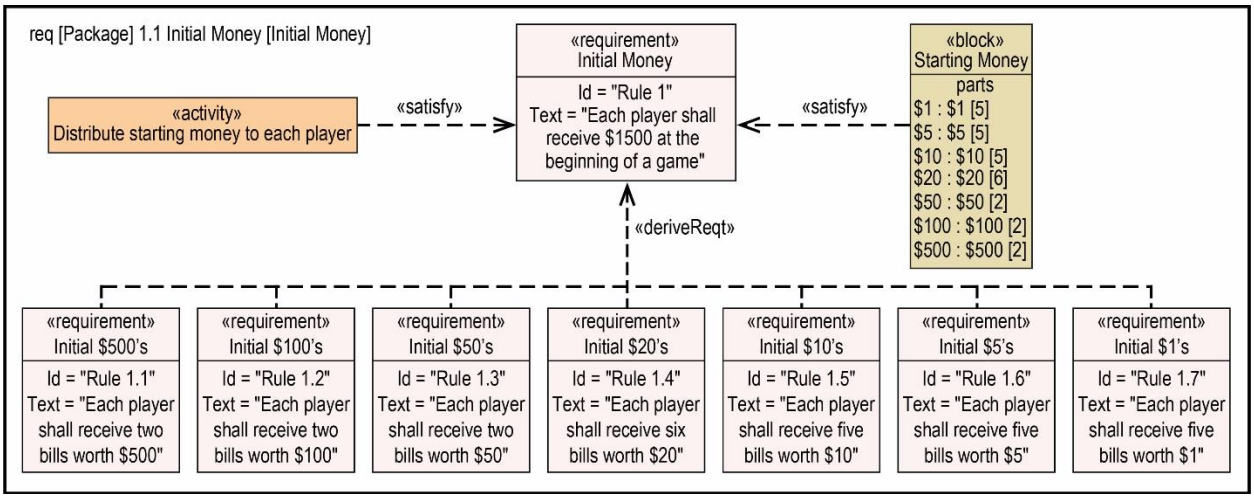

Figure 2. Sample requirements diagram describing rules of the game

not always possible to show, through some measurement or action, that a rule has been satisfied. However, there still are several cases in which this modeling technique can be practiced, as described above.

Requirements could also be developed to capture constraints of the game, just as a system must perform under a variety of operational constraints. For example, one rule in Monopoly could be that there shall be a certain finite quantity of money circulating throughout the game at a given time. This requirement could be satisfied by a constraint property representing the total amount of money in the game, which could be found by summing the total value that each player and the bank possesses.

Therefore, the rules of a game serve as an appropriate substitute for requirements in the scope of the game-system analogy. While the rules of games, Monopoly in particular, are not always extremely complex, there are a lot of opportunities to discuss subtleties, preferences, explore alternative modeling approaches, and create meaningful relationships between the rules of a game and other game characteristics. This process of modeling the rulebook of Monopoly then creates an excellent environment to practice modeling real requirements and refining modeling approaches to integrate them within the rest of the system model.

Block Definition

When modeling a game, the "structure" can be interpreted in more than one way and at multiple levels of detail, just as a system may utilize numerous variations of "structure and hierarchy" during the lifecycle (physical, functional, logical, organizational, etc.). The physical board and the illustrations or "spaces" on it contribute to the game's structure, but the players, cards, cash, and "tokens" should also be considered within the game's context to create a complete model. Figure 3 defines each of these aspects as being part of the game's context.

There are multiple ways the context described in Figure 3 could be represented, and it would be instructive to the learner to explore these options and the rationale for choosing one approach over another. Ideally, the learner could develop an intuition to know whether they should model with one technique over another, according to their overall modeling objective - be it simulate a game

or define what comes in the box. For example, there are multiple types of cards described in Figure 3: the student could evaluate whether there is any value to creating a generalization with specializations for each type, and assess which stakeholders would use that information and for what purpose. Or, the student could discuss any potential impacts the decision to create the generalized card would impact the model organization, or the ability to query the model.

In the Monopoly model, the primary packages that contain the structure of the game are those titled "Players and Pieces" and "Monopoly Board." The former package contains basic definitions of the different types of players that one can be in Monopoly. For instance, there are various tokens that a player can choose to represent him or herself in the game. To define these tokens, it is best to use a block definition diagram (BDD) like the one shown in Figure 4. Note that each of the tokens are shown as specializations of the generic Token block, which enables the defined part and value properties to be inherited by the tokens. These attributes would be necessary for potential

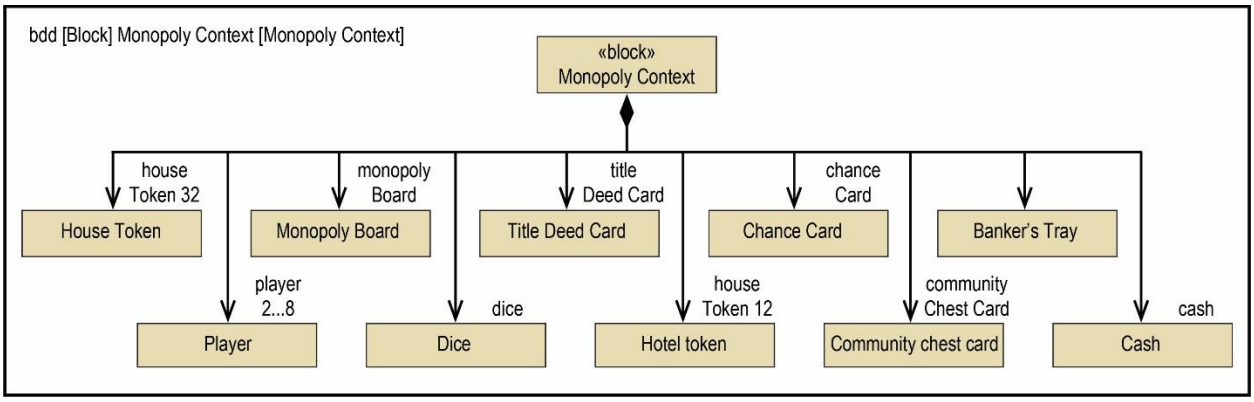

Figure 3. Sample block definition diagram of Monopoly Context

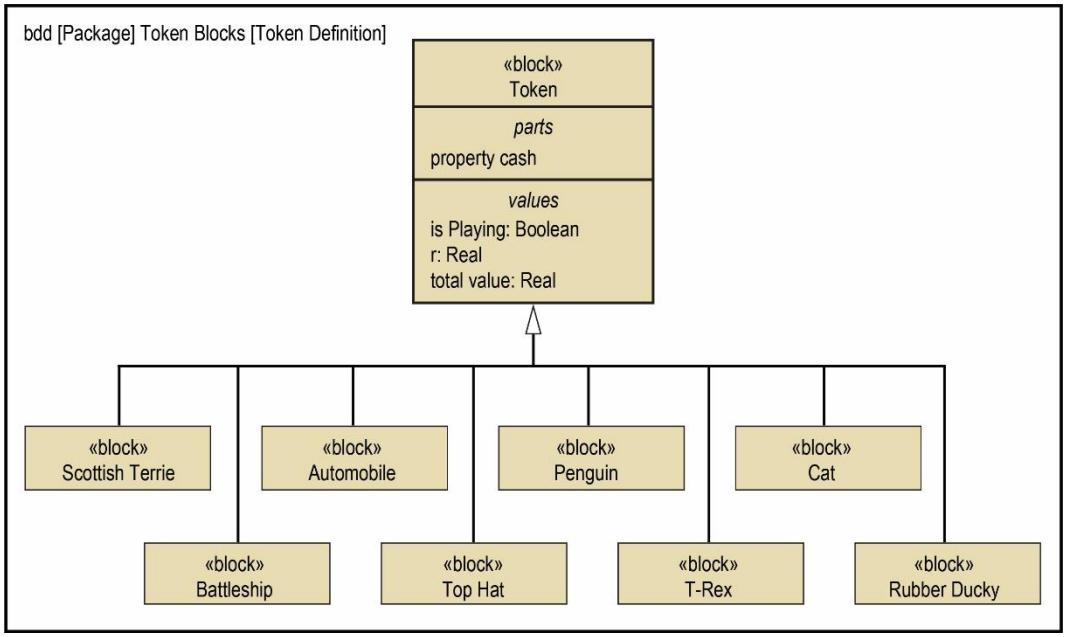

Figure 4. Token definition using a Block Definition diagram

future work to run a simulation of a game, or calculate the total value of money in circulation, for example. If stakeholders were not interested in simulating a game, the learner would consider whether those attributes are necessary, and whether the effort taken to detail each type of token is appropriate.

The latter package, titled "Monopoly Board," defines the structure of the physical board itself and the individual spaces. The associated BDD contains each of the 40 spaces, or properties, on an actual Monopoly board. Each space is then related to the Monopoly Board block via the part property relationship. This relationship, displayed in a BDD as a line with a black diamond, represents the idea that each of the spaces is a part of the board, and similarly, the board is the owning element of each of the blocks. This diagram, shown on Figure 5, is one example of an opportunity to work with part properties. The detail-oriented learner could go so far as to enumerate each Community Chest or Chance Card, for example.

The Monopoly model can be utilized to explore more advanced modeling concepts, such as the development of custom stereotypes. A <<monopolyBoardSpace>> stereotype is applied to each of the real estate property space elements. This stereotype allows each of the spaces to inherit a certain list of attributes typical of all Monopoly board spaces. The custom attributes of the <<monopolyBoardSpace>> stereotype are price, color, and type, as shown in Figure 6. The color and type attributes are also typed by custom value types, "Color" and "Space Type," respectively. Further stereotypes could be defined to group spaces by their space type, rather than simply utilizing a value type, which could enable more detailed implementation of cash flows to support a hypothetical simulation of a game, for example. The learner could assess either option in consideration of their overall modeling goals and expected model output, just as they would in a real project scenario.

Activities

The high-level package, titled "Game Play," contains the behaviors utilized in playing the game, represented as activities, and activity diagrams that convey the order in which the behaviors occur as control flows. The primary activity that was modeled is called "Play Game," seen in Figure 7. This activity details the cyclical nature of playing a game of Monopoly at a top level. Figure 7 takes a player through all of the possible paths of one "turn." A higher-level activity flow could be utilized to set this behavior in a loop until some conditions are met, for example. The "Play Game" activity makes use of several aspects of modeling activities, including merges, forks, decision nodes, and event probabilities (in concept). Future modeling activities could integrate the system model with an external tabulation of various event probabilities given a player's location on the Monopoly Board, further developing the learner's skills in cultivating an analysis ecosystem. While this diagram only conveys control flows, a learner could also consider the flow of cash or pieces as represented by object flows to fully develop experience in applying activity modeling techniques. Cross-cutting methods could also be explored via the use of swimlanes to represent multiple players or the role of the bank, for example.

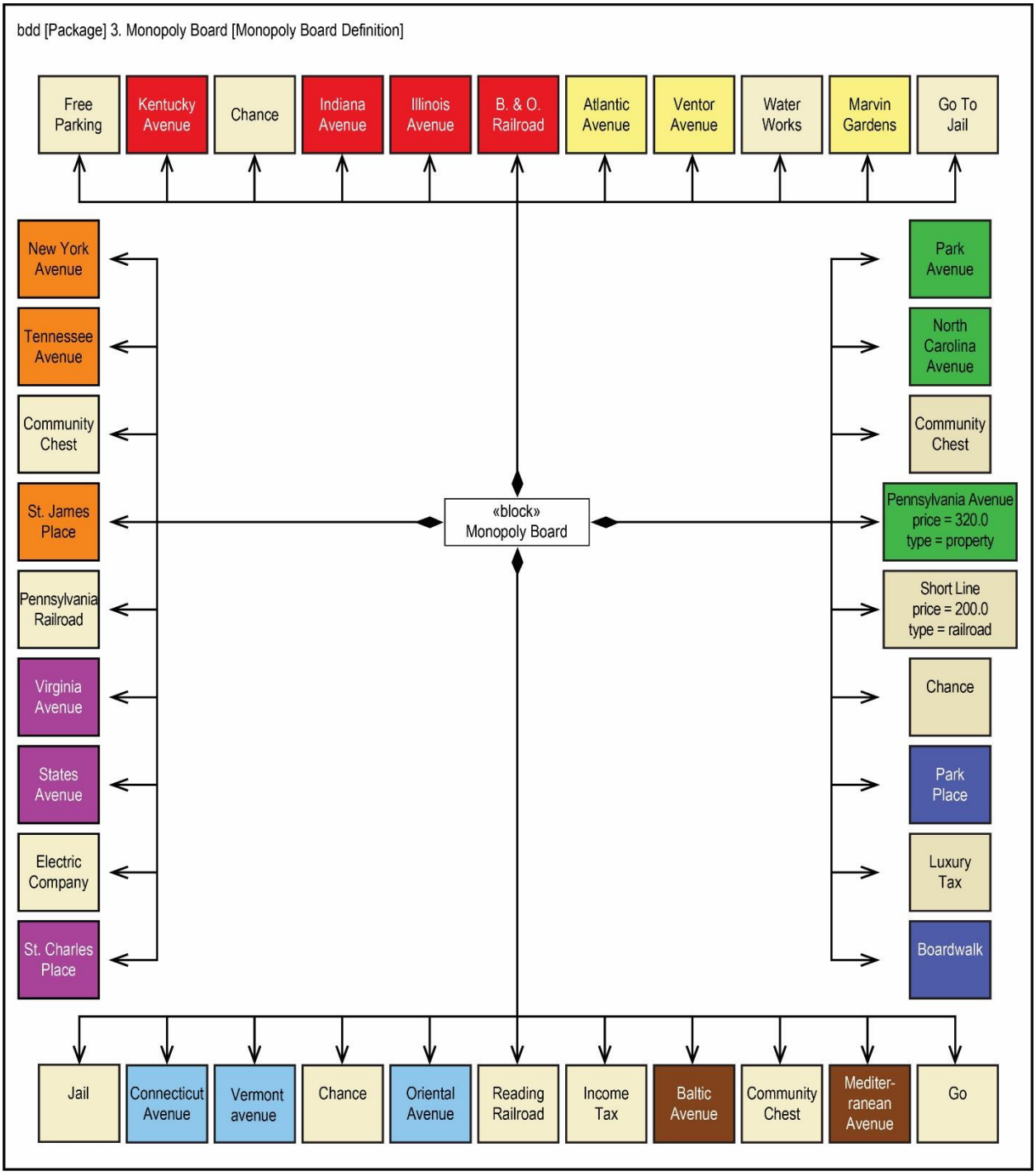

Figure 5. BDD describing the Monopoly Board and each space of real estate

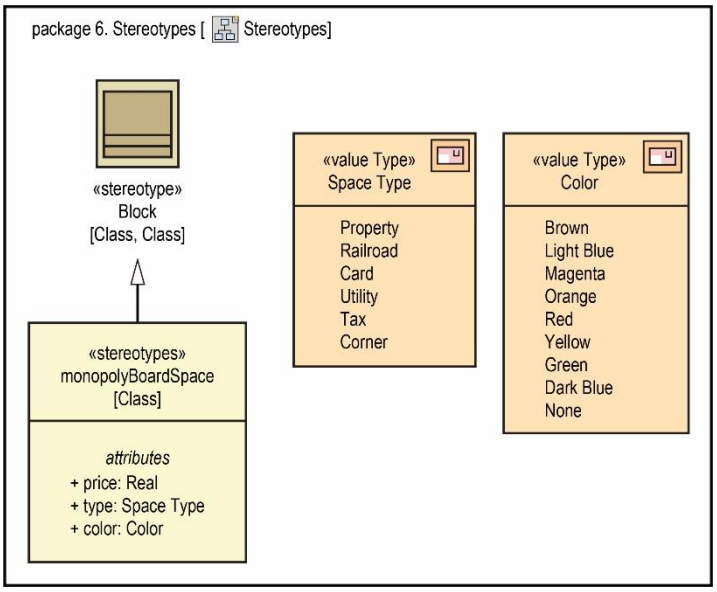

Figure 6. Definition of the <<monopolyBoardSpace>> stereotype

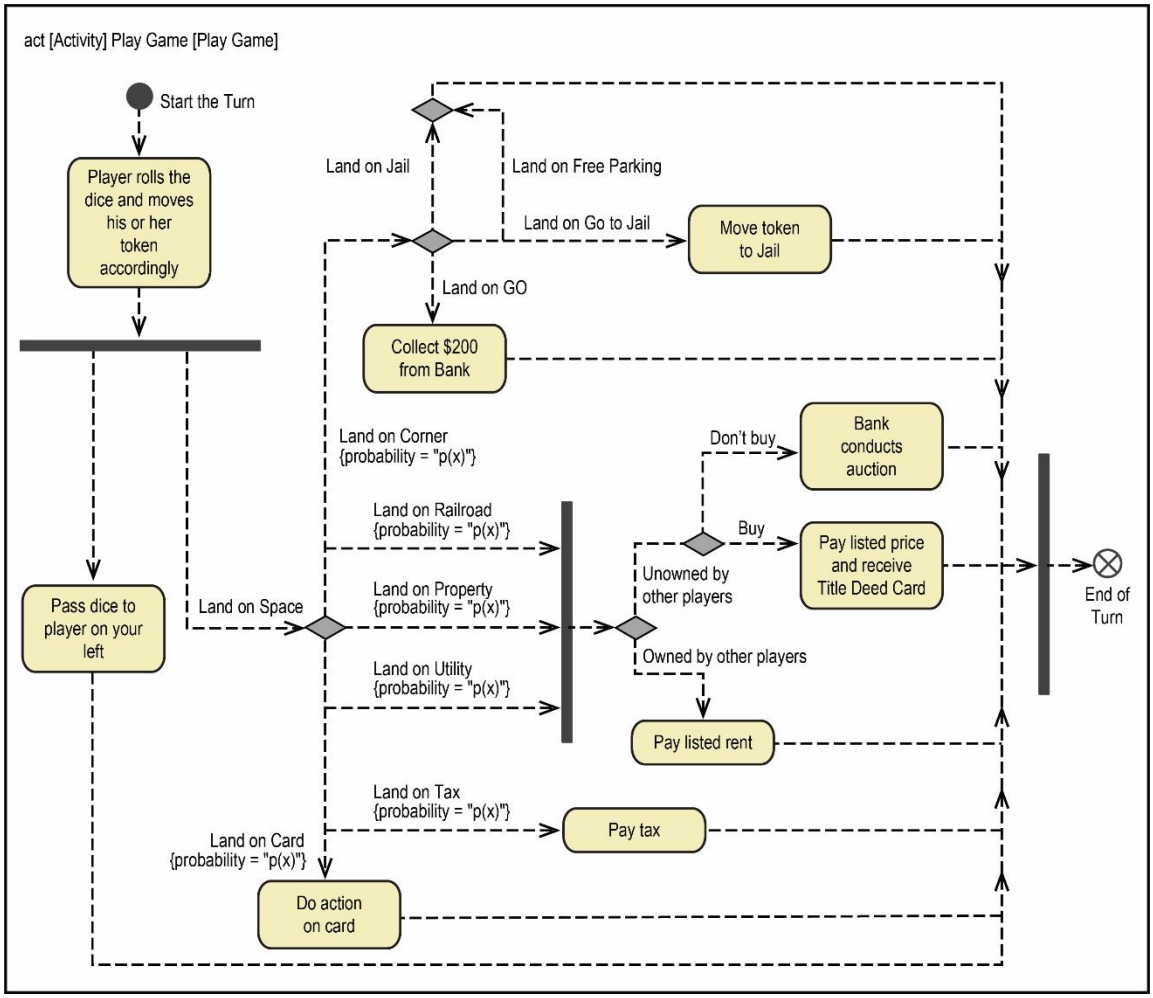

Figure 7. Activity diagram of "Play Game"

Parametrics

The most appropriate aspect of the game of Monopoly that one can analyze parametrically is the system of money. In the Monopoly model, this analysis is done in the package titled "Money." In Monopoly, numerous exchanges of money take place between the bank and the players and between the players and each other. One could envision using the model to monitor and track the money moving through this artificial economy. For instance, the Monopoly model contains blocks that define the "tokens" that a player can use in a game. Therefore, there is a model element that represents any player partaking in a game at a given time. This allows one to track the amount of money or other specific assets that a player has obtained. To accomplish this, one can assign a value property to these blocks that represents the amount of value (in money and assets) associated with one player. Ideally, this value can be automatically updated based on other activities that occur within model. For example, it is possible to track a player's actions during his turn and add or subtract money or property to or from his token based on the path that he takes. At the end of the game, it would be simple to determine who has the highest net worth (just look at the player's block).

Therefore, throughout the course of a game of Monopoly, players may easily calculate their net worth in order to establish who is currently in the lead. This act of summation can readily be represented in the model via the application of parametric "roll-ups." In order to do a roll-up in SysML, there first needs to be some property of some tiered system to aggregate. Figure 8 describes the approach taken to aggregate a player's net worth, which consists of cash on-hand,

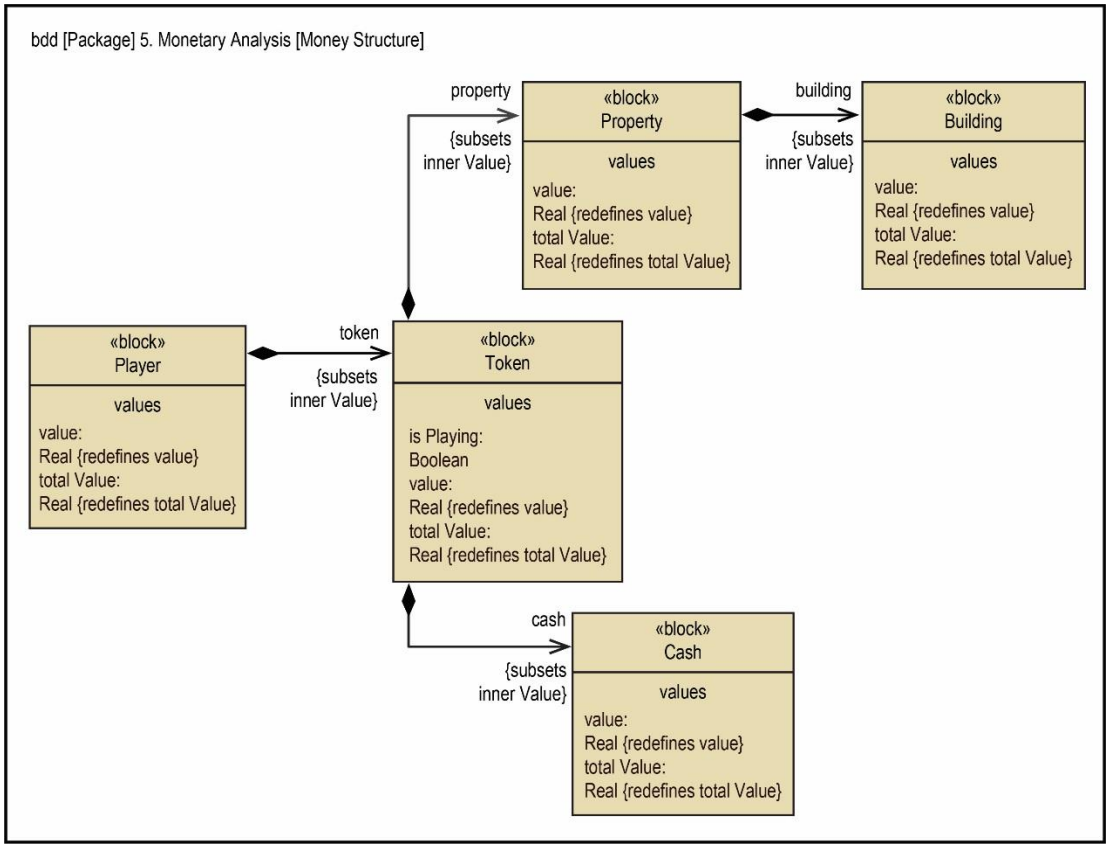

Figure 8. Definition of the tiers of value for a given player or token

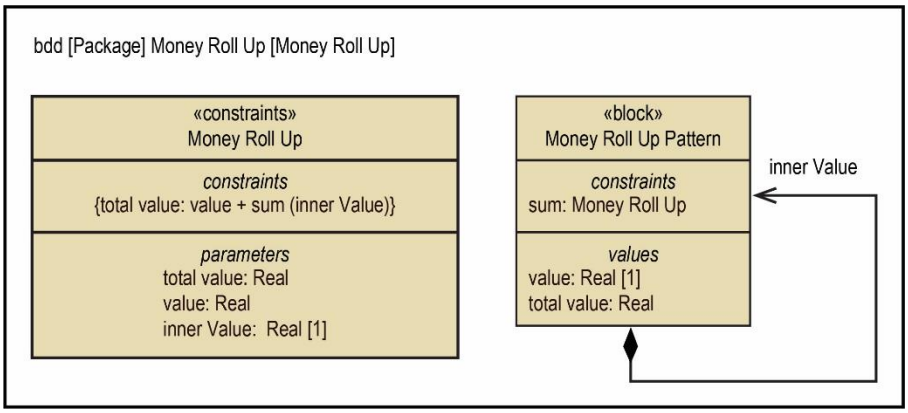

Figure 9. Summation constraint block and recursive 'Roll-Up' pattern

property value, and the value of buildings located on that property. However, the value of some of those parts is the sum of its own subparts, described in Figure 9 via the recursive "innerValue" part property and "Money RollUp" constraint property. Using this pattern, given the worth of each player's cash, property, and buildings, it is possible to derive the total "net-worth" for the player.

The purpose of this section on parametric modeling is not to explain the concepts and algorithms associated with a roll-up. Instead, it is to illustrate that the same tools used to model complex systems can be used to model a game as simple as Monopoly. It is also worth noting that a roll-up, a modeling pattern that can be applied to many different types of systems, necessitates the use of several other previously-discussed modeling techniques. As such, parametric modeling is an excellent method to gain experience in several of the pillars of SysML. Additionally, since most games feature some sort of scoring system, it is almost always possible to build parametric models of games, further highlighting their worth as MBSE learning tools.

This metric is useful not only for practice in creating roll-ups, but also because a player's total value has other important implications specific to the Monopoly model. For instance, earlier discussion mentioned that certain requirements (or, in the case of the Monopoly model, rules) are "satisfied" by elements or metrics external to the requirement. A parametric construct similar to the example described above, with aggregation of all the money across all players and the bank, could be used to satisfy the requirement that the quantity of money in circulation at a given point must remain constant. The roll-up pattern is one example of several parametric approaches that adds to the complexity of the model and enables more interesting aspects of the game to be conceptualized and modeled in SysML.

Discussion of Further Model Content

While the samples above did not capture use cases, interface definition, state machines, or sequences, each of these constructs could be utilized to more fully describe the game.

- Use Cases: Monopoly requires a player also take on the role of Banker, whose use cases might include such functions as distributing money and real estate or levying fees. Regular players key functions include making money, paying fees, and building real estate.
- Interfaces: Interfaces exist between players and the bank (as money and real estate are passed back and forth), between players, and between board spaces, which constrain the possible movement of players.

- State Machines: Monopoly has few technical game states, such as constraints on players when they are in jail, but should the modeler consider the human element of the game, additional states could be captured to represent a player in "spite mode," who will not accept any monetary offer for a property simply to prevent another player from owning it.
- Sequences: Over the course of gameplay, players exchange requests for money, submit bids, and exchange responses.

Limitations of the Approach

While modeling games may improve a learner's ability to accurately utilize SysML and gain insights into various modeling approaches, the premise does not provide full coverage of all practical concerns in transitioning to a model-based approach to systems engineering. The basic model of a game will not, without additional direction to the learner, provide an opportunity for the learner to consider the implications of the iterative nature of system design, nor the motivation to incorporate additional metadata or attributes to support the systems development lifecycle or configuration management. Similarly, given there is no motivation other than accurately describing the execution of the game, learners would not develop skills in strategically defining and prioritizing the model's scope, depth, or organization.

Both of these drawbacks could be mitigated by providing teams with different modeling "purpose prompts," and directing the learners to prioritize modeled content, format, and depth in consideration of their prompt. Providing different learners or teams with different prompts, but using the same subject matter (the selected game), could further enrich the discussion of the art of modeling, as learners compare the impacts of various modeling decisions made in light of differing priorities. Some rudimentary configuration management techniques could be introduced if the selected game subject matter had a legacy feature, where each play permanently changes the board and/or rules of the game: the modeling prompt could direct modelers to record the board or rules update as a change request and ensure appropriate metadata is captured in the process.

Unless the learner works for a game development company, this approach has an additional challenge in that the subject matter itself is not immediately relevant to the learner in their workplace. A formal study would be necessary to assess whether this proposed learning approach results in significant improvements in modeling capabilities that might outweigh the delay in direct application of the skills to the project.

Feasibility Assessment Summary

The authors organized a short feasibility assessment with six members of the National Aeronautics and Space Administration Glenn Research Center's (NASA GRC) MBSE Working Group to assess the hypothesis for further study. Before the modeling exercise, the participants were asked to: 1) rate their modeling expertise (on a scale from "I've never modeled before" to "expert"); 2) list how many SysML courses they had taken previously (on a scale from 0 to 3 or more); and 3) describe their confidence in their ability to model activities with different levels of abstraction or nesting (on a scale from "not confident at all" to "extremely confident"). The six participants were given the following model prompt: "Create a behavioral model of the game Monopoly using activity diagrams and associated elements. Generate model content that includes at least two levels of activity abstraction or nesting (decomposition)." The participants were divided into pairs and

given the option of working in a modeling tool or simply drawing their activity diagrams on paper. After 15 minutes of development time, the participants submitted their drawings or sample model to the assessment organizer to review as a group.

The assessment organizer asked each group to review their model with the group and discuss why they took their selected modeling approach. Following the brief model overview, the organizer invited the other participants to ask questions or provide feedback to the presenting team. Each of the three models covered the steps to set up and play the game, but each team added details in different areas: the first group detailed the game set-up processes; the second group elaborated the steps that define a player's turn; and the third group took a more algorithmic approach that included counters for dice values, board position, and number of turns. This model variety facilitated a rich discussion between the participants, which included questions about the SysML notation, correct application of the language, and use of the modeling tool. At the conclusion of the discussion, the organizer asked the participants to again assess their confidence in modeling activities with different levels of abstraction, and evaluate how confident they were, with more time to do exercises similar to this, that their understanding of SysML would increase.

At the end of the feasibility assessment, the organizer shared the premise of this paper with the participants, and invited discussion about the value of the exercise they had just completed with this context. The participants noted that the majority of their discussion had focused on comparing modeling approaches and model purpose, and there was very little discussion of the subject matter, Monopoly. The organizer asked the participants to describe what they did or did not like about the approach, and gave an opportunity to express any additional comments, questions, or concerns. Finally, the organizer asked participants how likely they would be to recommend this hypothesis for further study (on a scale from "1- not at all" to "5-very likely").

Feasibility Assessment Results

Of the six participants, one had never modeled before, three rated themselves as novice practitioners, and two rated themselves as seasoned practitioners. The participants had taken an average of 2.3 SysML courses previously. Five of the six participants reported an increase in confidence in their ability to model activities with different levels of abstraction after conducting the exercise, while the sixth participant (self-designated a seasoned practitioner) reported they remained extremely confident both before and after the exercise. When asked to assess their confidence that their understanding of SysML would increase after further exercises similar to this feasibility assessment, two participants indicated they were "somewhat confident" (the middle rating), three said they were "quite confident" (the fourth-highest rating), and one said he was "extremely confident" (the fifth and highest rating). On average, the participants rated their likelihood to recommend the hypothesis for further study a 4.25, on a scale from 1 to 5.

Participants cited the ease of understanding the subject matter and its potential to be described in varying levels of detail as aspects of the approach they liked. Most comments from participants regarding what they did not like about the approach were related to the very short nature of the exercise and the fact that it did not include any instruction prior to the group exercise. One participant noted the difference between modeling a subject one understands very well, compared to the real world, where projects are working to define a system that has never been created before. The individual commented: "Modeling a known system can help you learn a tool and SysML, but

it won't really help you learn how to engineer a new system while capturing it in a model (which is difficult to teach)."

While this feasibility assessment was not conducted in a rigorous scientific manner, the authors believe the effort did establish that the local MBSE community at NASA GRC is supportive of further research and application of this approach.

Forward Work and Future Applications

In each of the existing Monopoly samples, additional detail could be added to further develop more advanced modeling skills. One could capture more attributes of the game or enhance traceability between elements. This increase in detail would aid modelers in developing skills in properly abstracting concepts and maintaining consistency and traceability between levels of abstraction. Additional value properties and constraints could be added to enable game simulation, and thus develop skills in setting up more complex analyses, or interfacing the system model with other mathematical modeling tools. Report templates or other communication tools could be written to enable the outcome of such simulations to be generated from the model and transformed into a more traditional document summary.

INCOSE could host tool vendor, chapter, and/or university demonstrations or competitions, where teams would showcase their approach to modeling a game and discuss challenges and benefits of specific modeling approaches or tool capabilities. Should multiple teams address the same game, alternative modeling approaches would more readily be assessed and the lessons learned applied to real world modeling problems.

Organizations could give different teams different objectives for building the model, and compare differences in modeling approaches to help learners appreciate the 'art' of modeling. For example, a model that is built with the intent to simulate game play and the accumulation of wealth may devote more precision and effort to the behavioral language elements, such as state machines or activities, than a model built to describe the rules of the game.

Lastly, game designers could consider developing models of their games tied to relevant simulations, to test their selected arrangement of game mechanisms. Variants of the mechanisms and attributes could be simulated to more quickly arrive at the most engaging balance of complexity, with reduced reliance on human game testers and their subjective feedback.

For the game geeks among systems engineers, perhaps models may be a more instructive way to learn both the rules of the game and winning strategies than traditional paper instructions (documents!).

Acknowledgements

Monopoly subject matter used with permission. Hasbro does not sponsor or endorse the contents of this paper. The authors would like to recognize the NASA GRC MBSE Working Group for their technical review of this paper and willingness to participate in a feasibility assessment of this modeling approach. The authors also thank Kyle Wolff for applying his love for games to the review of this paper.

References

Andersson, H., Herzog, E., Johansson, G. & Johansson, O., 2010, 'Experience from introducing Unified Modeling Language/Systems Modeling Language at Saab Aerosystems', *Systems Engineering* 13, 369–380.

Delligatti, L., 2014, *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Pearson Education, Crawfordsville.

Ferreira, P., 2001, 'Tracing Complexity Theory', ESD.83 Research Seminar in Engineering Systems, Massachusetts Institute of Technology.

Friedenthal, S., Moore, A., Steiner, R., 2015, *A Practical Guide to SysML, Third Edition: The Systems Modeling Language*, Elsevier, Waltham.

Geekdo n.d., *Board Game Geek*, viewed 5 November 2017, from https://boardgamegeek.com/wiki/page/mechanism/.

Hallqvist, J. & Larsson, J., 2016, 'Introducing MBSE By using Systems Engineering Principles', *INCOSE International Symposium* 26, 512–525.

Jingles, R., 2013, 'What Lessons Can Your Kids Learn from Playing Monopoly? My Chat with Philip E. Orbanes, Author of *Monopoly, Money, and You*', in *Reaching for Greatness*, viewed 3 November 2017, from http://www.reachingforgreatnessguide.com/blog/.

Kismet, M., 2017, 'The Top Ten Board Games of all Time', in *Hobby Lark*, viewed 3 November 2017, from https://hobbylark.com/board-games/.

Vipavetz, K., Murphy, D. & Infeld, S., 2012, 'Model-Based Systems Engineering Pilot Program at NASA Langley', *AIAA SPACE 2012 Conference & Exposition*.

Voirin, J.-L., Bonnet, S., Normand, V. & Exertier, D., 2015, 'From initial investigations up to large-scale rollout of an MBSE method and its supporting workbench: the Thales experience', *INCOSE International Symposium* 25, 325–340.

Biography

Matt Corrado is an undergraduate Aerospace Engineering student at Georgia Tech. He has been engaged with MBSE since he started doing research in Georgia Tech's Aerospace Systems Design Laboratory, and he has since moved on to apply his MBSE skills to projects at several different organizations and laboratories. As an intern at NASA Glenn Research Center in the Summer of 2017, Matt helped to build models and simulations for the Solar Electric Propulsion project, and he has since been contributing to electric propulsion experiments in Georgia Tech's High-Power Electric Propulsion Laboratory. Matt plans to graduate with a Bachelor's Degree in the Fall of 2019 and pursue graduate studies the following Spring.

Kathryn Trase received a Bachelor of Science in Industrial and Systems Engineering from The Ohio State University in 2011. She has been with the NASA Glenn Research Center for seven years. Kathryn has led MBSE implementation in the Radioisotope Power Systems Program, Asteroid Robotic Redirect Mission, and Solar Electric Propulsion testbed. She currently supports the Orion Program. She is a member and former chair of NASA Glenn's MBSE Working Group and 2017 Past-President of the Cleveland-Northeast Ohio Chapter of INCOSE.