Rotary Dial Model - A Model-Driven Methodology for Autonomic Network Design

Arun Prakash, Ina Schieferdecker, Michael Wagner and Christian Hein
Modeling and Testing for System and Service Solutions
Fraunhofer FOKUS Institute for Open Communication Systems
Berlin, Germany
{arun.prakash, ina.schieferdecker, michael.wagner, christian.hein}@fokus.fraunhofer.de

***Abstract*—Designing and developing complex multi-domain systems such as autonomic systems and networks is not only a creative but also a systems engineering challenge. Current systems engineering methodologies are unsuitable, as autonomics is a multi-domain field that requires the knowledge of several domains, such as control-theory, computer networks, software-engineering, etc., and comes with its own set of requirements and challenges. We showcase why and how current systems engineering approaches fail, and advocate for an *avant-garde* systems engineering approach to designing safety-critical and multi-domain systems, such as autonomic entities. In this paper, we introduce a new model-driven systems engineering methodology, called the *Rotary Dial Model (RDM)*, whose functioning is inspired by the standard telephone rotary dial. We describe in detail the working of the RDM model, and showcase its robustness and flexibility for the system design process.**

***Index Terms*—Model-Driven Methodology; Systems Engineering; Autonomics; RDM; MDA; MDE; Safely-critical systems**

I. Introduction

CONTROL theory concepts in general and hierarchical controllers in particular are extensively used in the field of chemical, automotive, aviation and in robotic/industrial manufacturing settings [1, pp. 15-19]. Methodologies to design mechanical controllers are well researched, mature and available in the literature [1], [2]. Design methodologies for software based mechanical controllers, e.g. Electronic Control Unit (ECU) of a car, that requires the embedded software to control mechanical components are also available [3].

A field where control theory concepts are being freshly applied is the domain of Computer Networks/Networking, ergo autonomics/self-managing networks. Even though autonomic/self-* concepts have been well studied and researched, with several autonomic networking architectures such as 4D [4], FOCALE [5], CONMan [6], GANA [7], ANEMA [8] proposed recently [9], the domain as such still remains in its infancy. In addition, the domain of autonomics comes with its own set of challenges and requirements. As described in [10] these include behavior conflicts, stability problems, response time issues, operating regions overlap etc. Its acceptance and application is largely hindered due to the lack of a standardized *design & development* methodology and associated tooling infrastructures that meet industry benchmarks. In this paper, a model-driven methodology to design and verify hierarchical controllers of an autonomic network is presented.

The rest of the paper is structured as follows: In Section II, a brief background on the existing methodologies and approaches for systems engineering and for designing autonomic systems is provided. In Section III, we showcase our newly proposed model-driven methodology fashioned for designing and developing autonomic systems/networks. Finally, we provide our conclusions and insights in Section IV.

II. Background

A methodology with formal methods to model, simulate, verify and validate autonomic entities is an indispensable prerequisite for the proper design, development and functioning of an autonomic network. In [11], [12], the authors have provided a methodology to design hierarchical controllers used for the management of network resources. Their methodology, as shown in Fig. 1 is founded on a model-based dependability evaluation scheme for systems and components structured in an hierarchical manner [11], [12].

The methodology is described as a two step process. *Step 1* concerns with a model design process and deals with *"how to model a complex system starting from its functional specification and applying a stepwise refinement to decompose it in small sub-models."* [12]. Thus, in this step, abstract models are first modeled from the *system description* and later refined into *detailed model* artifacts. In *Step 2*, a solution based on the *detailed model* constructed in *Step 1* is produced. In both steps, the methodology focuses on a stepwise refinement of the model and its corresponding model solution.

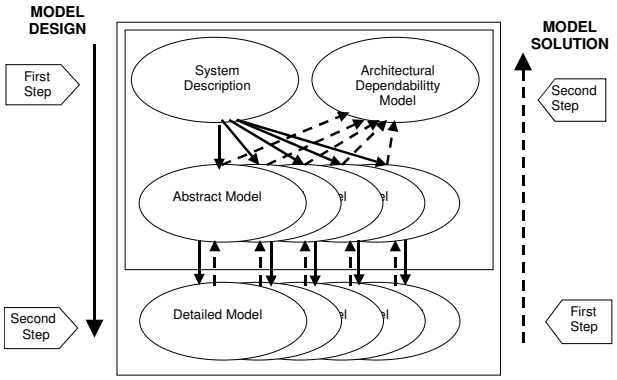


Fig. 1. Dependability Evaluation based Modeling Methodology for Designing Hierarchical Controllers for Network Resource Management[11], [12].

978-1-4673-4941-3/12/$31.00 ©2012 IEEE

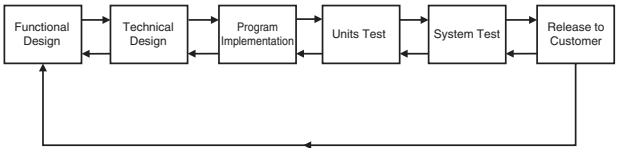


Fig. 2. Iterative Model for Systems Engineering [13]

Even though the methodology provides an efficient dependability evaluation mechanism, in the context of autonomics, the methodology described as such cannot be applied. Crucial requisites such as *Stability Analysis*, *Traceability*, *Variability Management*, *Behavior Simulation*, etc, that are essential for the completeness of an autonomic network model are missing. Providing a quantitative dependability evaluation for the systems involved, though important, is insufficient for developing industrial grade products.

Furthermore, from the experience in designing autonomic networks in the context of the EFIPSANS project [14], existing systems engineering methodologies such as the *Waterfall Model* [13], *Iterative Model* [13] or the *V-Model* [15] to name a few are unsuitable. The reasons for this is manifold. Current systems engineering methodologies have been conceived to be strictly sequential, with some iteration and feedback provided. Despite their robustness and perceived adaptability, in reality they allow for very little flexibility in their usage. For instance, a systems designer applying the *Iterative Model* [13], shown in Fig. 2 would be prohibited from jumping back to *Design* from *Testing*. They are based on the assumption that the output of one method is complete and perfect. Regrettably, in the real world this assumption does not hold. Furthermore, even though some feedback is now available, e.g. the *V-Model* [15] or *Incremental Model* [13] or in the improved *Waterfall Model*, the systems designer is still forced to complete the intermediate steps before such a feedback is available. Finally, the concept of a meta-model that constraints the designed system to its domain is unavailable in all the above mentioned methodologies.

Systems design is a creative process that is prone to requirement changes and new design ideas. Forcing a systems designer to a sequential design process is not only restrictive, but also wasteful, complex and inflexible. Furthermore, in the context of safety-critical systems *design consistency* is paramount. In the context of autonomics, a multi-domain field, the system engineering process of designing autonomic entities requires expertise and application of different fields such as software engineering, protocol engineering and modern control theory. Thus a model-driven methodology that is *simple*, *flexible* and provides *design consistency* is required, especially in the context of autonomic systems and networks, and is the novelty of our approach. The requirements for such a methodology has been studied and published in [17], [18].

III. Rotary Dial Model (RDM)

The *Rotary Dial Model (RDM)* is inspired from the standard rotary dial mechanism (Fig. 3) found in telephones manufactured between the 1920's to the 1980's. A rotary dial consists of a ring with holes, with each finger hole representing a number shown under it. The ring is mounted on a shaft that extends inside to the telephone through the dial plate. The dial plate is the one that contains the printed numbers. The ring rotates around the shaft, while the dial plate remains stationary. In order to dial a telephone number, which is a sequence of numbers, the user dials each number by putting the finger into the corresponding *finger hole* and turning the dial clockwise until the finger hits the *finger stop*. When the finger is released, the spring mechanism of the shaft brings the dial back to its default position. It is this simple arrangement, intuitive nature and a hassle free dialing mechanism that has provided the foundation for the conception of the *Rotary Dial Model* (RDM).

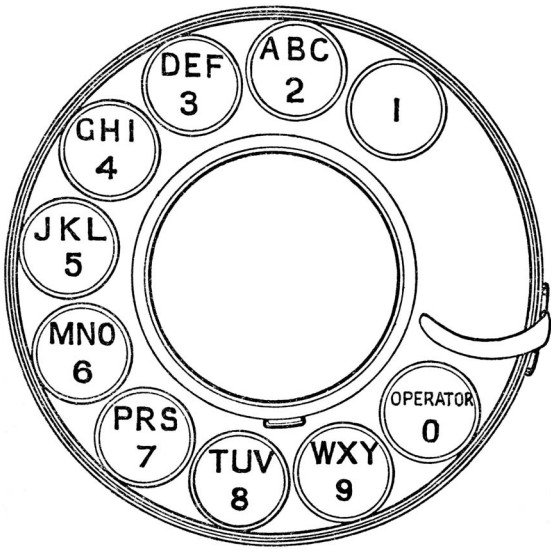


Fig. 3. Standard Telephone Dial (1920's - 1980's) [16]

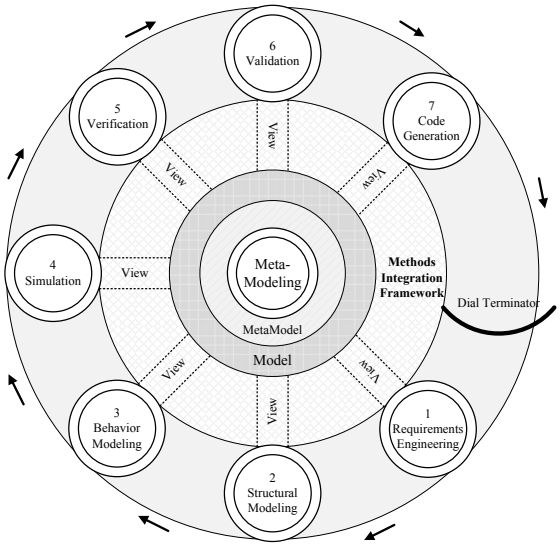


Fig. 4. Rotary Dial Model (RDM)

The *Rotary Dial Model* (RDM) is a model-driven systems engineering methodology that is being proposed here for the purpose of designing and verifying autonomic systems. The conceptualized methodology in its holistic form is shown in Fig. 4. The methodology is structured in such a way that it provides the same look and feel of a standard telephone rotary dial. However, the idea here is not to mimic the rotary dial in its entirety, but to profit from its visceral nature and to apply certain mechanical concepts, to the domain of systems engineering, specifically in the context of autonomics. For instance the concept of finger holes and the process of dialing a number in the standard rotary dial are equated with the positioning of formal methods and the execution of a method, respectively. Thus, the RDM methodology proposed here can be considered an *avant-garde* approach to systems engineering, and breaks away from the other well established methodologies for systems and software engineering [13].

The central component of the methodology is the *Methods Integration Framework (MIF)*. It is akin to the shaft of the telephone rotary dial. The structure and functionality of the MIF is conceptualized to ensure seamless integration, orchestration and sharing of the modeling and data assets among the various formal methods connected with it. A formal method is considered to be connected to the MIF through what is conceived as a *View*. All the modeling artifacts and data assets that are created and shared among the formal methods, are visualized to be disseminated through the integrations framework.

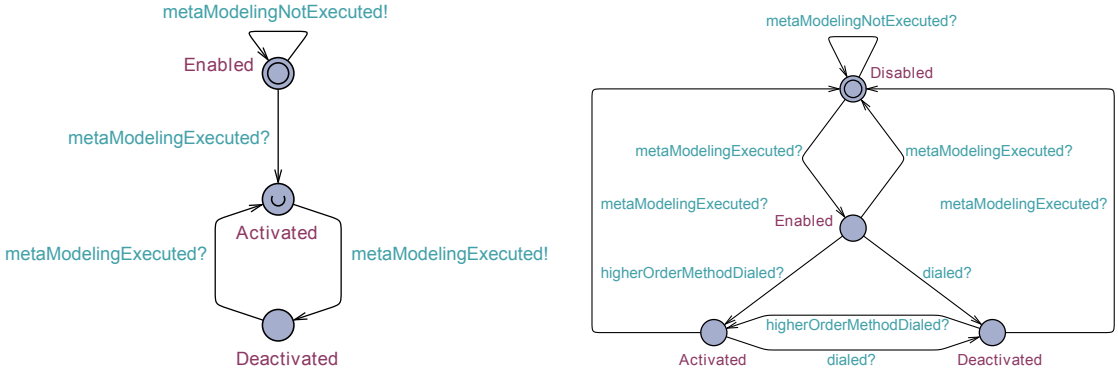
Views are analogous to the spokes of a wheel, in the sense that they connect the methods to the *shaft*, ergo the *Methods Integration Framework (MIF)*. A *View* is conceptualized to equip a method to hook itself to the MIF and render it with all the data resources, modeling artifacts and services available in the integration framework. Thus, a *View* is conceived to provide its associated method with a method-specific view of the modeling artifacts available in the integration framework. In other words, each method would view the same model artifact located in the MIF with a different perspective. Additionally, through the *View* a method can broadcast itself as a service that other interested methods can subscribe to.

In the *Rotary Dial Model (RDM)* eight different formal methods have been identified. Some of the formal methods, such as *Requirements Engineering*, *Verification* and *Validation*, are already in use in other well known methodologies, such as the *Waterfall Model*[13], *Spiral Model* [13] or the *V-Model* [15]. They have been adopted in RDM as such, without any modifications. Others such as *Structural Modeling* and *Behavioral Modeling* have been conceived by splitting up the existing formal method, *Modeling*. Thus for the purpose of designing and verifying hierarchical controllers of an autonomic network, the formal methods conceptualized in RDM are as follows :

- Meta-Modeling
- Requirements Engineering
- Structural Modeling
- Behavioral Modeling
- Simulation
- Verification
- Validation
- Code-Generation

In RDM, all the methods that require orchestration are numbered and placed along the dial in a specific order. The location of the numbered methods is akin to the *finger holes* of the rotary dial. The ordering and placement of the methods is of utmost significance and is explained below.

In RDM, there are seven numbered formal methods, numbered from 1 to 7, placed along the *finger holes* of the dial, and one unnumbered method placed at the center, on top of the *Methods Integration Framework*. The unnumbered formal method, namely the *Meta-Modeling* method is considered *prime*. In the context of this research work, *Meta-Modeling* is defined as a process of creating a *meta-model* that captures the concepts of the autonomic network architecture in meta terms. Additional numbered methods can be added to the *Rotary Dial Model*, if required. However, the unnumbered method position is reserved for *Meta-Modeling* and cannot be replaced by any other method, as all the other methods base their actions on the *meta-model* artifact. Thus, in RDM, only one common *conceptual* meta-model is designed and all the other formal methods participate in the construction of a single model that conforms to this conceptual meta-model.



(a) Dialing rules of the Unnumbered Method (Meta-Modeling) (b) Dialing rules of a single numbered method located on the dial

Fig. 5. State machine representation of the dialing rules in RDM.

As explained previously, all the methods that require orchestration are numbered and placed along the dial in a specific order. The ordering and placement of the methods is crucial, as it determines the correct sequencing and orchestration of the methodology. For this purpose, the concept of *order* is introduced.

The *order* of a method is inversely proportional to the distance between the method and the *dial terminator*, during the default stationary state of the dial. Thus the method farthest from the *dial terminator* is of the *highest order*, while the method placed closest to the *dial terminator* is of the *lowest order*. The *order* of the methods in between is determined by their relative positions on the dial. Thus, they are of a *higher order* in relation to some methods and of a *lower order* in relation to others.

In the RDM model shown in Fig. 4, the formal methods *Requirements Engineering* and *Code-Generation* are of the *highest order* and *lowest order* respectively. A formal method in between, say for instance the one placed at position 3, *Behavioral Modeling* is of a *higher order* in comparison to the methods placed at position 4, 5 and so on, and is of a *lower order* in relation to the methods placed at positions 1 and 2.

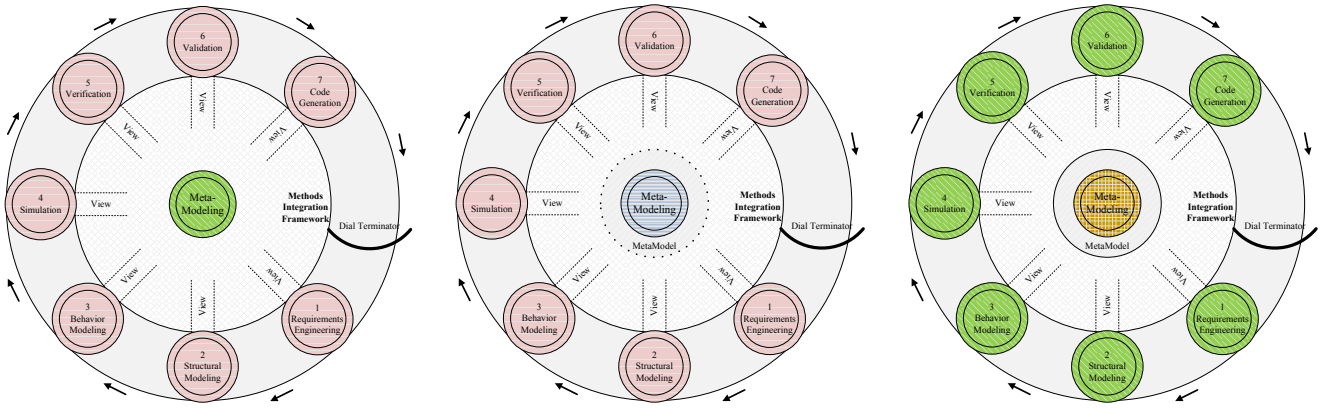
In *Rotary Dial Model (RDM)*, the concept of *executing* or *dialing* a formal method is visualized differently from the standard rotary dial. In RDM, the formal methods are associated with *states*, and thus the *execution* or *dialing* of a method is dependent on its current *state*. A method can be in one of the following *states*, described below, during the operation of the methodology.

- **Enabled** - A method in this *state* is available for use in the methodology, and can be *disabled*, *activated* or *deactivated*. Only when a state is enabled can it be *dialed* or *executed*.
- **Disabled** - A method in this *state* is unavailable for use in the methodology. A *disabled* state can only be *enabled*.
- **Activated** - A method becomes *activated* after a higher order formal method is *dialed*. An *activated* method can be either *deactivated* and/or *disabled*.
- **Deactivated** - A method becomes *deactivated* once it has been *executed* or *dialed*. A *deactivated* method can be *activated* or *disabled*.

The Finite State Machine (FSM) representation of the various *states* and their *transitions* is illustrated in Fig. 5. In Fig. 5(a), the *states* of the unnumbered method, i.e., *Meta-Modeling* are shown. As it can be seen, *Meta-Modeling* can never be *disabled*. This is because, the *Meta-Modeling* method is considered prime. In Fig. 5(b), the FSM for a method located on the dial is depicted. All the *state transitions* portrayed in the figure are a result of the modeler/user actions. Thus, unlike the standard rotary dial, in RDM, the *dialing* or *execution* of a formal method is governed by a set of rules. The rules are described as follows :

1) All methods with the exception of the *Meta-Modeling* are initially *disabled*.
2) Only one method can be *dialed* or *executed* at a time.
3) Only *enabled* formal methods can be *activated*, *deactivated* and *dialed* or *executed*.
4) *Dialing* is commenced when a chosen method is rotated in the clockwise direction.
5) *Dialing* is concluded when the chosen method hits the *dial terminator*.
6) When a formal method is *dialed*, all the methods that follow it are spontaneously *activated*.
7) Once a method is fully *dialed* it is *deactivated*.
8) All *activated* methods should be dialed at least once.
9) A method can be *redialed* any number of times.
10) The methodology is said to be complete when all the methods are in their deactivated state.

The RDM methodology is conceptualized to work as a two step process. *Step 1* deals with the conception of the *meta-model*. In *Step 2*, a *model* that is based on the meta-model (created in *Step 1*) is produced. The two steps of the RDM

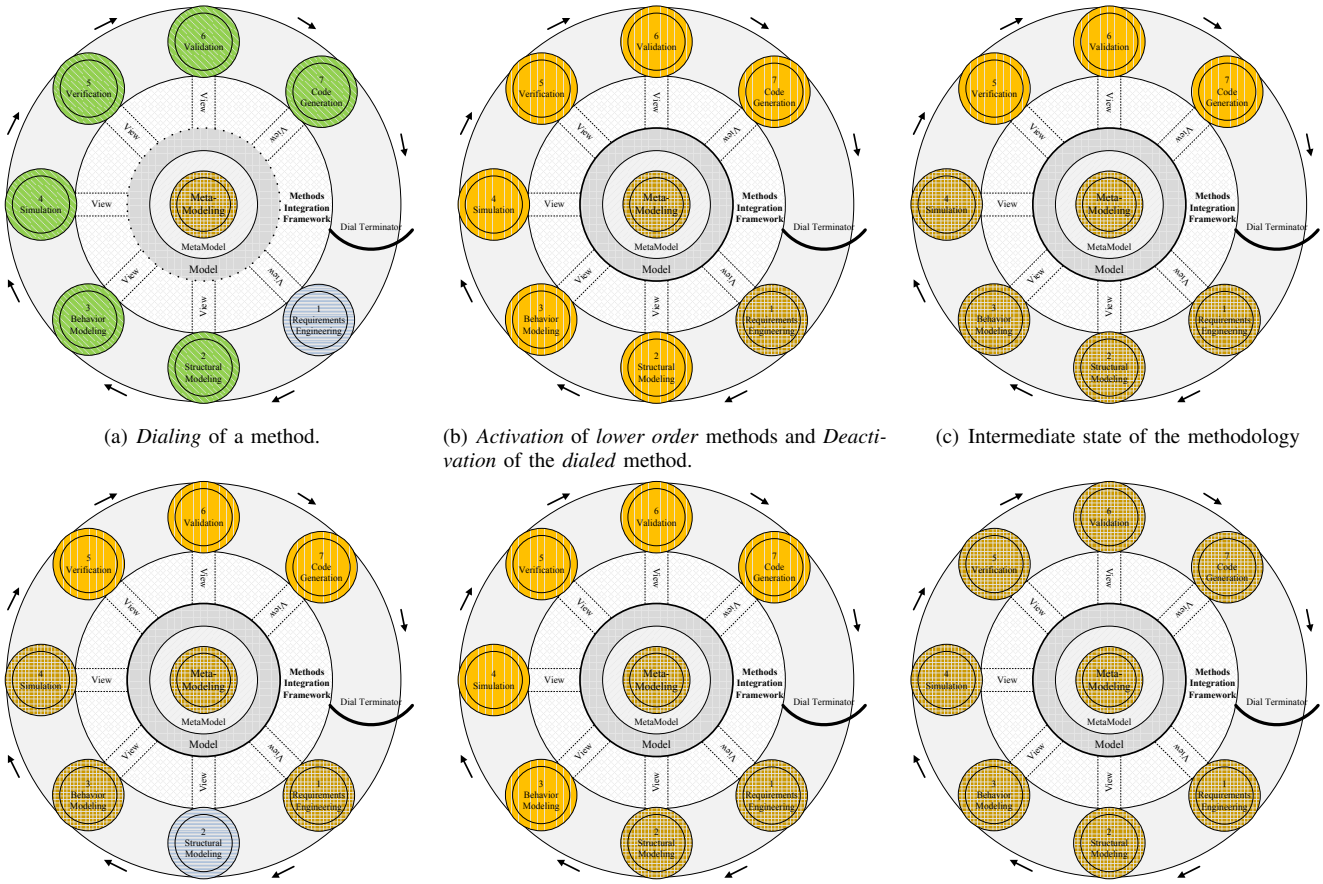


(a) Initial setup

(b) *Dialing* of the *Meta-Modeling* method

(c) *Deactivated Meta-Modeling* method and *Enabled dial methods*

Fig. 6. Step 1 of the RDM methodology



(a) *Dialing* of a method.

(b) *Activation* of *lower order* methods and *Deactivation* of the *dialed* method.

(c) Intermediate state of the methodology

(d) re*Dialing* of a method.

(e) re*Activation* of previously *Deactivated* methods.

(f) Completion of the methodology. All methods are in the *Deactivated* state.

Fig. 7. Step 2 of the RDM methodology

methodology are depicted through a series of illustrations in Fig. 6 and Fig. 7 respectively.

The methodology is initialized as shown in Fig. 6(a). All the methods with the exception of the *Meta-Modeling* method are in their *disabled* state. The methodology is commenced when the modeler executes the *Meta-Modeling* method, and produces a single meta-modeling artifact. Thus, at the end of *Step 1*, shown in Fig. 6(c), the *Meta-Modeling* method is *deactivated*, and all the other methods positioned on the rotary dial, are *enabled*. At any point of time during the execution of the methodology, the modeler can jump from *Step 2* to *Step 1*, but not vice versa. This property of the RDM methodology is a direct result of the *dialing* and *state transition* rules. This property is also visible in Fig. 5.

In *Step 2*, the modeler starts constructing the model by *dialing* the formal methods located on the rotary dial. Similar to the standard telephone dial, in RDM, the modeler is free to dial any method. However, if the intention of the modeler is to design and verify an autonomic system from scratch, it is *sensible* to start *Step 2* by *dialing* the formal method positioned at 1, namely the *Requirements Engineering* method, as shown in Fig. 7(a). It is considered *sensible* due to the nature of the *dialing* rules, especially *rule 10*, and the impact of *rule 6* on the overall operation of the methodology. Additionally, the benefits of commencing a systems engineering process with the *Requirements Engineering* is well understood and is in line with the other well established methodologies [13].

Once the *Requirements Engineering* method is *dialed*, it is *deactivated* and all the other *lower order* methods that follow it are *activated* and ready to be dialed. This is illustrated in Fig. 7(b). The modeler may now continue methodology by *dialing* the methods in the hierarchy of their *order*, as shown in Fig. 7(c). However, at all times the modeler is free to *redial* a method that was previously *dialed*. Thus for instance, after *dialing* methods positioned at 3 and 4, if needed, the modeler can come back and *redial* method located at 2. This can happen in cases, when for instance, the structure of a particular autonomic component needs to be re-designed based on the results obtained upon the *dialing* of the *Behavior Modeling* or *Simulation* method. The idea of forcing a modeler to continue with the execution of the remaining methods, when a feedback from an intermediate method is available is not sensible, and thus done with in RDM.

However, when a method is *redialed*, all the *lower order* methods that follow it are activated once again, as depicted in Fig. 7(e). What this implies is that when the model is changed or updated by a *higher order* method, all the methods of a *lower order* must be *redialed*. It makes a perfect method-

ological sense to *redial* (re-execute) the *Behavioral Modeling*, *Simulation*, *Verification* and *Validation* methods when the structural components of the model have changed. However, since *lower order* methods operate on the output of the *higher order* methods, and are constrained by the scope of their model view, it is not required to *redial* a *higher order* method when a *lower order* method is *redialed*. For instance, for the purpose of code-generation or recompilation of the model code, the execution of the *Requirements Engineering* or *Simulation* methods is not required. While there is a correlation between a *higher order* method to a *lower order* method, the correlation, *lower order* method to a *higher order* method, is orthogonal and is in line with other systems engineering and software development methodologies.

Finally, at the end of the methodology, all the methods in the RDM are in their *deactivated* state, as depicted in Fig. 7(f) and the autonomic systems model is considered to be complete, verified and validated. A detailed discussion of the individual methods, namely the *Meta-Modeling*, *Structural Modeling* and *Behavioral Modeling* methods is available in [19]. Case studies validating the concept and the usefulness of the RDM are detailed in [10], [19].

IV. Conclusion

In this paper, we showcase a new model-driven methodology for systems engineering. In contrast to other methodologies [13], [15], [20] that are rigidly linear or iterative, the Rotary Dial Model (RDM) has been conceived to be *simple*, *design consistent*, *flexible* and *intuitive*. It is a broadly decentralized methodology where all the methods share a single model perspective, share data and can operate almost simultaneously and independently of each other. It is the simple rotary based construction and the concept of MIF that allows for this near simultaneous and independent model development, simulation, verification and validation of autonomic entities. While, the focus of this paper has been the domain of autonomic systems and networking, RDM can be applied to other multi-domain fields like embedded systems, mechatronics, etc., where it is beneficial to have a unified meta-model in the design process.

In conclusion, the RDM would provide system designers with the requisite feedback and insights throughout the design process, rendering the much needed flexibility and robustness in their design. It would enable designers to make both model and if necessary, meta-model refinements during the operation of the methodology. Finally, the methodology is aptly suited to be realized as a tool-chain, as clear boundaries between the various components are established. This allows individual tools realizing the respective methods to be integrated and orchestrated through a tool integration infrastructure.

Acknowledgment

This work has been partially carried out in the context of the VARIES project [21]. The authors would also like to thank their colleague Diana Vega of Fraunhofer FOKUS, Germany for her critical insights.

References

[1] I. J. Nagrath and M.Gopal, *Control Systems Engineering*, 4th ed., K. K. Gupta, Ed. New Age International Publishers, 2006, ISBN: 81-224-1775-2.

[2] K. Ogata, *Modern Control Engineering*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[3] V. Jaikamal, "Model Based ECU Development - An Integrated MiL, SiL, HiL Approach," Presentation, Oct. 2008, Automotive Testing Expo North America 2008 Open Technology Forum.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A Clean Slate 4D Approach to Network Control and Management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, 2005.

[5] J. C. Strassner, N. Agoulmine, and E. Lehtihet, "FOCALE A Novel Autonomic Networking Architecture," in *Latin American Autonomic Computing Symposium (LAACS)*, Campo Grande, MS, Brazil, 2006.

[6] H. Ballani and P. Francis, "CONMan: A Step Towards Network Manageability," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 205–216.

[7] R. Chaparadza, "Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments," *International Engineering Consortium (IEC), Annual Review of Communications*, vol. 61, 2008.

[8] H. Derbel, N. Agoulmine, and M. Salaün, "ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks," *Comput. Netw.*, vol. 53, no. 3, pp. 418–430, 2009.

[9] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A Survey of Autonomic Network Architectures and Evaluation Criteria," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1 –27, 2011.

[10] A. Prakash, R. Chaparadza, and A. Starschenko, "A Model-driven Approach to Design and Verify Autonomic Network Behaviors," in *GLOBECOM Workshops*, 2011, pp. 701–706.

[11] S. Porcarelli, F. D. Giandomenico, P. Lollini, and A. Bondavalli, "A Modular Approach for Model-based Dependability Evaluation of a Class of Systems," in *International Service Availability Symposium 2004 (ISAS 2004)*, ser. Lecture Notes in Computer Science, vol. 3335. Munich, Germany: Springer-Verlag, May 2005, pp. 160–174.

[12] P. Lollini, F. D. Giandomenico, and A. Bondavalli, "A Modeling Methodology for Hierarchical Control Systems and its Application," *Journal of the Brazilian Computer Society* , vol. 10, no. 0104-6500, pp. 57–69, June 2005.

[13] B. Jayaswal and P. Patton, *Design for Trustworthy Software: Tools, Techniques, And Methodology of Developing Robust Software*. Prentice Hall, 2007.

[14] EFIPSANS - Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services, "EC FP7-IP Project," www.efipsans.org, 2008-2011, INFSO-ICT-215549.

[15] *V-Modell 97 - Development Standard for IT Systems of the Federal Republic of Germany*, IABG Industrieanlagen-Betriebsgesellschaft mbH Std., 1997. [Online]. Available: http://v-modell.iabg.de/

[16] *Encyclopaedia Britannica*, 12th ed. New York: The Encyclopaedia Britannica Company, 1922, vol. 32.

[17] R. Mortier and E. Kiciman, "Autonomic Network Management: Some Pragmatic Considerations," in *INM '06: Proceedings of the 2006 SIGCOMM workshop on Internet network management*. New York, NY, USA: ACM, 2006, pp. 89–93.

[18] A. Prakash, R. Chaparadza, and Z. Theisz, "Requirements of a Model-Driven Methodology and Tool-Chain for the Design and Verification of Hierarchical Controllers of an Autonomic Network," in *CTRQ'10: Third International Conference on Communication Theory, Reliability, and Quality of Service*, Jun. 2010, pp. 208 –213.

[19] A. Prakash, Z. Theisz, and R. Chaparadza, "Formal Methods for Modeling, Refining and Verifying Autonomic Components of Computer Networks," *Transactions on Computational Science XV - Special Issue on Advances in Autonomic Computing: Formal Engineering Methods for Nature-Inspired Computing Systems*, vol. 7050, pp. 1–48, 2012.

[20] D. Green and A. DiCaterino, "A Survey of System Development Process Models," Center for Technology in Government, Tech. Rep., February 1998.

[21] VARIES, "VARiability In safety critical Embedded Systems," 2012-2015. [Online]. Available: http://www.varies.eu/