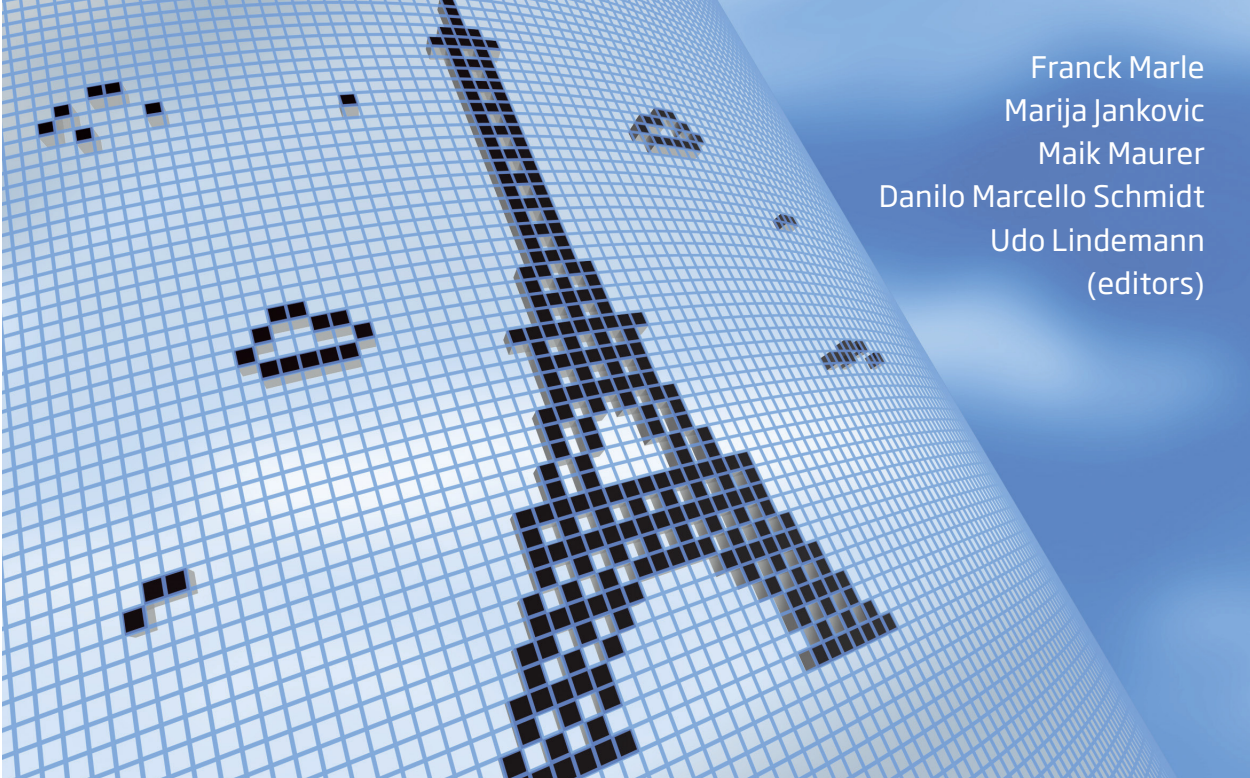




Franck Marle
Marija Jankovic
Maik Maurer
Danilo Marcello Schmidt
Udo Lindemann
(editors)

Risk and change management in complex systems

Proceedings of the 16th International DSM Conference
Paris, France, 2 - 4 July 2014



HANSER

Marle, Jankovic, Maurer, Schmidt, Lindemann
Proceedings of the 16th International DSM Conference
Paris, France, 2-4 July 2014

Note:

The CD-ROM for this book can be downloaded from
www.downloads.hanser.de by searching the word „Maurer“ or
<http://www.hanser.de/9781569904916>

Your password is: maurer491

Franck Marle
Marija Jankovic
Maik Maurer
Danilo Marcello Schmidt
Udo Lindemann
(editors)

Risk and change management in complex systems

Proceedings of the 16th International DSM Conference
Paris, France, 2 – 4 July 2014

HANSER

The Editors:

Franck Marle

Marija Jankovic

Maik Maurer

Danilo Marcello Schmidt

Udo Lindemann

Distributed by

Carl Hanser Verlag

Postfach 86 04 20, 81631 Munich, Germany

Fax: +49 (89) 98 48 09

www.hanser.de

The use of general descriptive names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

While the advice and information in this book are believed to be true and accurate at the date of going to press, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;

detaillierte bibliografische Daten sind im Internet über [<http://dnb.d-nb.de>](http://dnb.d-nb.de) abrufbar.

ISBN: 978-1-56990-491-6

E-Book-ISBN: 978-1-56990-492-3

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying or by any information storage and retrieval system, without permission in writing from the publisher.

© Carl Hanser Verlag, Munich 2014

Production Management: Steffen Jörg

Coverconcept & -design: Atelier Frank Wohlgemuth, Bremen

Printed and bound by Digital Print Group O. Schimek GmbH, Munich

Printed in Germany

Table of Contents

Foreword	IX
Scientific Committee	XI

Part I: DSM Methods and Complexity Management

Applying the Lessons of Matrix Representation to Box Diagrams <i>Mark Grice, Nick Kimball, Neeraj Sangal</i>	3
A Viable System Model Perspective on Variant Management based on a Structural Complexity Management Approach <i>Fatos Elezi, David Resch, Iris D. Tommelein, Wolfgang Bauer, Maik Maurer, Udo Lindemann</i>	13
The Explainer: A Software Aid to Solve Complex Problems <i>Donald V Steward</i>	23
The integration of DSM and Axiomatic Design in product design as part of a MDM process <i>Sergio Rizzuti, Luigi De Napoli</i>	35

Part II: System Architecture and Product Modularity

Towards a Capability Framework for Systems Architecting and Technology Strategy <i>Andreas M. Hein, Yuriy Metsker, Joachim C. Sturm</i>	45
A Spectral Analysis Software to Detect Modules in a DSM <i>Somwrita Sarkar, Andy Dong</i>	55
Visualizing and Measuring Software Portfolio Architecture: A Flexibility Analysis <i>Rober Lagerström, Carliss Baldwin, Alan MacCormack, David Dreyfus</i>	65
Investment Decisions in Modular Product Development <i>Ali A. Yassine</i>	75
Complex Mechatronic Product Modeling using a Multi-Solution, Multi-Instance eXtended Conceptual Design Semantic Matrix <i>Serigne Dagne, Amadou Coulibaly, Mbaye Sene, François de Bertrand de Beuvron</i>	85

Part III: DSM in Decision-Making

Electricity Investments and Nuclear Development: Investment Choice Modeling
based on Value Creation 97

Bianka Shoaie Tehrani, Jean-Claude Bocquet, Toshimasa Tomoda

Matrix-based decision-making for compatible systems in product planning
concerning technologies for the reduction of CO₂-emissions 107

Danilo Marcello Schmidt, Sebastian Alexander Schenkl, Markus Mörtl

Modeling a decisional framework by MDMs 117

C. Leardi

Reshuffling collaborative decision-making organization using a Decision-
Decision MDM 127

Franck Marle, Marija Jankovic, Hadi Jaber

Dependency Structure Modeling Framework Using Expert Survey Based Group
Decision 137

Jukrin Moon, Dongoo Lee, Taesik Lee, Jaemyung Ahn

Part IV: Clustering and Optimization

Application of Dependency Structure Matrix to Airspace Sectorization and
Improving the Distribution of the Workload Among Controllers 149

Mahsa Farsad, Seyed Mohammad-Bagher Malaek

Modeling and Simulation of Service Systems with Design Structure and Domain
Mapping Matrices 157

Andreas Petz, Sebastian Schneider, Sönke Duckwitz, Christopher M. Schlick

A Clustering Method Using New Modularity Indices and Genetic
Algorithm with Extended Chromosomes 167

Sangjin Jung, Timothy W. Simpson

Clustering Technique for DSMs 177

*Florian G.H. Behncke, Doris Maurer, Lukas Schrenk, Danilo Marcello Schmidt,
Udo Lindemann*

Using Importance Measures of Risk Clusters to Assist Project Management 187

Chao Fang, Franck Marle

Optimal Capacity Allocation for a Failure Resilient Electrical Infrastructure 197
Yi-Ping Fang, Nicola Pedroni, Enrico Zio

Part V: Dependencies between Tasks and Processes

Estimation of Work Transformation Matrices for Large-Scale Concurrent
Engineering Projects 211
Christopher M. Schlick, Sebastian Schneider, Sönke Duckwitz

Task Dependency Risk Visualisation using DSMs 223
Paschal Minogue

Structure-based Compilation of System Dynamics Models for Assessing
Engineering Design Process Behavior 233
Daniel Kasperek, Sebastian Maisenbacher, Maik Maurer

Discovering Hidden Tasks and Process Structure through Email
Logs for DSM 243
Lijun Lan, Ying Liu, Wen Feng Lu

Part VI: Process Management in Complex Projects

Multi-Domain Matrix As A Framework For Global Product Development
Project Process 257
Sonia Kherbachi, Qing Yang

The Collaborative DSM: a new way to handle complex collaborative planning
and scheduling processes 267
Mathieu Baudin, Pierre Bonnel, Jean-Michel Ruiz

Applying DSM Methodology to improve the Scheduling of functional
integration in the Automotive Industry 277
*Thomas Gaertner, Sebastian Schneider, Christopher M. Schlick, Carsten Zibull,
Cedric Heuer*

An application of Knowledge Management in Design Structure Matrix for a
process improvement phase 287
Arsalan Farooq, S.M.O. Tavares, Henriqueta Nóvoa, António Araújo

Part VII: Managing Multiple Domains in Complex Projects

Structured Methodology for Applying Multiple Domain Matrices (MDM) to Construction Projects	299
<i>Purva Mujumdar, Prasobh Muraleedharan, J. Uma Maheswari</i>	
Designing an integrated Project, Program and Portfolio System – A Case Study of Healthcare	309
<i>Richard Grönevall, Mike Danilovic</i>	
Managing a complex project using a Risk-Risk Multiple Domain Matrix	319
<i>Catherine Pointurier, Franck Marle, Hadi Jaber,</i>	
Reciprocal enrichment of two Multi-Domain Matrices to improve accuracy of vehicle development project interdependencies modeling and analysis	329
<i>Hadi Jaber, Franck Marle, Ludovic-Alexandre Vidal, Lionel Didiez</i>	
Application of Structural Domain-Spanning Criteria in an Industrial Case-Study	339
<i>Wolfgang Bauer, Daniel Kasperek, Sebastian Maisenbacher, Maik Maurer</i>	
Approach for recirculation of testing knowledge into product development supported by matrix-based methods	349
<i>Carsten Karthaus, Daniel Roth, Hansgeorg Binz, Maximilian Schenk, Bernd Bertsche</i>	
How to assess actors for an Open Innovation-project?	359
<i>Matthias R. Guertler, Fatos Elezi, Udo Lindemann</i>	
Integrating Risks in Project Management	369
<i>Elodie Rodney, Yann Ledoux, Yves Ducq, Denys Breyse</i>	
The new global factory: A systems perspective for addressing the complexity of localization in emerging markets	379
<i>Patrick Wehner, Hillary Sillitto, Simon Harris</i>	
Author Index	389
Keyword Index	391

Foreword

We are very pleased to welcome you to the 16th edition of the international DSM Conference.

The theme of this 2014 edition is “Risk and Change Management in Complex Systems”. It is justified by the ever-growing complexity of our systems, involving the difficulty to anticipate potential indirect consequences of events, whether desired or not. Accordingly, this implies improvement of the methods and tools supporting the design and management of such systems.

Dependency and Structure Modeling (DSM) techniques focus on system elements and their interdependencies related to product, process and organization domains. They contribute to support mastering the amount of information required to better understand, model, and analyze, then make improved decisions to design and manage complex systems.

The International DSM Conference is the annual forum for practitioners, researchers and developers to exchange experiences, discuss new concepts and showcase results and tools. Hosted by Ecole Centrale Paris and organized in collaboration with Technische Universität München, the 16th edition of DSM Conference takes place in Chatenay-Malabry, France, during 2 to 4 July 2014.

Preceding this year’s DSM Conference on July 2, will be a DSM Industry Special Interest Group (DSMiSIG) Industry Day workshop. Industry participants will contribute to the gathering of views and evidence of the risks in current product operations, from lack of advanced systems integration methods.

Regular attendees of the DSM Conference series will have noticed that a significant change has been introduced for this edition. The size of the paper is now 10 pages at most, without slides. This allocation expansion has allowed authors to put more details about their ideas, approaches and results. This was supported by the peer-reviews of at least two members of the Scientific Committee.

This volume contains 37 peer-reviewed papers, that describe the recent advances and emerging challenges in DSM research and applications. They advance the DSM concepts and practice in 7 areas:

1. DSM Methods and Complexity Management
2. System Architecture and Product Modularity
3. DSM in Decision-Making
4. Clustering and Optimization
5. Dependencies between tasks and processes
6. Process Management in Complex Projects
7. Managing Multiple Domains in Complex Projects

These Proceedings represent a broad overview of the state-of-the-art on the development and application of DSM. There are a significant number of papers with industry authors or co-authors, reflecting this balance and synergy between conceptual development and real-life industrial application, which are in the genes of the DSM Conference series.

The Organizing Committee

Scientific Committee

Organizing Committee

Professor Franck Marle, École Centrale Paris, France
Dr. Marija Jankovic, École Centrale Paris, France
Dr. Maik Maurer, Technische Universität München, Germany
Danilo Schmidt, Technische Universität München, Germany
Dr. Ludovic-Alexandre Vidal, École Centrale Paris, France
Dr. Romain Farel, École Centrale Paris, France
Delphine Martin, École Centrale Paris, France
Sylvie Guillemain, École Centrale Paris, France
Carole Stoll, École Centrale Paris, France
Professor Udo Lindemann, Technische Universität München, Germany

Program Committee

All contributions in these proceedings have undergone a rigid review process. We would like to cordially thank all reviewers for their invaluable support.

Professor Tyson Browning, Texas Christian University, USA
Ramy El Behery, Shell Canada Ltd., Canada
Professor Steven Eppinger, Massachusetts Institute of Technology, USA
Professor Mike Danilovic, Jönköping International Business School, Sweden
Professor Eric Bonjour, Institut Femto-ST / Département AS2M, France
Professor Andrew Kusiak, University of Iowa, USA
Professor Udo Lindemann, Technische Universität München, Germany
Dr. Maik Maurer, Technische Universität München, Germany
Wieland Biedermann, DFG German Research Foundation, Germany
Paschal Minogue, Analog Devices B.V., Ireland
Harold Stowe, Boeing Company, USA
Professor Ali Yassine, American University of Beirut, Lebanon
Dr. Venkatachalam Senthilkumar, University of Witwatersrand Johannesburg, ZA
Professor Nitin Joglekar, Boston University, USA
Dr. Maija Jankovic, École Centrale Paris, France
Professor Franck Marle, École Centrale Paris, France
Dr. Ludovic-Alexandre Vidal, École Centrale Paris, France
Dr. Romain Farel, École Centrale Paris, France
Kaushik Sinha, Massachusetts Institute of Technology, USA
Professor Koshy Varghese, Indian Institute of Technology, Madras

The International DSM Conference is an endorsed event of the Design Society.

Part I: DSM Methods and Complexity Management

Applying the Lessons of Matrix Representation to Box Diagrams
Mark Grice, Nick Kimball, Neeraj Sangal

A Viable System Model Perspective on Variant Management based on a Structural Complexity Management Approach
Fatos Elezi, David Resch, Iris D. Tommelein, Wolfgang Bauer, Maik Maurer, Udo Lindemann

The Explainer: A Software Aid to Solve Complex Problems
Donald V Steward

The integration of DSM and Axiomatic Design in product design as part of a MDM process
Sergio Rizzuti, Luigi De Napoli

Applying the Lessons of Matrix Representation to Box Diagrams

Mark Grice, Nick Kimball, Neeraj Sangal

Lattix, Inc.

Abstract: A matrix representation is dense, abstract and lends itself well for analysis. However, it requires training and familiarity with the notation. A box diagram, on the other hand, is easier to understand but does not scale well. In this paper, we present techniques learned from the matrix representation and apply them to box diagrams. We also found that a combination of a matrix and box representation was often quite effective. The focus of this paper is architectural representation, in particular the architectural representation of software systems. Our box diagrams were customized for this purpose. However, these techniques may be useful for other kinds of systems as well.

Keywords: DSM, CA, Box Diagram, Layering, Independent Subsystems, Partitioning

1 Introduction

A Dependency Structure Matrix (DSM) representation is a neutral and highly scalable representation for showing relationships between various elements of a system. This makes it quite useful for situations that involve multiple domains and a combination of technologies. For instance, it is easy to create a multi-domain matrix (MDM) that is comprised of processes, the organizations involved, and the systems used (Waldman and Sangal, 2009). However, using a DSM requires training and skill. In addition, domain specific visualizations often lend themselves to a natural way of picturing things. For instance, a variety of diagrams exist for an intuitive representation of interacting processes (Giaglis, 2001).

While the use of DSMs for visualizing and managing software systems is relatively new (Sangal et al, 2005), box-and-arrow diagrams have long been used for visualizing software systems (Müller et al., 1994). These diagrams are intuitive, with boxes representing the entities and lines representing the relationships between the entities. However, as the number of boxes and lines increase they can become cluttered making them hard to use. We found that many techniques from the matrix representation can be carried over into box-and-arrow diagrams to make them easier to understand. In particular, we adapted the box-and-arrow diagrams for representing layers, independent components and most importantly for identifying problematic dependencies. We call this type of box-and-arrow diagram a Conceptual Architecture (CA) diagram. While the ideas presented here have been tried for software systems, we believe that many of these ideas may be suitable for non-software systems as well.

DSMs have been used in a variety of domains (Eppinger et al, 2012). The effectiveness of visual representation is often dependent on the domain of underlying systems. For instance, layering is a common pattern used in software design (Clements et al, 2003). The explicit use of interfaces is also common in software systems. Physical systems, on

the other hand, are often represented using bidirectional dependencies between components (Browning, 2001). As a result, some aspects of the CA diagrams such as layering are likely to be better at representing software systems. While this paper does not delve into architectural erosion, CA diagrams could also be useful in depicting architectural erosion.

2 Mapping between DSM and CA Diagram

For representing architectures, we configured our tool for the following conventions:

- Dependencies of a subsystem are read down a column
- Ordering is from bottom to top

This corresponds to how we often think of the architecture of software systems – entities at the top utilize the entities at the bottom. For instance, an application sits on a framework which uses a library of utilities. Furthermore, this convention will often identify the problematic dependencies above the diagonal in a matrix representation.

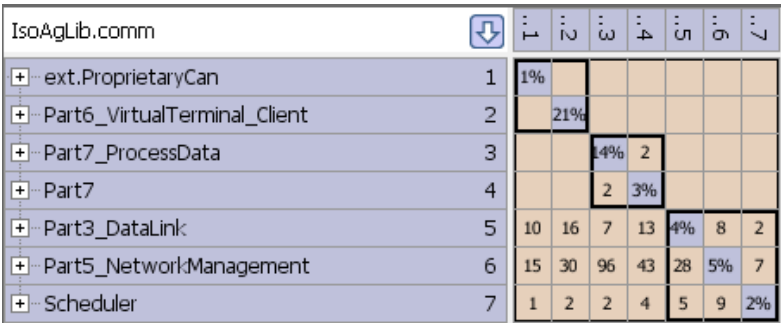


Figure 1: Partitioned DSM

Figure 1 shows a partitioned DSM. The algorithm that was applied is called a “Component Partitioner.” The highlighted boxes show the layers, the boxes with dependencies above the diagonal show the coupling, and the box without any dependencies shows independent subsystems within that layer.

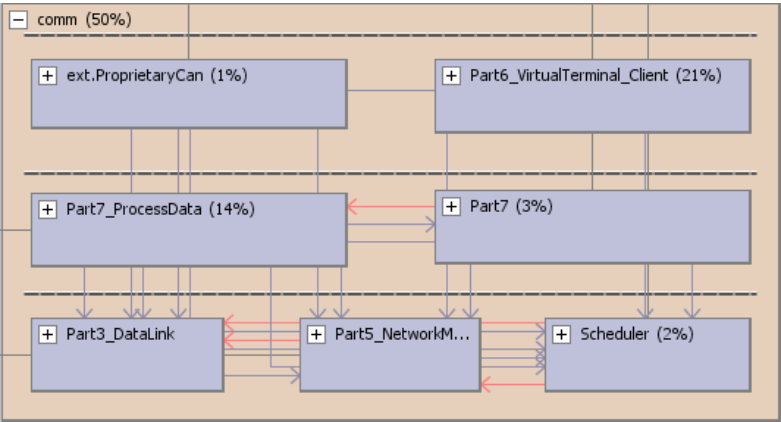


Figure 2: Partitioned CA

The same algorithm applied to a CA diagram results in a container divided into three areas each separated by a layer (dashed) line. The coupled subsystems are identified because they have dependency arrows going both to the right and to the left. Notice that the dependencies above the diagonal in Figure 1 are the ones going left in Figure 2 and are colored red. The independent subsystems are identified by a lack of dependency lines between the subsystems.

In our implementation, we left out the dependency strengths on the lines. This was done to avoid clutter. Instead, whenever a subsystem is selected, the lines to and from the selected box are highlighted along with their strength.

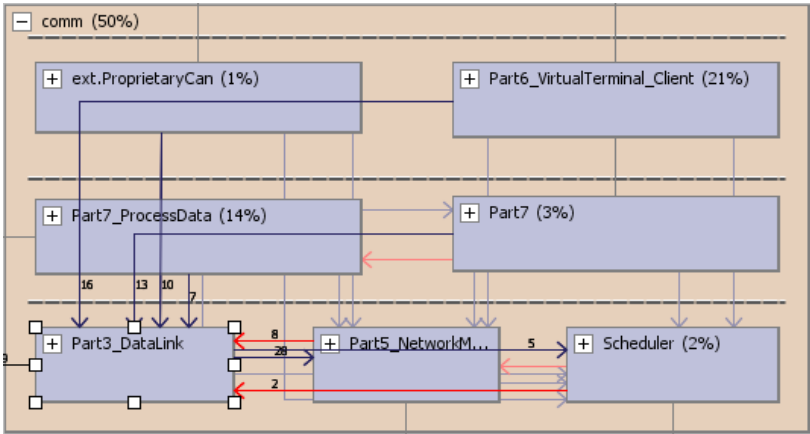


Figure 3: Highlighted Dependencies when Box selected

3 Basic Architectural Visualizations

3.1 Layering

Layering is one of the most common patterns in software architecture. Layering implies that the system is decomposed into an ordered set of components with dependencies between those components going downwards.

\$root		←	↖	↗	→
App-A	1	25%			
App-B	2	6	25%		
App-C	3	1	5	25%	
Framework	4	5	20	10	25%

Figure 4: DSM for a Layered System

In this particular case, every layer depends on every other layer underneath.

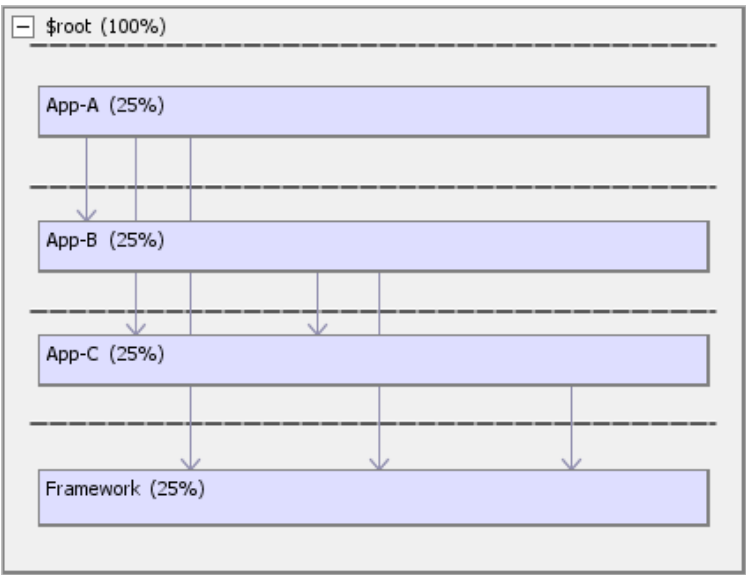


Figure 5: CA for a Layered System

Notice the placement of the dependency lines. The lines originating from a box in a higher layer are placed to the left of the lines originating from a box in a lower layer. Furthermore, the lines originating from a box are ordered so that the one which goes to

the higher layer is to the left of the lines going to a lower layer. This mimics the location of the DSM cell and makes it easier to see that every layer depends on every other layer underneath.

3.2 Independent Components

This is yet another common architectural pattern. The shared code is maintained in a common framework which in turn is used by a set of components which don't have any dependencies on each other.

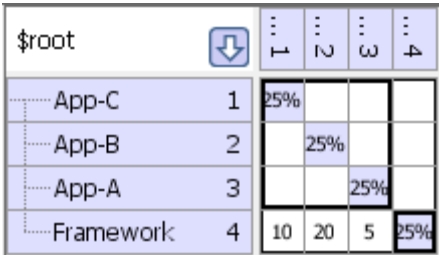


Figure 6: DSM of a System with Independent Apps Using a Common Framework

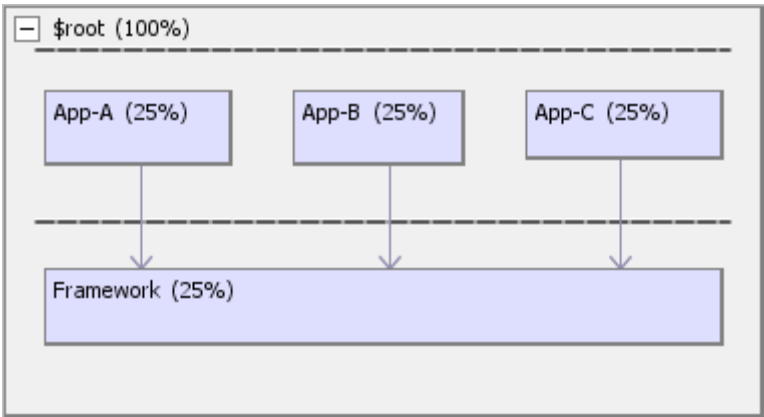


Figure 7: CA of a System with Independent Apps Using a Common Framework

The layering and the independence of the components is intuitively easy to see in the CA.

3.3 Change Propagator

The change propagator is a subsystem that has a large number of incoming and outgoing dependencies. Change propagators make systems brittle because they amplify the effect of small changes.

\$root		1	2	3	4	5
App-A	1	20%				2
App-B	2		20%			1
App-C	3			20%		1
App-D	4				20%	1
Framework	5	5	20	10	4	20%

Figure 8: DSM with Framework as Change Propagator

In a CA, a change propagator could be represented centrally to communicate its importance. However, as the number of propagators and dependencies increase, a CA can become cluttered and hard to read. We found that a combination of DSM and CA visualization worked well. The DSM was used for analysis and a CA was then used to convey the results of that analysis.

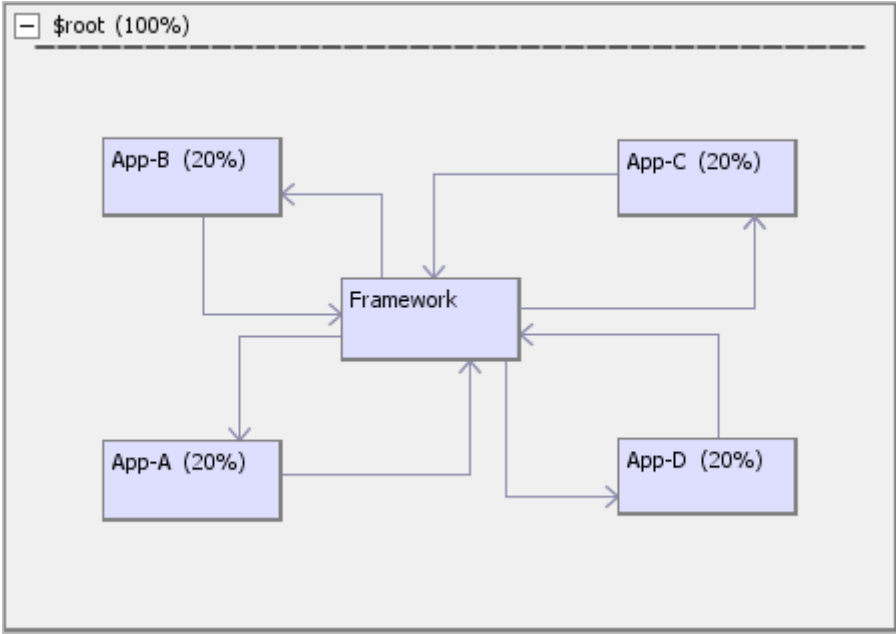


Figure 9: CA with Framework as Change Propagator

It should further be noted that we did not use color to distinguish between upward and downward dependencies in this diagram. Colors could be applied to certain lines after an analysis that deduces them as problematic dependencies.

3.4 Identifying Problematic Dependencies

Most real world software systems that have been around for some time show signs of architecture erosion. Layered systems have backward dependencies. Independent components end up with dependencies on each other that make it harder for teams to work independently. Identifying problematic dependencies is often a critical part of architectural analysis.

\$root		└	↳	ω	⤴
App-A	1	25%			1
App-B	2	6	25%	1	
App-C	3	1	5	25%	
Framework	4	5	20	10	25%

Figure 10: DSM of an Imperfectly Layered System

The dependencies above the diagonal are the problematic dependencies. The strength of the dependencies is typically smaller for problematic dependencies because they are against the intended design.

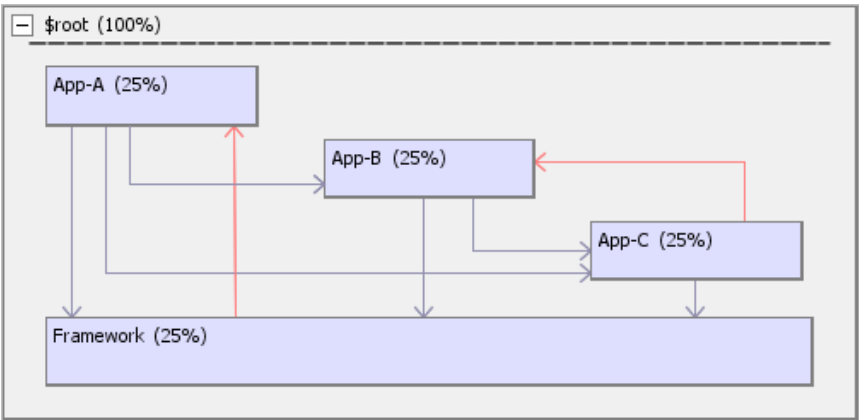


Figure 11: DSM of an Imperfectly Layered System

The problematic dependencies point upwards and are displayed in red. Again, we found that ordering dependencies like a DSM makes this diagram easier to peruse and understand.

4 Real World Example

ISOAgLib is an open source software library used in many devices with embedded software. It is a library that implements an ISO standard for electronic data communication in mobile machinery.

\$root									
+ supplementary_driver			1	19%	5	1	1		
- IsoAgLib	+ comm		2	1	50%		20		
	+ hal		3	89	5	24%	18	2	1
	+ driver		4	7	96	52	4%		
	+ util		5	42	248	32	34	2%	1
	+ *		6	17	99	27	17	6	.3%

Figure 12: DSM for ISOAgLib

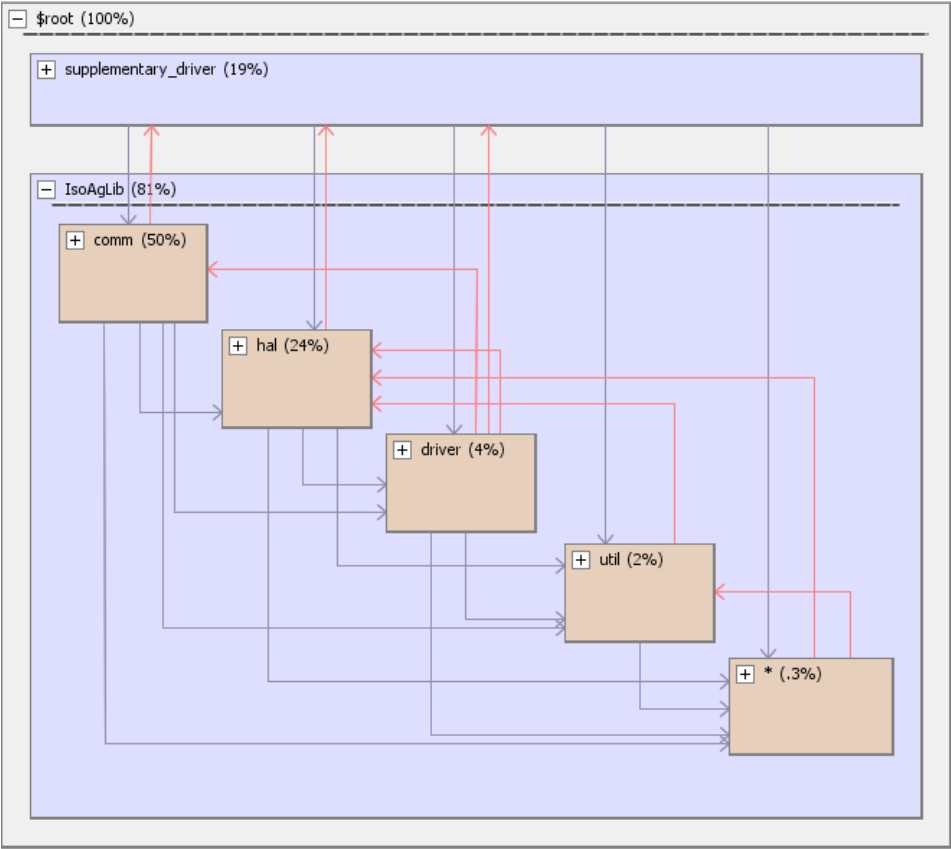


Figure 13: CA for ISOAgLib

Notice how the layout of the boxes mimics how problematic dependencies show up above the “diagonal.” This diagram conveys the layered nature of the system while highlighting the dependencies that show architectural erosion.

5 Conclusion

Techniques from matrix-based visualization can be used to visualize the architecture of software systems using box diagrams. The key is to use the ordering of subsystems in the ordering of boxes and dependencies. Box diagrams provide flexibility in terms of sizing systems and in terms of their placement. As a result, it is possible to create intuitive diagrams that also highlight key architectural issues. While DSMs are useful in architectural discovery, CAs intuitively convey the overall design. Together, DSM and CA are a powerful combination for visualizing the architecture of software systems.

References

- Browning, T.R., 2001. "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions". IEEE Transactions on Engineering Management, Vol. 48, No. 3, August 2001.
- Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J., 2003. "Documenting Software Architectures Views and Beyond". 1st edition 77-91. Pearson Education.
- Eppinger, S.D. and Browning, T.R., 2012. "Design Structure Matrix Methods and Applications". The MIT Press, Cambridge Massachusetts.
- Giaglis, G. M., 2001. "A taxonomy of business process modeling and information systems modeling techniques." International Journal of Flexible Manufacturing Systems 13.2 (2001): 209-228.
- Müller, H. A., Wong, K., and Tilley S. R., 1994. "Understanding software systems using reverse engineering technology." The 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences Proceedings (ACFAS 1994).
- Sangal, N., Jordan, E., Sinha, V. and Jackson, D., 2005. "Using Dependency Models to Manage Complex Software Architecture". Proceedings of the 20th annual ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications, Pages 167-176, San Diego, California, October 2005.
- Waldman, F. and Sangal, N., 2009. "Applying DSM to Enterprise Architectures". 11th International DSM Conference, Clemson, October 2009.

A Viable System Model Perspective on Variant Management based on a Structural Complexity Management Approach

Fatos Elezi¹, David Resch¹, Iris D. Tommelein², Wolfgang Bauer¹, Maik Maurer¹,
Udo Lindemann¹

¹ Institute of Product Development, Technische Universität München, Germany

² Department of Civil and Environmental Engineering, University of California,
Berkeley, USA

Abstract: This paper explores the applicability of Structural Complexity Management (StCM) on organizational design and diagnosis. As basic structural model for efficient management of organizations the Viable System Model (VSM) is used. The VSM represents an alternative organization model based on Management Cybernetics (MC) theory that describes the structure of all viable systems. Companies operating in dynamic environments strive for viability, therefore incorporation of VSM into their structure is essential. However, VSM requires complex communication and control structures that are not so intuitive at first sight. A methodology that supports the identification and analysis of these structures is still missing, which is why the VSM has not gained wider popularity. This paper addresses a methodology based on StCM that can prove to be beneficial for this purpose. The methodology is applied to an industry case study, where first improvement suggestions based on the newly derived insights are shown.

Keywords: management cybernetics, viable system model, structural complexity management, Multiple-Domain Matrix, variant management

1 Introduction

The hierarchical organization structure that is still dominating the organization charts of most companies shows its limitations in contemporary society. More than ever before, the companies are required to adapt fast to new environmental circumstances and this adaptability and evolution is not supported by hierarchical management structures. The hierarchical form of command and control is unnatural to complex evolutionary systems (Espinosa et al., 2007). Viable System Model (VSM) (Beer (1972, 1979, 1985)) is the alternative that managers should turn to for a new way of interpreting their organization, for this model allows cybernetic perspective and specifies how the structure of an organization should look like in order to survive in its environment – adapt and evolve in the turbulent environments. The VSM is a functional model that specifies functions (e.g. operating, controlling, adapting) and the relationships between those functions that all viable systems have in common. VSM is developed as a generic model in analogy to the human nervous system, consequently it can not only be applied to companies but all kinds of organizations, institutions, etc. (Leonard, 2009). Commonly, the VSM is used to diagnose existing or design new organizations by aligning the real-life organization with

the characteristics that all viable systems share. Although VSM was introduced forty years ago, its strength to address contemporary management challenges such as complexity, learning and adaptability has gained importance in recent years. As Gould (1999) states, “Probably the single most important thing about Beer’s work is that it anticipates much of the current fascination with chaos, complexity and the need for rapid strategic adjustments to environmental changes.”

Despite these advantages, VSM is still not widely spread among management practitioners. In the literature there are several successful attempts to apply VSM as basis for analyzing (read: diagnosing) the companies and public organizations. There are excellent recent publications on this topic and an active group of followers. Yet, the VSM has failed to enter the management research mainstream. People find the theory difficult to understand and do not succeed in the application of the model in practice (Anderton, 1989). While idea of having a generic model to analyze the viability of any kind of organization seems tempting, the actual application of the VSM in a particular management system can be cumbersome and prone to misinterpretations.

What would be required to make VSM applications more practical? The VSM lacks certain pragmatic tools and methods to take it from a rather theoretical model to an easy-to-apply framework. Major challenges of applying the VSM are linking the information flow to the organizational structure and the analysis and visualization of structural elements (Pfiffner, 2010). New methods and tools can help to overcome those challenges and promote the analysis of organizations with VSM knowledge. Several contributions set out to enhance the applicability of the VSM (Brocklesby and Cummings, 1996; Clemson, 1994; Espejo, 1989; Espejo et al., 1999; Schwaninger, 2009), yet all these contributions anchor their analysis on the same abstraction level as VSM model itself. This level of abstraction is quite intangible to practitioners, since they think and analyze at a much detailed level and always in a context related management systems. Structural analysis – examining the elements and their relationships in any kind of system – can offer the methods and tools needed to create structural transparency and depict relationships in such a detailed level. A Structural Complexity Management (StCM) approach that supports analysis and diagnosis of organisational structures based on VSM was recently introduced (Elezi et al., 2013). This approach focuses on modeling organizational structure, processes and communication in a pragmatic way and fosters the use of the VSM for handling organizational complexity issues. In this paper, we describe the preliminary results from implementation of this methodology in a variant management context.

The paper is structured as follows. The second section serves as a short reminder of the methodology described in (Elezi et al., 2013). Third section is about the (partial) application of the methodology in a real case study and discusses the results. Fourth section draws conclusions and proposes an outlook for further research.

2 Methodology

Structural Complexity Management (StCM) was introduced as a framework to manage and optimize system complexity (Lindemann et al., 2009). It combines matrix-based methodologies (i.e. DSM and DMM) to an approach based on the Multiple-Domain Matrix (Maurer and Lindemann, 2007). This approach enables modelling complex

systems that consist of multiple domains connected by various relationship types. The methodology has been applied several times so far and various adaptations have been developed, including waste reduction in product development (Elezi et al., 2011), mapping lean construction processes (Furtmeier et al., 2010) and creation of organizational modularity (Krinner et al., 2011), among others.

StCM is usually used in product design to make product structure, processes and other domains more transparent and identify structural constellations that are important for these domains. Multiple-Domain Matrix is also used in analysing and comparing organizational architectures and other domains with great success (Eppinger and Browning, 2012). In this paper, the StCM approach is used for making organizational structure transparent so the differences from VSM structure are identified and the control information flow is represented. Therefore, the following adapted methodology depicted in figure 1 is proposed.

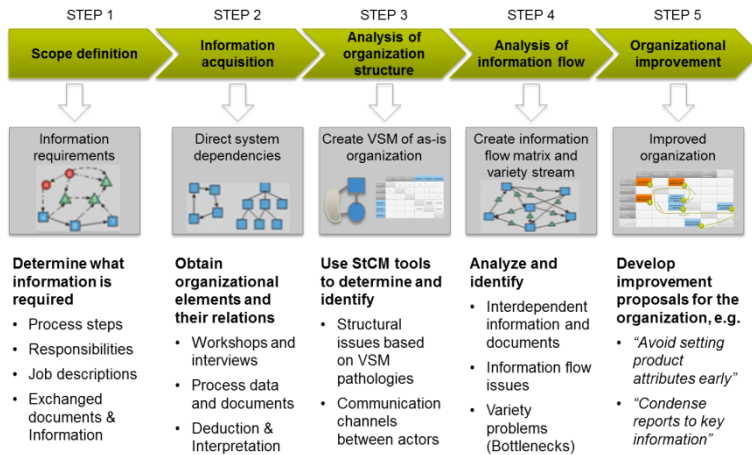


Figure 1. Adapted structural complexity management methodology, adapted from (Lindemann et al., 2009)

STEP 1: The first step of the methodology is to define the scope of the project. First, participants should be clear on what is the subject of the analysis. Therefore, the VSM identity is identified. This is an essential step for setting the boundaries of the analyzed system (Pérez Ríos, 2010).

STEP 2: Available information on organization structure (e.g. employee roles, departments, hierarchy levels) is collected and processed in order to supply the underlying matrices of the MDM with data. The goal of this step is to obtain the direct relationships between employees, their departments and their VSM systems.

STEP 3: The objective of this step is to map the collected data to VSM sub-systems and to compare the actual communication channels with the ones characteristic in viable systems. The activity steps are grouped by the corresponding actors with the information collected in the activity-actor DMM. Additional responsibilities of actors originating from job descriptions that are not reflected in the activity steps are added to the grouping

as well. The responsibilities of each actor are then matched to the correlating VSM sub-systems that typically perform such responsibilities.

STEP 4: After building a VSM of the system-in-focus and comparing stationary communication channels in the previous stage, this step analyzes how information moves between activities. Input and output information of activities is examined, interdependent information is identified and activities connected to each other through information flow are calculated.

STEP 5: The last step of the methodology is to suggest the organizational improvements that should close the gap between the current state and the ideal organization structure that derives from VSM theory. Characteristic situations where an organization deviates from an ideal viable system are referred to as *pathologies* (Pérez Ríos, 2012).

As shown in previous section, one of the main barriers for application of VSM in practice is the analysis and visualization of structural elements and their relationships. The proposed methodology can tackle those challenges and provide a value-adding contribution to the diagnosis and design of organizations. It can be the pragmatic approach needed to take the VSM from a rather theoretical to a more practical approach.

3 Case Study

Case study background and setup

The company in focus is one of the largest manufacturer of household appliances in the world, with 45,000 employees in 50 countries and around 10 Bn€ revenue in 2011. Company's product range is split up into divisions for cooking, refrigeration, dishwashers, laundry and small appliances. The focus of this case study lies on the refrigeration department (RD), which alone offers ca. 1000 variants of freezers, refrigerators and combined refrigerators. The process analyzed in this paper is the variant management of RD. The objective of this case study is to first by using the above-mentioned methodology extract the as-is functional structure of the variant management in RD. Further, the "as-is" functional structure is then compared with the "should-be" functional structure (derived from VSM model). A gap analysis with the "should-be" structure should identify organizational pathologies and issues with communication channels.

3.1 Step 1 – Scope Definition

At the start of the case study, the organization structure of the company was unfolded into recursion levels. The company organization unit on recursion level 1 was first unfolded into commercial divisions to form recursion level 2. Then, these divisions were unfolded by a geographical recursion criterion into the various factory locations on recursion level 3. Finally, the factories were split up into assembly lines, each of which should again form a viable system on its own. The results of this complexity unfolding process are depicted in figure 2.

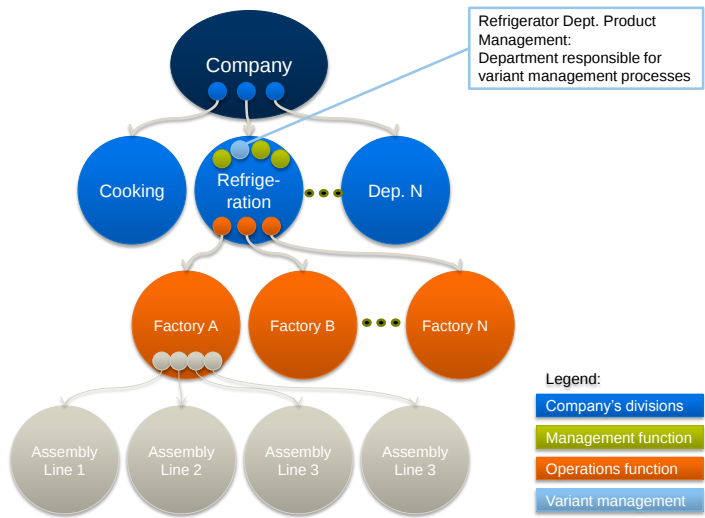


Figure 2. Recursion levels in the company

As the case study should focus on variant management for the RD division and these activities are mostly handled centrally by RD management, this division was selected to be the system-in-focus for the case study. From a cybernetic perspective, variant management can be interpreted as a large control loop. The controlled system is RD operations, which develops and produces refrigerators and freezers. The output of the system is measured in terms of revenue and profits. The controller (RD variant management) monitors this output and adjusts the product portfolio by introducing new variants, refreshing the variants, or discontinuing existing variants. Thus, the input of the system is modified and the control loop is closed. In addition to determining the system-in-focus and clarifying its purpose, a multiple-domain matrix was defined as proposed in (Elezi et al., 2013) . The incorporated domains were activities, actors and information.

3.2 Step - Information acquisition

The information on the four key activities received in the workshop and additional data from organization charts, job descriptions and process documentation was incorporated into the MDM defined in the first step. An activity-activity DSM and three DMMs that reflect the connections between activities, actors and information were created. A schematic overview of the MDM filled with acquired data (and DSMs calculated subsequently) is shown in figure 3. The MDM was subsequently validated with the variant management employees at RD.