DAGSTUHL
REPORTS

Volume 7, Issue 12, December 2017

ISSN 2192-5283

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at
http://www.dagstuhl.de/dagpub/2192-5283

Publication date
March, 2018

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at `http://dnb.d-nb.de`.

Digital Object Identifier: 10.4230/DagRep.7.12.i

Report from Dagstuhl Seminar 17491

Computational Metabolomics: Identification, Interpretation, Imaging

Edited by

Theodore Alexandrov[1], Sebastian Böcker[2], Pieter Dorrestein[3], and Emma Schymanski[4]

1 **EMBL Heidelberg, DE,** theodore.alexandrov@embl.de
2 **Universität Jena, DE,** sebastian.boecker@uni-jena.de
3 **UC – San Diego, US,** pdorrestein@ucsd.edu
4 **University of Luxembourg, LU,** emma.schymanski@uni.lu

Abstract

Metabolites are key players in almost all biological processes, and play various functional roles providing energy, building blocks, signaling, communication, and defense. Metabolites serve as clinical biomarkers for detecting medical conditions such as cancer; small molecule drugs account for 90 % of prescribed therapeutics. Complete understanding of biological systems requires detecting and interpreting the metabolome in time and space. Following in the steps of high-throughput sequencing, mass spectrometry (MS) has become established as a key analytical technique for large-scale studies of complex metabolite mixtures. MS-based experiments generate datasets of increasing complexity and size.

The Dagstuhl Seminar on Computational Metabolomics brought together leading experts from the experimental (analytical chemistry and biology) and the computational (computer science and bioinformatics) side, to foster the exchange of expertise needed to advance computational metabolomics. The focus was on a dynamic schedule with overview talks followed by break-out sessions, selected by the participants, covering the whole experimental-computational continuum in mass spectrometry. Particular focus in this seminar was given to imaging mass spectrometry techniques that integrate a spacial component into the analysis, ranging in scale from single cells to organs and organisms.

Seminar December 3–8, 2017 – http://www.dagstuhl.de/17491
1998 ACM Subject Classification J.3 Life and Medical Sciences
Keywords and phrases algorithms, bioinformatics, cheminformatics, computational mass spectrometry, computational metabolomics, databases, imaging mass spectrometry
Digital Object Identifier 10.4230/DagRep.7.12.1
Edited in cooperation with Marcus Ludwig

1 Executive summary

Theodore Alexandrov
Sebastian Böcker
Pieter Dorrestein
Emma Schymanski

Metabolomics is the study of metabolites (the small molecules involved in metabolism) in living cells, cell populations, organisms or communities. Metabolites are key players in almost all biological processes, play various functional roles providing energy, building blocks, signaling, communication, and defense and serve as clinical biomarkers for detecting medical conditions such as cancer. Small molecule drugs (many of which are derived from metabolites) account for 90 % of prescribed therapeutics. Complete understanding of biological systems requires detecting and interpreting the metabolome in time and space.

Mass spectrometry is the predominant analytical technique for detecting and identifying metabolites and other small molecules in high–throughput experiments. Huge technological advances in mass spectrometry and experimental workflows during the last decade enabled novel investigations of biological systems on the metabolite level. Research into computational workflows, the simulation of tandem mass spectra, compound identification and molecular networking have helped disentangle the vast amount of information that mass spectrometry provides. Spatial metabolomics on different spatial scales from single cells to organs and organisms has posed data analysis challenges, in particular due to an unprecedented data volume generated that grows quadratically with the increase of spatial resolution.

Continued improvements to instruments, resolution, ionization and acquisition techniques mean that metabolomics mass spectrometry experiments can generate massive amounts of data, and the field is evolving into a "big data" science. This is particularly the case for imaging mass spectrometry, where a single dataset can easily be many gigabytes or even terabytes in size. Despite this dramatic increase in data, much of the data analysis in metabolomics is still performed manually and requires expert knowledge as well as the collation of data from a plethora of sources. Novel computational methods are required to exploit spectral and, in the case of imaging, also spatial information from the data, while remaining efficient enough to process tens to hundreds of gigabytes of data.

Dagstuhl Seminar 17491 on Computational Metabolomics: Identification, Interpretation, Imaging built on the success of the first Computational Metabolomics Dagstuhl Seminar (15492) in 2015. A number of topics overlapped with the 2015 seminar, while the focus on imaging introduced new perspectives, participants and topics. In contrast to the first seminar, 17491 was a large seminar, with 45 very active participants and a large portion of young scientists. From the first hours of the seminar, effort was made to integrate these young scientists in the discussions and presentations and this paid off leading to lively discussions involving all participants. Many participants were new to Dagstuhl and the concept of Dagstuhl seminars, which led to a seminar that was a combination of being semi-structured and spontaneous. Very positive feedback was received from all during a comprehensive feedback session before lunch on Friday, including constructive ideas for a new focus for a possible new seminar in 2019.

On the scientific side, the seminar covered numerous topics which were found to be most relevant for the computational analysis of mass spectrometry data, and ranged from the "dark matter in metabolomics" to "integrating spatial and conventional metabolomics"; see the full report for a comprehensive description.

The seminar has fully achieved its key goals: to foster the exchange of ideas between the experimental and computational communities; to expose the novel computational developments and challenges; and, to establish collaborations to address grand and priority challenges by bridging the best available data with the best methods.

2 Table of Contents

3 Overview of Talks

3.1 Challenges in “conventional” metabolomics

Corey Broeckling (Colorado State University – Fort Collins, US), and Nicola Zamboni (ETH Zürich, CH)

Metabolomics is a field of research which relies on a broad collection of preparation and analytical approaches and techniques. We began this conference by outlining the range of approaches used and provide a quick overview of the problems with metabolomics that may be addressed using computational approaches. We identified computational opportunities in the following areas:

1. Increase in breadth and coverage of metabolomics
2. workflows for efficient, reusable and objective annotation
3. processing standards for interoperability and testing
4. systematic analysis of MS features
5. network-driven data mining
6. standardization / normalization of non-targeted metabolomics

3.2 Challenges in spatial metabolomics

Theodore Alexandrov (EMBL Heidelberg, DE)

Spatial metabolomics is about capturing metabolome in its full complexity across various spatial scales. Currently there are several methods, in particular imaging mass spectrometry, which generate 1 TB of data per sample. Challenges are:

1. the interpretation of data,
2. data curation, and
3. multi-omics integration.

This requires cloud computing and collaboration.

3.3 Challenges in environmental exposomics and environmental cheminformatics

Lee Ferguson (Duke University – Durham, US)

Identification of unknown pollutants and toxicants in environmental and biological samples is complicated by incomplete molecular databases. Difficulty in prioritizing chemicals used in commerce or causing adverse effects. Advances in mass accuracy, resolution and data acquisition rates have increased data acquisition rates, but annotation remains difficult. Priorities for future advancements include the incorporation of false discovery rates for both *in silico* and library-based MS/MS identification strategies. Incorporation of metadata for molecular candidate prioritization will be critical to enhancing identification rates in environmental samples. Computational methods for establishing networks associated with molecules in environmental systems will be vital for understanding relationships among pollutants.

Emma Schymanski (LCSB, University of Luxembourg, LU)

The dark matter in environmental (and small molecule) samples is still a huge challenge. We have untreated wastewater discharged into rivers with measurable toxic effects that can not be clarified with known (target) chemicals. Homologues (UVCBs, complex mixtures) are a massive part of this dark matter with thousands of homologous series. On the other hand we have 100,000s of complex chemical mixtures produced in thousands of tonnes where we cannot even assign a structure to the complex name. How can we reconcile this?

3.4 False discovery rate estimation

Sebastian Böcker (Universität Jena, DE)

FDR allows for an automated, objective and reproducible estimation of "how unsure we are": This is accepting the fact that in science, there is no "absolutely sure". It is build on the assumption that the score of the hit is allowing us to discriminate between true hits (correct identifications) and bogus hits (incorrect identifications). For metabolomics we face a number of issues and challenges:

1. ID rates are still much smaller for *in silico* tools than in, say, proteomics.
2. Spectral libraries are notoriously incomplete.
3. Separation by score is significantly worse than, say, in proteomics.
4. We have no ideas how to generate decoy structures, or how to transform them into fragmentation spectra.

Andrew Palmer (EMBL – Heidelberg, DE)

Many scoring systems exist for measuring the quality of match between experimental data and reference databases, for example tandem spectra of isotope patterns. At some point a threshold must be established for those scores in order to separate the comparisons into "interesting" and "uninteresting". We discussed some newly developed approaches for estimating the false discovery rate for such comparisons in metabolomics experiments, in particular the prediction and evaluation of target-decoy approaches and the requirements for accurate estimation. It is clear that no approach is perfect but the community agrees that quantifying database matching performance is essential.

3.5 Computational challenges in environmental metabolomics and exposomics

David Wishart (University of Alberta – Edmonton, CA)

Humans are exposed to all kinds of chemicals throughout their lifetimes. These "environmental" exposures account for many of the chronic diseases that develop later in life. In the USA, more than 90% of all deaths (in 2013) could be attributed to some kind of chemical, biological or environmental exposure. The measurement of chemical exposures – both within the body and outside the body – is called exposomics. In this presentation I presented a brief overview of exposomics and identified 6 key challenges that are facing the field. These include:

1. the problem with automated workflows
2. the missing "pure" compound problem
3. the missing "metabolized" compound problem
4. the missing "observable" problem
5. the missing ontology problem and
6. the missing funding problem.

Potential solutions for each of these challenges are presented.

3.6 Improved molecular networks with LC-MS feature detection and *in silico* annotation

Louis-Felix Nothias-Scaglia (UC – San Diego, US)

Recent advances and challenges in MS-data preprocessing for molecular networking on GNPS web-platform (http://gnps.ucsd.edu) were presented and discussed during the session. This processing step improves qualitatively the molecular networks by filtering-out noisy features, by reducing the data redundancy, and by enabling the discrimination of isomeric ions based on the retention time. Additionally, this approach allows the integration of semi-quantitation in the network which is a key for new mining strategy, such as the "bioactive molecular networking" (Nothias, J. Nat. Prod., 2018, accepted). One of the most exciting outcome of that development relies, is the possibility of using *in silico* annotation tools such as Sirius / CSI:FingerID (Dührkop, PNAS, 2015) or Network Annotation Propagation. The biggest challenge remains the need to optimize LC-MS feature detection parameters on a dataset-basis, which hamper the systematic large-scale use of that approach on all public dataset. The development of a LC-MS processing tool that would include an "auto-tuning" feature is needed to solve that bottleneck.

3.7 Issues in MS1 data processing and annotation

Julijana Ivanisevic (University of Lausanne, CH), Michael Andrej Stravs (Eawag – Dübendorf, CH)

A presentation and subsequent discussions highlighted current unsolved issues in MS1 data processing, including:

- feature detection – i.e. chemical and bioinformatics noise
- feature annotation – issues with componentization, i.e. grouping of isotopes, adducts etc.
- batch correction as a problem of both experimental and computational approaches.

In the presentation it was pointed out that only small portion of acquired MS1 data is indeed assigned as isotopes, adducts, etc. The amount of non-annotated MS1 data remains extremely high (at least – more than a half of detected signals) implying the presence of noisy features (i.e. detector artifacts, data artifacts) but especially that the redundancy is still poorly annotated like in the case of multiple charged species, in-source fragments, etc. Discussions highlighted the need for solid annotated test data necessary for the development of improved computational approaches for feature detection and annotation.

3.8 *In silico* structure prediction with CSI:FingerID

Kai Dührkop (Universität Jena, DE)

Identifying metabolite structures via tandem MS is one of the main challenges in metabolomics. *In silico* / combinatorial fragmenters start from a hypothetical structure (taken from a structure database) and match it against the measured spectrum, reporting some kind of likelihood or match score. In contrast, CSI:FingerID predicts a hypothetical structure (in form of a probabilistic molecular fingerprint) directly from the measured spectrum using machine learning techniques. This allows to deal with so called “unknown unknowns” - structures which are not contained in any structure database. But it is also a starting point to extract knowledge from MS/MS data without the necessity to identify the exact structure. We discussed about visualization of predicted structures and possible applications for structure prediction.

3.9 Long-term monitoring of aquatic systems, combining LC-HRMS and chemometric tools to highlight organic pollutants

Martin Loos (looscomputing – Dübendorf, CH)

Liquid chromatography coupled to high-resolution mass spectrometry has become the method of choice to trace highly diluted (yet possibly toxic) organic micropollutants. Despite their widespread release, data mining workflows to prioritize trends of concern with respect to increasing pollutant concentrations have remained scarce within an environmental monitoring context. Here, we have presented certain steps within any such workflow to automatize, streamline and facilitate the fast detection of such patterns, even from n $\geq$ 1000 LC-HRMS files.

3.10 Topic modelling for substructure discovery in metabolomics data: state of the art and challenges ahead

Justin van der Hooft (Wageningen University, NL)

Mass Spectrometry-based metabolomics workflows result in large amounts of data often containing fragmentation spectra of many detected molecules. Ever since fragmentation spectra of biomolecules could be produced, data analysts have been looking for specific fragmentation patterns that they could couple to key structures or substructures in their samples. Doing so, they could start to structurally annotate the molecules in a complex biological mixture. However, the manual analysis of MS/MS spectra is tedious and impossible when faced with over 5000 MS/MS spectra for each sample. To overcome this hurdle, computational approaches have been proposed over the last years. The application of topic modelling, originally used for text-mining, to MS/MS data was recently introduced. In this talk, the MS2LDA algorithm was introduced: concurring mass fragments and/or neutral losses defined from fragmented molecules are discovered and grouped into Mass2Motifs (similar to topics in text-mining). This is the first unsupervised approach that enables the detection of potential substructures in mass spectrometry fragmentation data. Validation results using standards from MassBank and GNPS were shown, as well as Mass2Motifs discovered in beer samples. Indeed, MS2LDA found biochemically relevant substructures that could be annotated with amino acid, sugar, and aromatic moieties, amongst others. Furthermore, it was shown that MS1 comparisons can be mapped on the Mass2Motifs, thereby guiding the user to relevant/interesting Mass2Motifs, for example, those more present or absent in Indian Pale Ale (IPA) beers. Finally, it was discussed how to best take this approach forward, in particular regarding annotation of the discovered fragmentation patterns. As is true for text-mining, the discovered Mass2Motifs (topics) are a collection of fragments/losses (words) that need to be interpreted by the user. It was concluded that further integration with other tools and efficient storage of annotated Mass2Motifs are prerequisite for full exploitation of this innovative approach.

3.11 Integrating mass spectrometry with other imaging modalities: Improving biological insight through data-driven multi-modal image fusion

Raf Van de Plas (TU Delft, NL)

Studies in medicine and biology increasingly employ a multitude of different imaging technologies to answer a specific biological question. A growing number of such multimodal imaging studies include imaging mass spectrometry (IMS) as one of these modalities. Although different modalities are routinely registered and overlaid to generate a single display, true integration of data across technologies is largely left to human interpretation, resulting in a significant underutilization of the potential of multi-modal measurements. This talk gives an overview of our recent work on the integration or "fusion" of IMS with measurements from other imaging modalities (Van de Plas et al., Nature Methods, 2015) and demonstrates the potential of data driven image fusion for IMS through several predictive applications. Example applications include:

1. the "sharpening" of IMS images, using microscopy measurements to predict ion distributions a la spatial resolution that exceeds that of measured ion images by ten times or more;
2. the enrichment of biological signals and the removal of instrumental noise by multi-modal corroboration; and
3. the prediction of ion distributions in tissue areas that were not-measured by IMS.

We also highlight more recent work in which contrary to fusing IMS with microscopy, our data-driven fusion method is used to combine liver mass spectrometry-based modalities into a single predicted modality that combines advantages of the several modalities. In this new IMS-IMS fusion setting, MALDI-FTICR IMS measurements (lower spatial resolution, higher mass resolution) enabling ion distributions to be predicted with both high spatial as well as high mass resolution. Examples are shown in lipid imaging, where there is both a need to spatially resolve fine tissue structure, as well as a need for high chemical specificity due to nominal isobaric species.

4 Working groups

4.1 Metabolite structure ambiguity – representation, standardization, and naming

Nils Hoffmann (ISAS – Dortmund, DE)

The structure of small molecules can not be fully established with current MS/MS methods. Ideally, structures should be unambiguously resolvable down to the isomer level, however, current mass spectrometric methods are limited to measuring accurate masses of precursor and fragment ions. MSn fragmentation patterns can assist in the elucidation of structures, but still fail to identify e.g. the positions of double bonds or specific ligands, since the corresponding fragment masses are virtually identical.

For lipids specifically, ambiguities exist concerning the position of double bonds in lipid chains and how to encode them in a consistent naming scheme. The current nomenclature uses lipid category abbreviations (e.g. PC) and encodings for the number of Carbon atoms in the fatty acid (FA) side chains (R1, R2, R2 ...) of a lipid and optionally, the number of double bonds (unsaturated bonds) in them, e.g. for a triglycerol with a total of 52 carbon atoms and one double bond, the name would be reported as TG(52:1), or as TG(16:0_18:0_18:1) if is known that one of the fatty acid chain consists of sixteen carbon atoms, the second FA chain of eighteen, and the third one of eighteen carbon atoms with one double-bond at an arbitrary position.

In principle, the same issues arise not only in lipidomics, but similarly for other "small molecules" (<1kDa) in environmental chemistry, glycomics, natural product chemistry (flavonoids and terpenes) and metabolomics. We identified the following, cross-cutting requirements for a nomenclature and representation of incomplete or ambiguous structural information:

1. representation of such information as extended SMILES / SMARTS and InChI (with extension of the current standard),
2. visualization of generalised structures, e.g. using CDK-depict, based on extended SMILES or other structural representations,
3. curation and availability of uncertain structures in databases,
4. support for reporting of ambiguity / structural uncertainty in community data standards, e.g. in mzTab, mzIdentML etc.,
5. search of patterns, e.g. conserved and variable substructures in structure databases
6. collapsing / superposition of defined, unique structures represented as a common conserved and a "common" varying part (e.g. exact ligand positions in Markush structures).

We want to address these requirements by gathering examples from the different communities that reflect the status quo of reporting ambiguous, not fully-resolved structures, especially when identification is only based on MS1 / MS2 database identification. We intend to use those to illustrate the benefit of having a common standard in a statement article that defines the problems and shortcomings of current reporting of structural information and raises awareness in and receives input from the affected communities.

We will work towards better interoperability between the tools used to generate extended SMILES (support for Rs, *s, etc. in the CDK, ChemAxon, and OpenBabel) and to support the encoding of uncertainty in ligand positions, e.g. for aromatic compounds, and to visualize them.

4.2 R-based computational mass spectrometry

Michael Andrej Stravs (Eawag – Dübendorf, CH)

Features and benefits of the base package MSnlib for treatment of mass spectrometry data were introduced, with a particular focus on the classes for representation of MS1 and MS2 spectra. A brief overview about features and interface of the new XCMS3 version were presented. A discussion highlighted a current gap in packages for MS2 library management and search, and adoption/extension of current approaches was discussed. Finally, the current trend of "tidyverse" packages was briefly exposed, and benefits and drawbacks of a possible adoption of "tidyverse" data structures were discussed.

4.3 Connecting genome and metabolome data: combining structural information from genome and metabolome mining

Justin van der Hooft (Wageningen University, NL)

Metabolomics workflows result in the discovery of molecular families sharing common core structures. Genome mining workflows result in the prediction of biosynthesis gene clusters responsible for the production of specialized molecules – those gene clusters can then be grouped in gene cluster families that produce similar molecules. Connecting genome and metabolome mining workflows can enhance the structural and functional annotation and identification of both metabolites and genes. By linking existing and novel networking approaches, accelerated substructure annotation can be accomplished, which will facilitate structural elucidation of specialized molecules. Through the link with the genome, the microbial producers of these molecules can also be linked. This break-out session discussed ideas and major challenges to exploit the linkage between these two largely separately developed omics fields: the availability of paired data sets including validated links between genome and metabolome; the type of statistics needed to correct for potential bias when correlating presence/absence of molecular and gene cluster families across different strains; the application of fragmentation trees to find substructures connecting to genes as well as information on how substructures are linked; the application of biotransformation rules to both substructures and complete structures; the use of (a subset of) CSI:FingerID substructures to assess the likelihood that they are present in fragmented molecules; and the possibility to also use transcriptomics data to assess if BGC are active or silent. It was concluded that the currently available paired data sets are scattered and not easily accessible: cross-linking tables and validated links are needed to apply machine learning tools. Furthermore, both genome and metabolome mining are highly evolving fields that could benefit from integration; examples for peptide-based specialized molecules were mentioned; however, for most classes novel tools are needed. Exciting ideas to apply machine learning to combine specific structural information from genome and metabolome mining were also shared.

4.4 Dark Matter

Ricardo Da Silva (UC – San Diego, US)

Dark matter was defined as frequently observed spectral signals for which no annotation can be assigned. The first source of dark matter was attributed to experimental design, highlighting the importance of having controls, in order to differentiate real signal from noise and contamination. The second source was attributed to spectrometric data pre-processing, due to signal complexity (adducts, isotopes, fragments, contaminants) and also due to the low accuracy of existing tools (missing peaks, integration of noise, split peaks, merged peaks). The third source was attributed to the limited coverage of spectral libraries, data repositories, specially for benchmark datasets, and the lack of connection between the known chemicals and its biological source. The most important long term solutions cited were the expansion of reference and raw data databases, as well as the design and analysis of benchmark datasets by multiple orthogonal methods and multiple labs.

4.5 Retention time and retention time prediction

Michael Witting (Helmholtz Zentrum – München, DE)

Retention time (RT) represents an orthogonal information to mass spectrometry. It is especially important to distinguish isomers which cannot be separated solely by MS and MS/MS. In this beak-out group several small presentations showed the state-of-the art in reporting RT data and predicting it. Different ways of improving RT prediction were discussed and important parameters to model retention time order were collected. These will serve as future reference for collecting metadata and data around RT and its prediction.

4.6 Data independent acquisition

Corey Broeckling (Colorado State University – Fort Collins, US)

Several members discussed the increasingly common 'data independent acquisition' (DIA) of MS/MS data. A workflow was presented for processing 'all ion fragmentation' data, an acquisition approach in which no precursor selection is performed. The various flavors of data independent acquisition methods that have been published were summarized graphically, in an effort to inform developers of DIA processing workflows to design processing tools which are sufficiently versatile to handle the many iterations. A considerable effort is being made to update and expand the ramclustR package with novel clustering methods, and a novel R based workflow was presented demonstrating efficacy in using extracted ion chromatograms to remove contaminating signals from DIA spectra. More broadly, the attendees discussed the many strengths of DIA approaches, as well as the frequent tradeoff between sensitivity and selectivity, and the difference between actual (physical isolation defined by precursor isolation window size) and processing assisted selectivity (overlapping windows can enable contaminant signal subtraction).

4.7 Molecular networking and integration with annotation tools

Madeleine Ernst (UC – San Diego, US)

Mass spectral molecular networks enable the observation of similarities and differences in MS/MS fragmentation patterns of complex samples. MS/MS fragmentation patterns are typical of a molecular structure, and molecular structures can thus be identified/annotated manually or by using *in silico* approaches. This break-out session discussed ideas and challenges to integrate molecular networks with *in silico* annotation tools. Mass spectral molecular networking has been made widely accessible by Global Natural Products Social Molecular Networking (GNPS), allowing the community to share, analyze and annotate MS/MS data. Nevertheless, mass spectral molecular networking is currently a per-study approach and comparison of different datasets is only possible with limitations. Major challenges discussed in this session ranged from questions on how to integrate information on mass shifts in an automated way, simulate enzymatic reactions to predict molecular structures and incorporate biochemical properties (e.g., pH, bioactivity) in the networks to using network topology to improve *in silico* annotation. The need for sharing data with the community and making datasets public was stressed as well as the aim of integrating all types of information into the mass spectral molecular networks, ultimately enabling inter-study comparisons.

4.8 Bridging the gap

Sarah Scharfenberg (IPB – Halle, DE)

Metabolomics science faces several gaps, such as language and communication problems between the tool developers and the experimentalist, insufficient visibility of available tools and inappropriate usability of tools. Collaborative projects will fail as soon as one of the involved parties assesses its part as a service. Early communication of study design should be mandatory for each experimentalist who wants statistical support. To match the needs of both sides, we should further go for user driven development, which is based on a real study and a fixed problem and accompanied and constantly feedbacked by the corresponding experimentalist. To improve the usage of existing tools we could provide more educational documentation, such as webinars, video tutorials and example datasets.

4.9 Creating the "perfect" benchmark / reference dataset

Corey Broeckling (Colorado State University – Fort Collins, US)

The "perfect benchmark dataset" discussion began by trying to define the computational problems in data processing, including being able distinguish signal from noise and metabolites from artifacts. The concept of theoretical data and what sort of artifacts we might be able to predict/model was discussed, in an effort to determine how realistic the predicted data would be. A small "ring trial" was proposed, with a sample set based on the notion of a sample set containing a moderately complex (20 - 100) set of pure authentic standard compounds. This compound mixture will be analyzed under realistic conditions as if it were an authentic sample set. The sample mixture could additionally be sent to interested labs for analysis. Computational collaborators would be delivered data from one or more laboratories rather than samples. Participants intend to move the concept forward with a more detailed planning document.

4.10 Integrating spatial and conventional metabolomics

Andrew Palmer (EMBL – Heidelberg, DE)

The computational communities that focus on traditional metabolomics and imaging mass spectrometry have both approached the large scale processing of mass spectra from different perspectives. This session was an opportunity for both communities to describe their experiments, data, and processing strategies to seek opportunities for the exchange of ideas and methodologies. After all participants had an opportunity to present their methodologies, discussion focussed on the similarities between imaging mass spectrometry and high throughput flow injection experiments: both in terms of the experimental aims (to maximally annotate the peaks present) and data (large numbers of high resolution mass spectra from complex mixtures). To begin to assess the efficacy of computational strategies across these modalities a source of well characterised publically available data from flow injection studies needs to be identified.

4.11 *Ab initio* network reconstruction challenges

Fabien Jourdan (INRA-ENVT – Toulouse, FR)

First part of the discussion was about sharing views on network research field related to metabolomics. In fact, "network" is a very versatile concept used for many applications from molecular networks (nodes represent fragmentation spectra and edges cosine score) to biochemical networks (nodes are compounds and edges correspond to metabolic reactions). It is thus of utmost importance to define carefully what nodes and edges are modelling. Discussion then focused on the lack of identifier standardisation between modelling and metabolomics community. In particular, as a community, we advice that more care should not be taken when curated metabolic networks in order to provide InChIKeys and InChIs. The other issue, related to other break-out session, is the necessity of providing identifiers with flexible substructures identifiers (e.g. to deal with class of lipids). Finally, reconstruction from peak lists was discussed (ab initio reconstruction). This approach is based on mass shifts between masses in a peak list. If the mass shift corresponds to a biochemical transformation mass difference (e.g. methylation) then an edge is added. This definition implies the presence of a lot of false positive edges. Discussion then focused on ways to automatically filter out these edges.

4.12 Version control and CI for MS/MS libraries

Steffen Neumann (IPB – Halle, DE)

The Spectral Libraries session dealt with spectral libraries, such as GNPS, MassBank or HMDB. We discussed several routes to get spectra into libraries, ranging from conversion of existing (in-house) libraries to automated workflows extracting clean spectra from raw data. The MassBank team presented an approach to use established processes from software engineering, in particular version control and continuous testing. MassBank is moving towards a github-supported workflow in the near future, and first prototypes were shown. Such setups also simplify large-scale automatic curation, for structures, additional metadata and links. Several participants have done curation and processing of libraries, and these could then be fed back to the repositories.

4.13 False discovery rates

Marcus Ludwig (Universität Jena, DE)

It is understood that false discovery rate estimation is a necessity to allow comprehensive statistical analysis of structural identifications from mass spectrometry data. Currently no computational method provides a score to sufficiently separate true and bogus hits. Some strategies already applied in proteomics might help, such as fitting score distributions and estimating p-values. We discussed whether improvements on the measurement side might provide better data to discriminate between different candidates. However, it is unclear if mass spectrometry data provides enough information to distinguish highly similar metabolites. Due to metabolites large structural diversity and insufficient information we might need to reformulate what we can in fact deduce from the data and what we accept as a “correct“ hit.

4.14 Feature prioritization

Sarah Scharfenberg (IPB – Halle, DE)

Feature prioritization is one of the main steps in a biological study although it is sparsely addressed in general MS1 workflows. Reducing the number of correlated variables in advance also positively affects the outcome of the multivariate analysis. Strongly dependent on the study goal there is a set of commonly used methods, such as PCA, PLS, variable clustering, fold changes, hypothesis testing, random forests, or a combination of these. Based on MSMS networks or visual models it is possible to connect feature intensities to bioactivities or regions. A proper experiment design enables strict criteria on how to select the relevant features, e.g. the most differentially expressed features.

Participants

- Hayley Abbiss
Murdoch University, AU
- Theodore Alexandrov
EMBL Heidelberg, DE
- Manor Askenazi
Biomedical Hosting – Arlington, US
- Eric Bach
Aalto University, FI
- Ruth Birner-Grünberger
Universität Graz, AT
- Sebastian Böcker
Universität Jena, DE
- Corey Broeckling
Colorado State University – Fort Collins, US
- Christoph Büschl
Universität für Bodenkultur – Wien, AT
- Ricardo Da Silva
University of California – San Diego, US
- Pieter Dorrestein
University of California – San Diego, US
- Edward M. Driggers
General Metabolics – Wilchester, US
- Kai Dührkop
Universität Jena, DE
- Madeleine Ernst
University of California – San Diego, US
- P. Lee Ferguson
Duke University – Durham, US
- Nils Hoffmann
ISAS – Dortmund, DE
- Julijana Ivanisevic
University of Lausanne, CH
- Fabien Jourdan
INRA-ENVT – Toulouse, FR
- Alexander Kerner
Lablicate GmbH – Hamburg, DE
- Oliver Kohlbacher
Universität Tübingen, DE
- Martin Krauss
UFZ – Leipzig, DE
- Martin Loos
looscomputing – Dübendorf, CH
- Marcus Ludwig
Universität Jena, DE
- Marnix Medema
Wageningen University, NL
- Kris Morreel
Ghent University, BE
- Rolf Müller
Helmholtz-Institut – Saarbrücken, DE
- Steffen Neumann
IPB – Halle, DE
- Louis-Felix Nothias-Scaglia
University of California – San Diego, US
- Andrew Palmer
EMBL – Heidelberg, DE
- Alan Race
Universität Bayreuth, DE
- Stacey Reinke
Murdoch University, AU
- Simon Rogers
University of Glasgow, GB
- Juho Rousu
Aalto University, FI
- Sarah Scharfenberg
IPB – Halle, DE
- Jennifer Schollee
Eawag – Dübendorf, CH
- Emma Schymanski
University of Luxembourg, LU
- Jan Stanstrup
University of Copenhagen, DK
- Michael Andrej Stravs
Eawag – Dübendorf, CH
- Raf Van de Plas
TU Delft, NL
- Justin van der Hooft
Wageningen University, NL
- Kirill Veselkov
Imperial College London, GB
- Ron Wehrens
Wageningen University, NL
- David Wishart
University of Alberta – Edmonton, CA
- Michael Anton Witting
Helmholtz Zentrum – München, DE
- Nicola Zamboni
ETH Zürich, CH

Report from Dagstuhl Seminar 17492

Multi-Level Modelling

Edited by

João Paulo A. Almeida[1], Ulrich Frank[2], and Thomas Kühne[3]

1 **Federal University of Espírito Santo – Vitória, BR,** jpalmeida@ieee.org
2 **Universität Duisburg-Essen, DE,** ulrich.frank@uni-due.de
3 **Victoria University of Wellington, NZ,** thomas.kuehne@ecs.vuw.ac.nz

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17492 "Multi-Level Modelling". This seminar brought together researchers and industry practitioners from the fields of conceptual modeling, ontologies, and formal foundations to discuss and share the benefits of Multi-Level Modelling (MLM), to develop an agreement on MLM terminology and scope, and to drive future research directions in MLM. Some foundational presentations were given by the seminar organizers to ground the discussions and provide an initial set of open questions which would lead to the formation of the working groups. In addition, six industry representatives gave talks explaining the needs, challenges, utility, and possible issues with adoption of MLM in industry. Based on the original seminar goals, the talks, and the resulting discussions, four working groups were established to investigate: the formal and ontological "Foundations" of MLM; promising "Applications" and potential evaluation criteria for MLM methods; the "Dynamic Aspects" of MLM, such as processes and behaviour; and, the use of and impact on "Model Transformations" in the context of MLM.

Seminar December 3–8, 2017 – http://www.dagstuhl.de/17492
1998 ACM Subject Classification D.2 Software Engineering, D.2.13 Domain engineering
Keywords and phrases metamodeling, multi-level modeling
Digital Object Identifier 10.4230/DagRep.7.12.18
Edited in cooperation with Matt Selway

1 Executive Summary

João Paulo A. Almeida (Federal University of Espírito Santo – Vitória, BR)
Colin Atkinson
Ulrich Frank (Universität Duisburg-Essen, DE)
Thomas Kühne (Victoria University of Wellington, NZ)

Multi-Level Modeling (MLM), i.e., the explicit exploitation of multiple levels of classification when modeling, represents a significant extension to the traditional two-level object-oriented paradigm with the potential to dramatically improve upon the utility, reliability and complexity level of models. It therefore has benefits in many domains of modeling such as software engineering, process modeling and enterprise modeling. Research into multi-level modeling has increased significantly over the last few years, manifesting itself in lively debates in the literature, four international workshops (MULTI 2014–2017), a published journal theme issue (SoSyM), a special issue for the EMISA journal (in preparation), and a growing number of tools and languages. While the enthusiasm around MLM provides momentum to this promising research area, the recent speed of growth and the focus on exploring new language features has raised some challenges, including the following:

growing diversity of approaches On the one hand diversity is welcome in order to spawn a competition of ideas but on the other hand it can slow down the approach's growth and industry adoption unless steps are taken to consolidate ideas and bundle resources.

lack of integration with related disciplines In particular, ontology engineering has overlapping application areas and could form a powerful synergy with MLM due to its complementary strengths and weaknesses. Moreover, areas such as logic, philosophy, and linguistics are also highly relevant for the further advancement of MLM.

neglect of real world applications It is natural to initially focus on core principles when developing a new approach, but at some point it becomes important to make the transition into industrial practice in order to validate claims about the utility and need for MLM, and to promote the uptake of MLM in new domains and industries.

Goals

In order to address the aforementioned challenges the seminar brought together researchers and industry practitioners from the areas of conceptual modeling, ontologies, and formal foundations. In particular, to further the coherence of future research into multi-level modeling, we aimed at

- having some consolidating discussion on terminology and scope;
- strengthening (formal) foundations; and
- identifying objective criteria for comparing competing approaches, e.g., by developing respective benchmarks in cooperation with modelers from industry.

Working Groups

A talk on "What is Multi-Level Modeling?" by Thomas Kühne (cf. talk abstract) set the stage for the terminology discussion and presented results from a survey that the organizers ran prior to the seminar. The survey results were in good agreement with the ideas the talk put forward in terms of what core multi-level modeling concepts are, such as "multiple levels of abstraction", "classification as the core abstraction principle", "modeling the real world" (as opposed to engineering languages). The survey design carefully avoided introducing bias, hence there were no multiple-choice questions on subjects like this one. The lack of answer standardization required a manual allocation into answer categories such as the aforementioned ones, but only clear cut cases were counted. Overall, there was very little controversy over what multi-level modeling constitutes. Furthermore, the survey nicely confirmed that the initial seminar goals were congruent with what most participants found to be interesting and important work in the area of multi-level modeling.

Two further talks were aimed at supporting the formation of working groups:

1. A talk on foundations and ontologies by João Paulo A. Almeida contrasted ontology engineering to language engineering, asked which questions should be addressed by a foundation, and explored some answers.
2. A talk on applications by Ulrich Frank elaborated on challenges for multi-level modeling in practical applications.

Subsequently four working groups were established by identifying the themes that both aligned with the original workshop goals and garnered the highest interest among participants.

A working group on "Foundations" formed and decided to start on investigating which ontological commitments and metaphysical choices may be required or useful for a foundation of multi-level modeling (cf. "Foundations" group report, 4.1).

A working group on "Applications" set out to identify promising application domains for MLM, find common properties for such application domains, identify anticipated benefits of MLM, and determine evaluation criteria for MLM methods (cf. "Applications" group report, 4.2).

A further working group on "Dynamic Aspects" focused on a sub-area of enterprise modeling, i.e., modeling process-related and/or dynamic behavior aspects (cf. "Dynamic Aspects" group report, 4.3).

We strove to both address the originally planned goals of the seminar but also to allow new goals to be formed, based on the final composition of the participants. As a result, the group formation process yielded one more group that focused on transformations in the context of multi-level modeling (cf. "Transformations" group report, 4.4).

Due to the overlap between foundation work and the area of "integration with ontologies", a group dedicated exclusively to exploring synergies between MLM and ontology engineering did not emerge. We are hopeful that the working group on "Foundations" will explore more of the synergy aspect in future collaborations.

Industry Focus

As closing the gap between academia and industry with respect to multi-level modeling was a primary goal of the seminar, we had a total of six talks by industry representatives. These talks gave the speakers an opportunity to explain actual needs, set challenges, comment on utility, etc., plus allowed the audience to inquire about hurdles for adoption, etc. Please see the industry talk abstracts included in this report for further details.

We, the organizers, are extremely grateful to the staff of Dagstuhl for providing a perfect seminar venue and to the participants who not only made this seminar a success but also provided a wealth of generous positive feedback.

The organizers, Dagstuhl, 2018
João Paulo A. Almeida
Colin Atkinson
Ulrich Frank
Thomas Kühne

2 Table of Contents

3 Overview of Talks

3.1 What is Multi-Level Modeling?

Thomas Kühne (Victoria University of Wellington, NZ)

Main reference Colin Atkinson, Thomas Kühne: "The Essence of Multilevel Metamodeling", in Proc. of the «UML» 2001 - The Unified Modeling Language, Modeling Languages, Concepts, and Tools, 4th International Conference, Toronto, Canada, October 1-5, 2001, Proceedings, Lecture Notes in Computer Science, Vol. 2185, pp. 19–33, Springer, 2001.
URL http://dx.doi.org/10.1007/3-540-45441-1_3

A community will struggle to coherently grow unless it has a shared sense of which core concepts define its discipline. The purpose of this talk was therefore to set the stage for allowing the community to form a consensus about what the scope of multi-level modelling (MLM) is and which aspects should be considered necessary versus those that either form sub-disciplines or are better thought of being outside the intended scope.

To this end, I first provided an account of the history of MLM, in particular concerning how MLM differs from prior work on language-oriented metamodeling, i.e., the use of multiple linguistic type models. I argued that the use of domain-induced metahierarchies [1] implies a modeller focus in contrast to the language-engineering perspective of prior metamodeling work.

I then suggested a list of key characteristics that could be considered to be part of every MLM approach. I contrasted these with existing choices that may or may not be considered to also fall under the MLM umbrella. I briefly looked at some dimensions of variability that are likely to give rise to MLM variants and referenced work that aimed to explore ways to systematically evaluate and compare such variants [2, 3].

Finally, I motivated the four core topics of the seminar – Foundations, Ontologies, Applications, and Evaluations by pointing out their role in providing a sound basis for MLM, furthering MLM by integrating it with other disciplines, and demonstrating MLM's relevance with respect to two-level technology and industry applications.

I concluded the talk with an analysis of the responses we received to a survey we ran prior to the seminar.

References

1 C. Atkinson and T. Kühne. The essense of multilevel metamodeling. In M. Gogolla and C. Kobryn, editors, *Proceedings of the 4th International Conference on the UML 2000*, volume 2185 of *LNCS*, pages 19–33, Toronto, Canada, October 2001. Springer Verlag.

2 C. Atkinson, R. Gerbig, and T. Kühne. Comparing multi-level modeling approaches. In *Proceedings of the 1st International Workshop on Multi-Level Modelling*, volume 1286 of *CEUR Workshop Proceedings*, pages 43–52. CEUR, 2014.

3 C. Atkinson and T. Kühne. On evaluating multi-level modeling. In *Proceedings of the 4th International Workshop on Multi-Level Modelling*, volume 2019 of *CEUR Workshop Proceedings*, pages 274–277. CEUR, 2017.

3.2 What Kind of Foundations do we Need for Multi-Level Modeling? (in Language & Ontology Engineering)

João Paulo A. Almeida (Federal University of Espírito Santo – Vitória, BR)

In this introductory talk, I have identified some key challenges to be addressed in a foundation for multi-level modeling. First and foremost, I have tried to argue that it is paramount for multi-level modeling as a discipline to investigate the guiding notion of "level". While this seems to be a trivial observation, different multi-level modeling approaches structure models differently, sometimes using the same term "level" for different underlying organization principles. Thus, "level" talk is, in many cases, semantically overloaded, with "levels" serving different purposes. A foundation for multi-level modeling should identify the "job" that a notion of "level" is put to. Foundations for multi-level modeling should allow us to understand what are the options for the nature of the various organizing principles and clarify the relation between terms such as "levels", "potencies", "orders", "stages", "layers", "strata", etc.

I have also identified some desiderata for a suitable foundation, discussing that it should ideally:

- Encompass different approaches, e.g., two-level, level-agnostic/blind, strictly stratified, power-type based;
- Serve to settle semantic questions;
- Justify modeling choices or explain qualitative differences between modeling approaches;
- Account for shallow and deep instantiation and other multi-level modeling mechanisms;
- Account for language engineering and ontology engineering based on general principles (since language engineering and ontology engineering have employed similar object-oriented representation schemes).

In order to provoke discussions on foundations, the following questions were posed:

1. What is the nature of a (multi-level) model?
2. What is the nature of a "level"? (What principles arise from organization into "levels"?)
3. What does it mean for an entity to be in a "level"? (What is the job a "level" does?)
4. What is the nature of entities in a "level"?
5. What is the nature of relations between entities? (Within a "level" and across "levels")

I have concluded that a suitable foundation for multi-level modeling would constitute an ontology of multi-level phenomena, and that failing to make this ontology explicit results in adopting a poor ontology inadvertently.

3.3 On the Application of Multi-Level Modelling – Prospects and Challenges

Ulrich Frank (Universität Duisburg-Essen, DE)

It is undisputed that conceptual modelling is a prerequisite of designing large software systems. However, the development and use of traditional modelling languages are compromised by serious problems. First, there are no convincing criteria to clearly decide whether a certain concept should be part of language or should rather be defined with the language. However, the traditional paradigm requires corresponding decisions to be made. Second, the design of modelling languages has to cope with an inherent conflict. On the one hand, a DSML should include concepts that address the specific needs of a domain. Hence, it should aim at modelling productivity. On the other hand, economies of scale demand for less specific concepts in order to reach a wide scale of reuse. Third, there are concepts that make perfect sense to us, and that would foster reuse and integration, which cannot be represented in the traditional paradigm. Fourth, model-driven software development creates a notorious problem, that is, the synchronization of models and code. In this presentation, it is shown that multi-level modelling in general, the FMMLx and the Xmodeler in particular are suited to overcome the limitations of the traditional paradigm. Furthermore, it is demonstrated that multi-level modelling and programming enables self-referential systems, which clearly contribute to more flexible systems and promote user empowerment.

3.4 Implications When Migrating to Multi-Level Modeling (Industry Perspective)

Ta'id Holmes (Deutsche Telekom – Darmstadt, DE)

Modeling enables different stakeholders to participate in an engineering process. Therefore modeling is (also) about roles, responsibilities, and collaboration in particular. What are the effects and consequences when a traditional two-level modeling approach (i.e., comprising a metamodel and models) is being transformed to a multi-level modeling approach where part of the language engineering is delegated to a domain expert? What makes a model "resilient" in regard to evolution? Which anti-patterns in modeling can be effectively avoided (using tool-support)? Which best-practices can be incentivized? Is it possible to prepare for the worst-case usage? How would models look if language features were abused and could an organization live with the (long-term) consequences? When is standardization indicated (e.g., the higher model elements are located in a multi-level model the higher potential reuse, the higher the need for standardization)?

3.5 Potential of Multi-Level Modelling in Model Based Systems Engineering: A "National Research Lab" Perspective (Industry Perspective)

Philipp Martin Fischer (DLR – Braunschweig, DE)

The lifecycle of a spacecraft follows a process of phases. There are important goals between these phases such as a preliminary design review (PDR). These goals have to be successfully passed by the whole project team. Usually they are connected to contractual conditions, e.g. after the PDR the design is usually settled and the spacecraft will be built. Such contracts consider agreements with industry and suppliers on part and equipment orders, etc. [1, 2]

Changing the design after the PDR may require contractual changes. These changes tend to be expensive and have to be avoided. [3]

To avoid the aforementioned issues, model based systems engineering (MBSE) has been introduced in satellite design. It is focusing on data bases that provide a conceptual data model (CDM also known as meta model). The engineers start modeling the system within such data bases usually on a functional level. The system model is then used as central source of knowledge for further processes. [3] Such processes may cover on the fly analysis [4], configuration of simulators [5], new ways of modelling including interactive visualization [6] and emerging trends such as mixed reality. This notion of an MBSE approach has been implemented e.g. in DLR's data base called Virtual Satellite. Until today it is successfully applied in early spacecraft design. [7] Nowadays, the data base is applied beyond the early phases. Therefore it has to deal with more detailed information and has to cope with yet unknown requirements of tomorrow. Accordingly the CDM has become more complex and offers extension mechanisms. [3]

The success of these data bases is partly founded in their configuration control capabilities. In fact, the engineers require not just one model of the spacecraft but several. These models are a master model, derived simulator models, or models for the actual satellites one and two. The different models are needed to reflect differences e.g. a different electrical harness for the simulator, or individual command ids for individual satellites. [3, 5] This is handled by the product structures of the data bases. They are standardized to a certain extend in the activities of European Ground Segment – Common Core (EGS-CC) as well as the European Cooperation for Space Standardization Technical Memorandum (ECSS-E-TM) 10-23. [8, 9] These structures are used for a hierarchical decomposition of the system. The first tree modelled, is usually a product tree. The engineers are using this tree for an initial description of the required parts such as a reaction wheel (RW) or magnetic-torquer (MTQ). They don't yet model every instance of such a part. Instead, they are modelling them as an idea of a type. The second tree is the configuration tree where these types are instantiated into a virtual configuration of several RWs and MTQs. The final trees are the assembly trees, which are based on the configuration trees but representing how actual satellites are build. They reflect the assembly e.g. of an actual satellite number one and two. Since all trees, including their parts, are based on each other, defining the mass of the RW once in the product tree will update all its further instances in the configuration and assemblies as well. Override functionality allows changing the values when needed or to combine the information with so called realizations. These realizations represent the actual ordered parts that have been delivered. E.g. their calibrations are measured and the best fitting parts are now assigned to the assembly. The information such as a mass is modelled by so called engineering categories. They are based on what is known as type/object pattern. [10, 11] An engineering category defines a property such as a mass and assigning this category to an element in the product tree instantiates it. Now information can be stored in the instance of this property. [3]

Even though successful, there are some drawbacks, where Multi-Level Modeling promises some reasonable improvements. For example, an engineering category for storing geometric information of a part consists of a position, a size and a shape. These three properties make sense at configuration level, but not yet at product level. At product level or type level the position is simply not yet known. By today this is handled by either providing arbitrary values or by complex class inheritance hierarchies. Nevertheless such handling is a workaround rather than a proper solution to such problems.

Multi-Level Modelling and the idea of potencies and deep instantiation [12] in particular, seems to offer a solution. Assigning a potency of two for the position property creates awareness of this property already at product level. The actual value of that property can now be set starting from configuration level. Considering another example based on an engineering category for tele-commands, it consists of a purpose, e.g. RW turn on, an equipment identifier as well as a satellite identifier. With the concept of potency and deep instantiation, engineers are aware of all three properties already at product level. The actual necessary information needs to be provided at the stages of configuration and assembly.

This approach works well with the accepted set of product structures. Nevertheless current work indicates that there might be further trees needed in future applications. Introducing a new integration tree in between configuration and assembly breaks the potency mechanism for the tele-command example. A decrease of that potency with every level of instantiation is not suitable. A fix to this issue could lead in the direction of context aware potencies.

At the moment, the MBSE data bases do not yet apply such Multi-Level Modelling. Nevertheless, it can be seen that certain directions of Multi-Level Modelling could improve the overall modeling activities. For sure this view is a highly practical driven adoption of the theories of Multi-Level Modelling and potentially breaks some of the clear cut semantics. Still it shows that these theories provide some answers to the problems of spacecraft related models of today. In general, the described idea of applying potencies and deep instantiation to the concept of engineering categories looks promising. In order to prove its applicability, further research is needed and some first prototypes are envisaged for evaluation.

References

1 ECSS Secretariat. ECSS-M-ST-10C Space project management – Project planning and implementation. ESA-ESTEC Requirements & Standards Division, Noordwijk, Netherlands, 2009.

2 NASA. Systems Engineering Handbook (NASA/SP-2007-6105 Rev 1). National Aeronautics and Space Administration, Washington, D.C. / USA, 2007.

3 P. M. Fischer, D. Lüdtke, C. Lange, F.-C. Roshani, F. Dannemann and A. Gerndt. Implementing model based system engineering for the whole lifecycle of a spacecraft. CEAS Space Journal, 12 July 2017.

4 M. Deshmukh, V. Schaus, P. Fischer, D. Quantius, V. Maiwald and A. Gerndt. Decision Support Tool for Concurrent Engineering in Space Mission Design. In *Proceedings of the 19th ISPE International Conference on Concurrent Engineering*, pages 497–508. Springer London, 2013.

5 P. M. Fischer, H. Eisenmann and J. Fuchs. Functional Verification by Simulation based on Preliminary System Design Data. In *Proceedings of the 6th International Conference on Systems & Concurrent Engineering for Space Applications (SECESA)*, Stuttgart, Germany, 2014.

6 M. Deshmukh, R. Wolff, P. M. Fischer, M. Flatken and A. Gerndt. Interactive 3D Visualization to Support Concurrent Engineering in the Early Space Mission Design Phase. In *Proceedings of the 5th CEAS Air and Space Conference*, Delft, Netherlands, 2015.

7 P. M. Fischer, M. Deshmukh, V. Maiwald, D. Quantius, A. Martelo Gomez and A. Gerndt. Conceptual Data Model – A Foundation for Successful Concurrent Engineering. In *Concurrent Engineering: Research and Applications*, 2017.

8 M. Pecchioli, A. Walsh, J. M. Carranza, R. Blommestijn, M.-C. Charmeau, M. Geyer, C. Stangl, P. Parmentier, H. Eisenmann, J. Rueting, P. Athmann, W. Bothmer, I. Krakowski, P.-Y. Schmerber, F. Chatte, P. Chiroli and M. Poletti. Objectives and Concepts of the European Ground Systems Common Core (EGS-CC). In *Simulation & EGSE for Space Programmes*, Noordiwjk, Netherlands, 2012.

9 ECSS Secretariat. ECSS-E-TM-10-23A – Space engineering – Space system data repository. ESA-ESTEC Requirements & Standards Devision, Noordwijk, Netherlands, 2011.

10 F. D. Lyardet. The Dynamic Template Pattern. In *Proceedings of the 4th Pattern Languages of Programming Conference*, Monticello, Illinois, USA, 1997.

11 R. Johnson and B. Woolf. Type object. In *Pattern languages of program design 3*, pages 47–65, Bosten, MA, USA, 1997. Addison-Wesley Longman Publishing Co., Inc.

12 C. Atkinson and T. Kühne. Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359, July 2008. Springer Verlag.

3.6 Personal View on Multi-Level Modeling (Industry Perspective)

Vinay Kulkarni (Tata Consultancy Services – Pune, IN)

Coming from the industry, I look at Multi Level Modeling (MLM) through a composite lens comprising of need, effectiveness and robustness. I was involved in development of model driven engineering technology for automating development of business applications. Several large ($>$ 10 MLOC) applications are developed using this technology over past 20+ years. MLM was not required barring one specific use-case in user interface modeling – the 3-levels i.e., meta-meta, meta and user model, sufficed. My present research is on adaptive resilient enterprises that calls for modeling languages and associated technology to specify predictive, prescriptive and descriptive aspects of modern enterprises. Here too, I am unable to justify MLM as the most appropriate modeling approach. I do see value of MLM for conceptual space, but, that does not seem to trickle down to design and implementation spaces.

3.7 Commercial Introduction – BORO Solutions (Industry Perspective)

Chris Partridge (Brunel University, GB)

This presentation introduces BORO Solutions. It briefly introduces the company and its main products. It provides an overview of the work done in a recent project. It outlines the current research being done in one major area – a general theory of multi-levelling to include mereology as well as classification and generalisation.

3.8 Multi-Level Modelling at BiZZdesign (Demo)

Maarten Steen (BiZZdesign – Enschede, NL)

BiZZdesign is a fast growing global software company that helps its customers to manage change in their organization. BiZZdesign Enterprise Studio is an advanced, graphical modelling and analysis platform for a wide variety of enterprise, business and architecture modelling techniques, such as ArchiMate, BPMN, DMN and UML. All these modelling techniques:

- Are graphical DSLs.
- Based on international standards.
- Support a specific domain or discipline.
- Can be combined and cross-reference each other.
- Adhere to a meta-model.
- Which can be customized.

MLM Challenges – a tool vendor's perspective

At the workshop I presented the multi-level modelling-related challenges that we face as a modelling tool vendor, as well as some of our pragmatic solutions. Below a summary of our MLM challenges:

- Many customers want to customize the standards-based metamodels we provide out of the box.
- Often there is a need for adorning concepts with additional properties/attributes that are specific to a model, view or analysis, but if we would add these properties to the metamodel they would appear in all models and views.
- Users sometimes want to dynamically change the type of an object or relation, but this may break constraints imposed by the metamodel.
- When the metamodel is changed, existing models may not satisfy that metamodel anymore and have to be migrated, which is a non-trivial task.
- Whenever BiZZdesign publishes a new base metamodel, customers need to merge their own metamodel customizations done on an earlier version, but this may lead to difficult to resolve merge conflicts.
- When we import models from other tools, we also need to import or map their metamodel (customizations). We currently cannot do this on the fly.
- Metamodel-model co-evolution is manageable in a single repository, but very hard in a distributed version management system such as employed by our team server.

3.9 The MLM Application of FOM (Demo)

Mira Balaban

The MLM theory of FOML is built around the notion of level, which is a class model. This approach enables reuse of existing class model tools for correctness checking, problem identification, repair and optimization. FOML provides an MLM application that is based on level specification, with intra-level and inter-level constraints and inference rules, model query and computation. The application exploits essential features of FOML that enable unrestricted chains of instantiation and subtyping.

FOML is a path-oriented executable logic-programming language. It includes navigation structures and polymorphic typing, and can support user-defined computation and reasoning. FOML and its MLM application are available on SourceForge.

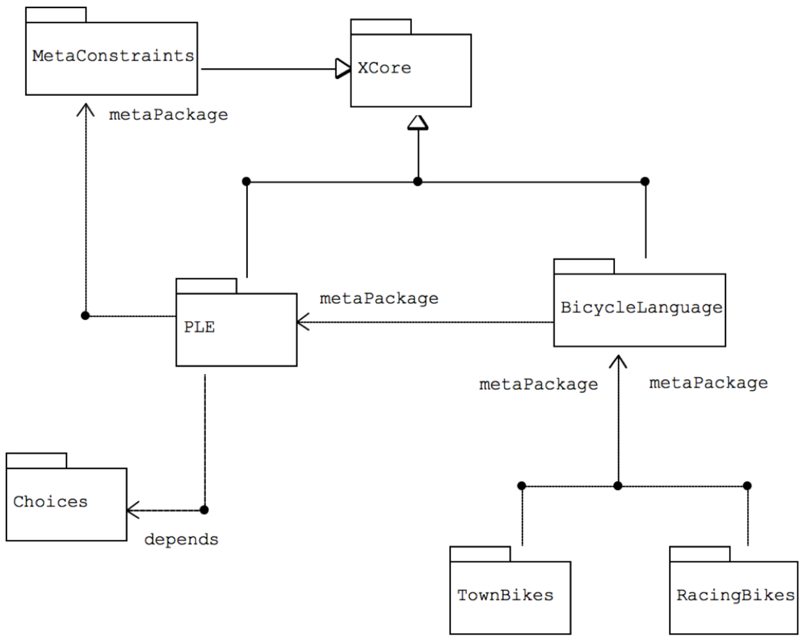


Figure 1 Language definitions for product configuration.

3.10 Multi-Level Modelling in XModeler (Demo)

Tony Clark (Sheffield Hallam University, GB) and Ulrich Frank (Universität Duisburg-Essen, DE)

Joint work of Tony Clark, Ulrich Frank
Main reference Tony Clark, Paul Sammut, James S. Willans: "Applied Metamodelling: A Foundation for Language Driven Development (Third Edition)", CoRR, Vol. abs/1505.00149, 2015.
URL http://arxiv.org/abs/1505.00149

Multi-level modelling occurs naturally at all levels of system design, especially that which takes a *Language Engineering* approach to Software Engineering. Such an approach involves the definition of languages (domain-specific or general-purpose) that are to be used within the definition of a system. Axiomatically, this approach leads to multiple levels since the languages must be defined in a meta-language and the languages are instantiated to produce particular occurrences which can be considered to exist at different phases of the system development such as 'specification', 'design' or 'run-time'. Furthermore, the approach is greatly enhanced if the levels can be integrated or, ideally, if the levels can be freely mixed (providing level information is clear when required). For example, any type of software tool, typically manages representations of the program definition and the run-time of the program at the same time: the design of such a system will naturally involve concepts that cross level boundaries. Generalising this example to a Language Engineering approach means that the design of *any* system will involve multiple levels that usefully cross boundaries as the use of abstraction leads the designer to want to express information that relates to particular instances at higher and higher the levels.

This talk provided an overview of the XModeler toolkit that has been designed to support a Language Engineering approach to system design. It is based on a small core meta-model that is meta-circular and thereby allows any number of instantiation levels, including new meta-languages. The demonstration showed how XModeler can be used to define a number of languages relating to product configuration, each language supporting a different aspect of the solution or problem domain. For example a language is defined to represent choices within configurations. Another language is used to define constraints (in the style of OCL) that are applied over a number of type boundaries (unlike OCL). The demonstration showed that the XModeler defined languages can be used to define bicycle product-families (racing bikes, touring bikes, domestic bikes) leading to instances that are product models whose instances must conform to constraints defined on the product-family (or higher) levels. The demonstration clearly shows the utility of the Language Engineering approach, the utility of multiple-levels in modelling, and the utility of crossing type boundaries.

3.11 The ML2 Multi-Level Modeling Language (Demo)

João Paulo A. Almeida (Federal University of Espírito Santo – Vitória, BR)

Joint work of João Paulo A. Almeida, Claudenir M. Fonseca, Victorio A. Carvalho

In this demonstration, the ML2 multi-level modeling language and accompanying Eclipse plugin were presented. ML2 is a textual modeling language built in conformance with the MLT* multi-level modeling theory [1]. It aims to address a comprehensive set of requirements for multi-level modeling. It was developed in the scope of the M.Sc. thesis of Claudenir M. Fonseca [2], in collaboration with João Paulo A. Almeida and Victorio A. de Carvalho. The proposed language is supported by a featured Eclipse-based workbench which verifies adherence of the ML2 model to the MLT* rules. The capabilities of ML2 were demonstrated in different modeling tasks involving multi-level domains. The language supports chains of instantiation of arbitrary sizes, supports the notion of type "order" to guide the construction of sound models, and also supports what is known as "orderless" type – a feature which enables expressiveness of scenarios that defy stratification of a model into "orders". It further supports the so-called cross-level relations in order to address variations of the powertype pattern, and supports the notion of "regularity attributes" to flexibly address the relations between attributes of types at different orders. It was shown that the tool is capable of detecting a number of modeling problems by enforcing ML2 syntactic rules.

References

1 João Paulo A. Almeida, Claudenir M. Fonseca, Victorio A. Carvalho. *A Comprehensive Formal Theory for Multi-Level Conceptual Modeling*. Conceptual Modeling. ER 2017. Lecture Notes in Computer Science, vol. 10650, Springer, 2017
2 Claudenir M. Fonseca. *ML2: An Expressive Multi-Level Conceptual Modeling Language.* M.Sc. Thesis, Federal University of Espírito Santo, Brazil, 2017

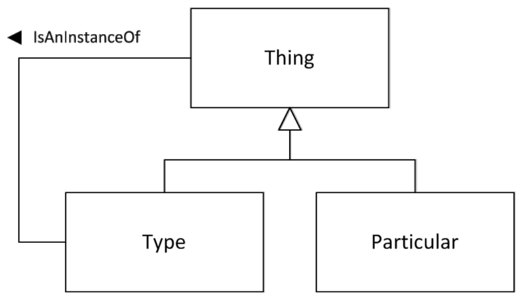


Figure 2 Foundations of Multi-Level Models.

4 Working groups

4.1 Formal Foundations and Ontology Integration

Cesar Gonzalez-Perez (CSIC – Santiago de Compostela, ES), João Paulo A. Almeida (Federal University of Espírito Santo – Vitória, BR), Victorio Albani de Carvalho (Federal Institute of Espírito Santo – Colatina, BR), Anne Koziolek (KIT – Karlsruher Institut für Technologie, DE), Thomas Kühne (Victoria University of Wellington, NZ), Chris Partridge (Brunel University, GB), Michael Schrefl (Universität Linz, AT), Matt Selway (University of South Australia – Mawson Lakes, AU), and Friedrich Steimann (Fernuniversität in Hagen, DE)

The group agreed that a foundation for multi-level modelling would benefit from discussing the philosophical grounding of multi-level modelling, including an elaboration of required or useful ontological commitments and metaphysical choices, plus attempts at standardising terminology. However, the group also agreed that a recognition of the value of a philosophical grounding must not come at the expense of losing sight of pragmatic engineering concerns.

The group decided to be inclusive of various fundamental choices and record consensus as well as different options, i.e., not rule out anything unless it is clearly wrong.

Basics

Model elements refer to things in the world, regardless of a model's genesis.

A model is an artefact external to our minds, constructed with the intention to represent some portion of the world, where "world" means the set of absolutely everything, including real and fictitious, natural and social.

There are particulars (i.e., individuals or ur-elements) and types in the world (see Figure 2). Particulars cannot have instances, whereas types can (they may or may not). Instances of types can be particulars or types, and particulars may be instances of types or not.

Types have a (classification) order and can be related by subtyping. There may be different kinds of types, such as rigid types, phases, or roles.

The above consensus is partially and very roughly shown in Figure 2.

Instantiation

There are two options regarding the semantics of instance-of relationships:

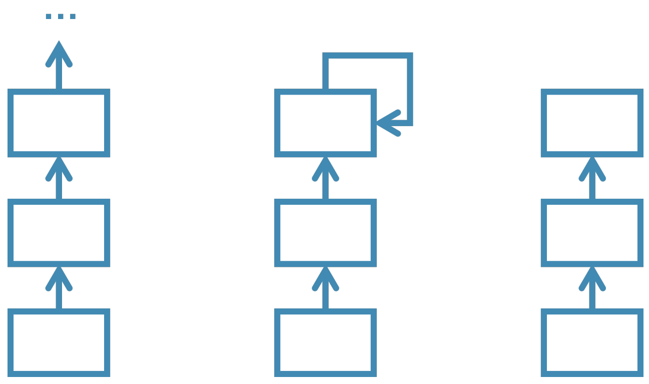


Figure 3 Possible architectures of type hierarchies.

1. All of them have the same semantics, regardless of the kind of type involved.
2. They have different semantics, depending on the kind of type involved.

In any case, direct instance-of relationships connect an instance to its most specific type in a model, whereas indirect instance-of relationships connect an instance to all the types that are ancestors of its most specific type in a model, with ancestry being fully determined by subtyping relationships.

There are also two options regarding dynamic classification:

1. It exists only for non-rigid types, such as phases or roles.
2. It exists for any kind of type.

Overall Architecture

Type hierarchies may be constructed according to three different kinds of overall architecture (see Figure 3):

1. Topless, without a defined top element and an infinite regression, as shown on the left-hand side in the diagram below.
2. Top-unified, with a self-referential top element (i.e. a loop), as shown centrally in the diagram below.
3. Top-special, with a top element that is informally or axiomatically defined and is not a proper instance of anything within the hierarchy, as shown on the right-hand side in the diagram below.

These architectural kinds relate to Münchhausen's trilemma (https://en.wikipedia.org/wiki/Münchhausen_trilemma). In Figure 3, boxes represent model elements, and arrows represent instance-of relationships.

Levels

Model levels are fully determined by instance-of relationships. However, there are other ways in addition to levels to provide hierarchical structure to a model:

- Social or business usage, plus the associated hierarchy of creation and usage of models, related to the process followed to incrementally commit to ontologies and further refine them. For example, a small group may create a standard model that is later adopted (and perhaps extended) by many users.

- Differences in level of abstraction, i.e., organising model elements according to how they refer to the world in more or less abstract ways.
- Specification relationships other than classification between model elements in different levels, i.e., recognising that some model elements may work as specifications of other model elements in the same model. The relationships occurring when using powertypes are a notable example.

Levels may be used in combination, and/or coincide, with other structuring mechanisms as described above, but should not be confused with them.

Levels may or not be numbered but are always fully ordered.

Level Organisation

There are two options regarding how to organise model elements into levels:

1. Level = order, i.e., the level in which a type is placed is equal to its order. This fully determines what level a type belongs to.
2. Level >= order, i.e., the level in which a type is placed is equal or higher than its order, depending on domain semantics, and often informed by associations/links to other model elements. Here, type order constrains but does not fully determines what level a type belongs to.

Crossing Level Boundaries

Instance-of or subtyping relationships cannot flow from a level towards another below it. One reason for this restriction is to avoid cycles.

There are two options as to whether associations or links may cross level boundaries:

1. They can
2. They cannot (cf. "strict metamodeling")

Additional Topics

The group did not have the time to discuss the following relevant issues:

- What are the options for further well-formedness rules?
- Should foundations distinguish between types and other concepts such as templates?
- The notions of prototype, template, or specification were not defined.
- The semantics of instantiation were not fleshed out.
- Can instance-of relationships span multiple (i.e. more than two) levels?
- Are multiple top-level types possible in one model?
- Are primitive types (such as Integer or String) and other related kinds of model elements part of a foundation, or do they pertain to specific models?
- Can primitive types be "imported" into multiple levels?

4.2 Applications and Evaluation of MLM

Iris Reinhartz-Berger (Haifa University, IL), Mira Balaban (Ben Gurion University – Beer Sheva, IL), Philipp Martin Fischer (DLR – Braunschweig, DE), Manfred Jeusfeld (University of Skövde, SE), Agnes Koschmider (KIT – Karlsruher Institut für Technologie, DE), Wendy MacCaull (St. Francis Xavier Univ. – Antigonish, CA), Bernd Neumayr (Universität Linz, AT), Maarten Steen (BiZZdesign – Enschede, NL), and Dustin Wüest (Fachhochschule Nordwestschweiz, CH)

This is a summary of the discussion in the working group entitled "Applications of MLM and Evaluation of MLM Methods", at Dagstuhl seminar 17492.

Our mission was to (1) identify promising application domains for MLM, (2) find common properties for such application domains, (3) identify anticipated benefits of MLM, and (4) determine evaluation criteria for MLM methods.

Application domains

We discussed several application domains that we believe are promising for multi-level modelling:

- Spacecraft engineering: this domain deals with highly complex products (i.e., composed of many parts interacting with each other). Configuration rules exist both on the type level and the instance level (e.g., requiring documentation of which individual physical parts are used).
- Enterprise architectures: enterprise architectures represent large sets of artefacts such as business processes, application systems, and IT infrastructure components. This application domain currently suffers from obstacles in customization at the type level.
- Industrie 4.0 manufacturing and IoT: we did an extensive discussion on this application domain, as elaborated below.

Industrie 4.0

Industrie 4.0 is the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of things, cloud computing, and cognitive computing. Important characteristics of this application domain are:

1. There are many product types, product variants, and product instances, calling for individual mass-production and configuration control.
2. There are different types of models, e.g., conceptual, data, process, ontology models.
3. There is a need for interoperability, supporting a variety of communication protocols.
4. The lowest level, IoT, includes different resources: humans + machines and devices.

The first two characteristics suggest that there are opportunities for MLM to classify products, their types, their composition, and categories. The third one indicates that MLM may allow to reason which products, tools, and processes can be combined in a meaningful way. Further, Industrie 4.0 shall lead to a massive set of things (individuals) that are identified, belong to a certain type of things, and have the need to interact/combine with other things. The move to more flexible mass productions will also require to manage large sets of connected things that share some properties but that are also customized, i.e., have their own set of individual properties and capabilities. In this scenario of configuration control, MLM may improve comprehension of model and variant differences, as well as their common ancestors, thus leveraging inference of communication protocol interoperability.

Common properties and anticipated benefits

We further attempted to determine common properties of promising application domains for MLM and we identified the following:

- Existence of composition hierarchies at different levels (instances, types, metatypes)
- Need for configuration control: adding/modifying components
- “Materialization”/“Realization” links
- Persistence of MLM models due to long lifespan of products
- Different dimensions (views?) required by different stakeholders
- Need for reuse at all levels

We discussed the following anticipated benefits of MLM: (a) supports avoiding accidental complexity; (b) allows reuse during runtime; (c) permits extension of models at runtime; (d) supports easier evolution, updates, and maintenance; (e) promotes improved interoperability; (f) permits generation of constraints from top to bottom (e.g. symmetry). With respect to specialization, MLM promotes flexibility and propagation of properties down the hierarchy and helps avoid problems of inheritance hierarchies, for example problems associated with overriding.

Evaluation of MLM methods

We agreed that evaluation of MLM methods is very important. Hence, we suggested some criteria, including avoidance or detection of modeling mistakes, amount of generated code (i.e., reduction in the amount of manually-written code), compactness of the MLM model for a given modeling problem (e.g., the “bicycle case”), suitability to a wide range of applications, ease of changing, and understandability & comprehensibility. This is not meant to be a full list of evaluation criteria. Further exploration of evaluation criteria and their mapping to the anticipated benefits of MLM are needed.

4.3 Dynamic Aspects of Multi-Level Modelling

Georg Grossmann (University of South Australia – Mawson Lakes, AU), Tony Clark (Sheffield Hallam University, GB), Ulrich Frank (Universität Duisburg-Essen, DE), and Vinay Kulkarni (Tata Consultancy Services – Pune, IN)

Motivation

Business process modelling and management face some challenges which can be addressed by multi-level modelling approaches:

- Current business process management tools have limited support for reuse.
- Process model languages offer generic concepts such as activity and event only and do not support domain specific concepts on a lower, more specific level.
- Every process model, even though there will usually be general domain-specific knowledge, has to be built from scratch.
- Functionality such as copy, paste, and adapt are only provided as basic functionaliy on individual modelling elements and are not part of a process modelling methodology to increase reusability.

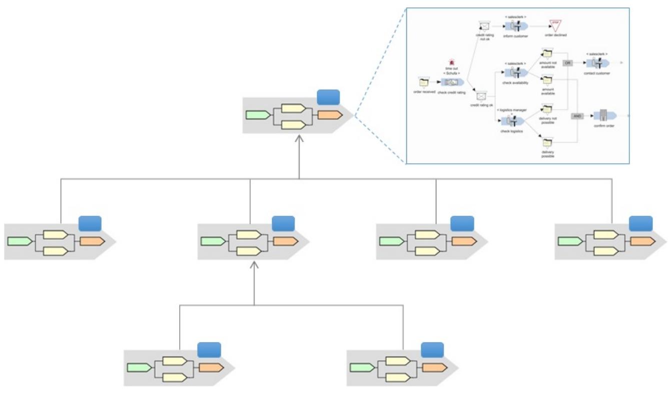


Figure 4 Goal of MLM in process modelling is to have a hierarchy of process models.

These challenges are a threat to productivity, integrity and maintainability of business processes in large and complex organisations such as health care, engineering, law enforcement and enterprise resource planning [5].

The goal of MLM in process modelling is the ability to create and manage a hierarchy of processes as shown in Figure 4 and capture knowledge about this hierarchy.

Opportunities

The main opportunity for multi-level approaches in process modelling is the reusability of concepts on multiple levels:

- There is generally a lack of abstraction in process modelling and MLM provides a basis for a contribution in this area.
- Reuse for processes seems to be a grand challenge. It would be interesting to investigate further the reasons why it is challenging.

▶ **Example 1.** Defining an order management process: It should be possible to define a reusable definition of a process that can be specialised to produce specific types of order management processes. In addition, changes to the abstract definition should be propagated to the specialisations.

▶ **Example 2.** Amazon: What would be needed to fully automate the system? People are currently required to answer questions because the current system is not sufficiently expressive. If the process and data models are sufficiently rich then queries such as “is this product available?” can be automatically answered. Such models can also be used to predict change or actually change the process, and could be relevant, for example, for modelling adaptive systems.

Challenges

Some challenges we have identified during the Dagstuhl seminar discussion include:

- Specialization of process types as discussed in related work [3, 6] is not applicable in a straightforward way, as:
  - we are dealing with non-monotonic extension of control flow, and
  - substitutability constraints are often not satisfied.
- Classification and instantiation:

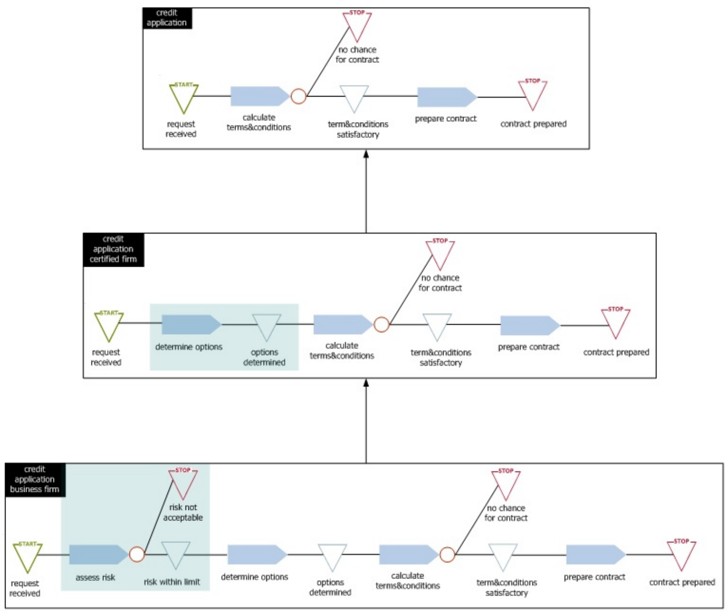


Figure 5 Credit application example highlighting the challenge of making a process more specific.

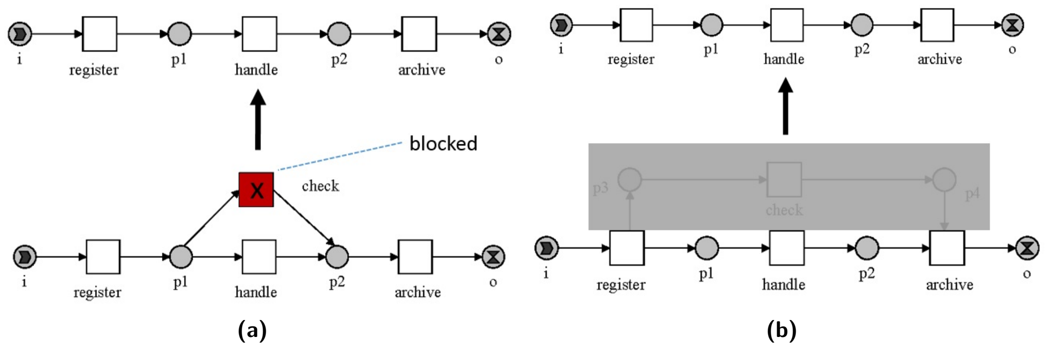


Figure 6 Example taken from van der Aalst et al.[6].

- concepts require clarification with respect to the semantics of "process type" and "process instantiation", and
- are usually given an extensional definition.

- Complexity regarding:
  - (types of) resources required to execute process,
  - roles, and
  - constraints.

Figure 5 shows a *credit application* process example. On the top level is a generic process which is specialized on the lower levels. Using existing specialisation techniques would not allow such specialisation because they would identify it as inconsistent with their specialisation rules. A more flexible approach is needed that formalises the relationship between the processes and specifies what can be changed or adapted on a lower level.

Previous work on protocol inheritance [6] is too limited as shown in Figure 6a. Such an approach would not allow to add activity x into a process on a lower, more specific level.

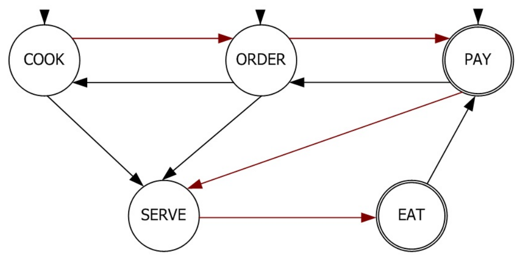


Figure 7 Restaurant transaction example from Wyner and Lee[7].

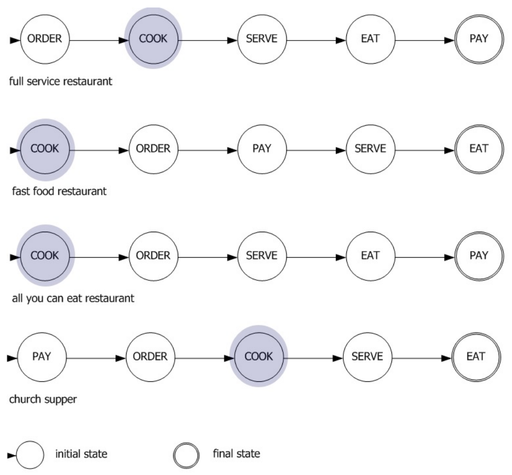


Figure 8 Restaurant transaction specialisations by Wyner and Lee[7].

Another related approach called "projection inheritance" [6] is shown in Figure 6b but has similar limitations to protocol inheritance.

Another challenge mentioned above is the identification and separation of generalisation vs. classification in process modelling. Wyner and Lee [7] provide an example of a restaurant transaction shown in Figure 7. Possible specialisations of the generalised restaurant transaction example are shown in Figure 8.

Use Cases

We have identified some use cases to demonstrate the benefits of MLM approach in process modelling.

Adaptive Systems: Some question that need to be addressed in this context: What needs to be automated within a company? Can models be used to minimize the effort in contingency management?

A more specific use-case could be a supply chain in a car manufacturer organised around 3 tiers at different locations:

- Current ERP systems can generate a production schedule for generating n cars per week.
- In usual circumstances, something may happen that is not planned up-front, for example, road closure, an accident, or natural disaster like fire.
- Current strategies include slowing down the entire network to deal with delays.
- Humans can cope with this strategy but not automated systems.

Simulation: Simulation can be used to help influence design decisions. What kinds of abstraction and languages are required to capture simulations?

Regulatory Compliance: A challenge is the specification of compliance criteria within a model. If possible then the system can check itself against the criteria and can verify changes against compliance. Compliance checking should be automated as far as possible using meta-information encoded in the system.

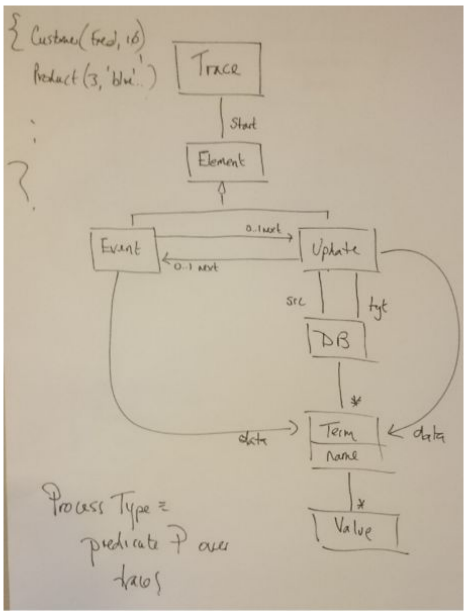


Figure 9 Some core concepts of MLM in process modelling.

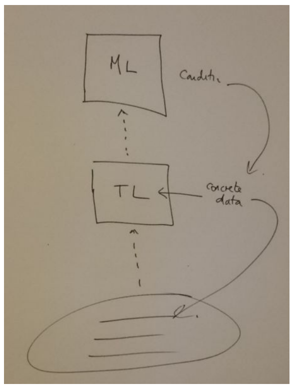


Figure 10 Hierarchy concept of MLM in process modelling.

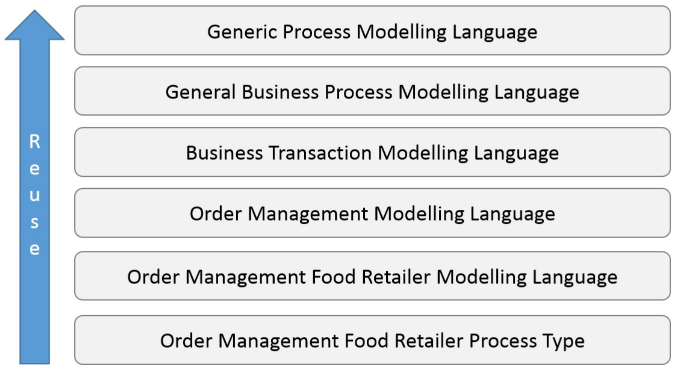

Reuse
Generic Process Modelling Language
General Business Process Modelling Language
Business Transaction Modelling Language
Order Management Modelling Language
Order Management Food Retailer Modelling Language
Order Management Food Retailer Process Type

Figure 11 High-level illustration of the idea.

Proposal

The diagram shown in Figure 9 was discussed during the Dagstuhl seminar and shows a simple semantic domain for process execution. This can be used to study how MLM applies to process definitions. The idea is that process descriptions denote instances of the model shown above. Another way of saying that is that a process definition is a predicate over instances of the semantic domain model. Therefore meta-process definitions are predicates that apply to both process definitions and process executions, and so on.

The diagram shown in Figure 10 is intended to capture the motivation for MLM in the sense that there is a level distinction when the meta-level needs to refer to concrete data at the type level and where the particular concrete data value influences the execution traces that the process definition denotes. If this cannot be established then there is no need for a meta-type level distinction and probably the information can be expressed using inheritance. Note that there is no implication that inheritance and type-of cannot both span a given level.

Another example is provided in Figure 11 to illustrate the idea. A more detailed example of the idea is presented in Figure 12 (pg. 41).

Hierarchy of Knowledge about Processes:

- stepwise abstraction of particular process traces
- all invariant knowledge about all instances (which may be types) is moved to a higher level
- repeated up to a level, where no further abstraction is possible or useful
- promises clear advantages
- no redundant specification
- domain-specific process modelling languages provide support with respect to productivity (through reuse of knowledge)

Next steps

There is some related work available that needs to be investigated further: For example, Schütz [4] wrote a book about multi-level business processes based on his PhD thesis. Recent surveys by La Rosa et al. [1] on variability in business process and configuration of business processes [2] are potentially relevant as well. Further, the concepts of process instantiation need to be refined. Collecting and studying larger examples of possible process hierarchies is required as well as defining relationships between process levels. An analysis of the combination of imperative and declarative process modelling might be worthwhile in this context too.

References

1 M. La Rosa, W.M. Van Der Aalst, M. Dumas, and F.P. Milani. Business process variability modeling: A survey. *ACM Comput. Surv.*, 50(1):2:1–2:45, 2017.

2 M. La Rosa, M. Dumas, A. H.M. ter Hofstede, and J. Mendling. Configurable multi-perspective business process models. *Information Systems*, 36(2):313–340, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.

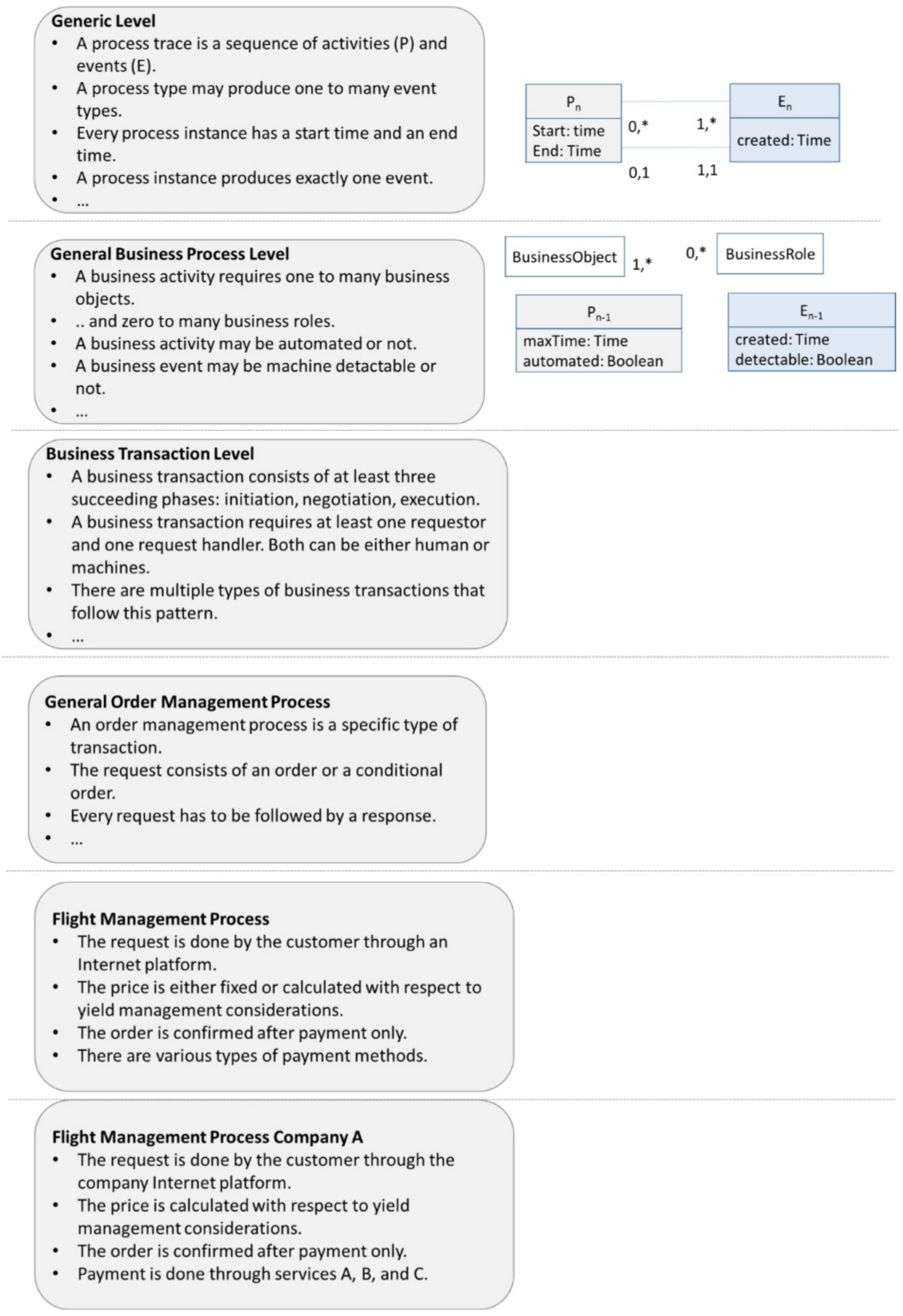
3 M. Schrefl and M. Stumptner. Behavior-consistent Specialization of Object Life Cycles. *ACM Trans. Softw. Eng. Methodol.*, 11(1):92–148, January 2002.

4 C. Schütz. *Multilevel Business Processes – Modeling and Data Analysis.* Springer, 2015.

5 U. Frank. Specialisation in business process modelling: Motivation, approaches and limitations. Technical Report 51, Institut für Informatik und Wirtschaftsinformatik, Universität Duisburg-Essen, 2012.

6 W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.

7 G. M. Wyner and J. Lee. Process Specialization: Defining Specialization for State Diagrams. *Computational & Mathematical Organization Theory*, 8(2):133–155, Jul 2002.



Generic Level
- A process trace is a sequence of activities (P) and events (E).
- A process type may produce one to many event types.
- Every process instance has a start time and an end time.
- A process instance produces exactly one event.
- ...

General Business Process Level
- A business activity requires one to many business objects.
- .. and zero to many business roles.
- A business activity may be automated or not.
- A business event may be machine detactable or not.
- ...

Business Transaction Level
- A business transaction consists of at least three succeeding phases: initiation, negotiation, execution.
- A business transaction requires at least one requestor and one request handler. Both can be either human or machines.
- There are multiple types of business transactions that follow this pattern.
- ...

General Order Management Process
- An order management process is a specific type of transaction.
- The request consists of an order or a conditional order.
- Every request has to be followed by a response.
- ...

Flight Management Process
- The request is done by the customer through an Internet platform.
- The price is either fixed or calculated with respect to yield management considerations.
- The order is confirmed after payment only.
- There are various types of payment methods.

Flight Management Process Company A
- The request is done by the customer through the company Internet platform.
- The price is calculated with respect to yield management considerations.
- The order is confirmed after payment only.
- Payment is done through services A, B, and C.

Figure 12 Illustration of the idea using a specific example.

4.4 Multi-Level Model Transformation

Dirk Draheim (Talinn University of Technologies, EE), Ta'id Holmes (Deutsche Telekom – Darmstadt, DE), and Manuel Wimmer (TU Wien, AT)

This is the report of the working group on transformation with regards to multi-level modeling in the framework of the Dagstuhl seminar on Multi-Level Modelling. The aim of this report is to identify and explain subjects of investigation in the area of multi-level model (MLM) transformation. For this purpose, we will look at a concrete case of an industrial domain-specific language (DSL) project in the domain of cloud data center deployment. Furthermore, we walk through a series of exemplary technologies that are relevant to the field of multi-level transformation.

Introduction

Model transformation is a highly relevant topic. We find it in the form of engineering and exploitation of domain-specific languages in many industrial projects. Also, we find it as essential part of model-driven software engineering [14], i.e., any advanced computer-aided software engineering (CASE) tool initiative and infrastructure, be in the form of forward, reverse or simultaneous round-trip engineering or in the form of CASE tool integration. Always, model transformation deals with the linguistic dimension of a background modeling language infrastructure. Different approaches and terminologies exist for modeling language infrastructures. In practice, we see a huge success of the meta-level type polymorphic transformation approach based on the de-facto standard parser generator ANTLR[1] and its surrounding tool family. The industrial-strength Atlas Transformation Language (ATL)[2] – which is oriented towards established Open Management Group (OMG)[3] terminology – is an example of a systematic meta-level type-driven approach. Therefore, model transformation is always an essential multi-level modeling research topic, even if it is agnostic to ontological multi-level modeling. Subjects of investigation are always:

- Design of the modeling language infrastructures
- In particular, the reductionist syntactic and semantic core of modeling language infrastructures
- Model exchange languages
- Design of organizational roles and processes in meta-model definition and exploitation
- Transformation rules in terms of more than one linguistic meta-level

Beyond the yet unresolved issues in model transformation there is wide potential research and design space in systematically combining ontological modeling with model transformation. Here, additional subjects of investigation are:

- Kinds of exploitation/operationalization of ontological instantiation
- Domain-expert transformation languages
- Object language-based transformation languages
- Extension points in model transformation languages
- Transformation rule refinement

[1] http://www.antlr.org
[2] http://www.eclipse.org/atl
[3] http://www.omg.org

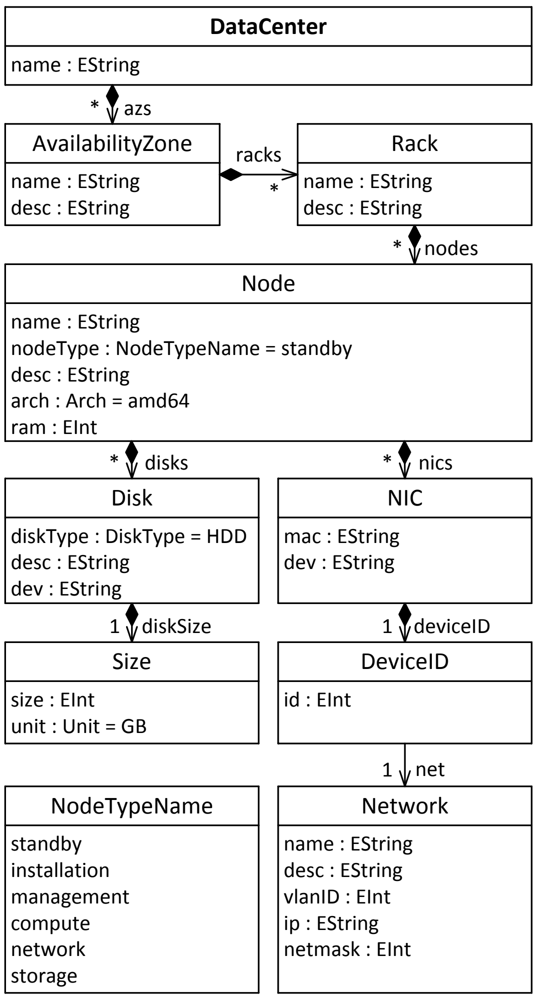


Figure 13 A Data Center Metamodel [11].

- In-place transformation of object languages
- Streamlined/moderated group/community-based domain-specific languages
- Archetype modeling
- Multi-staged model transformation

Multi-Level Model Transformation Case

Having coarsely introduced the field of model transformation in the specific context of multi-level modeling, this section aims at investigating effects using a particular industrial case. For this, the case is introduced first using previous work [11] that followed the classical two-level modeling approach using models that comply to a metamodel. An alternative modeling approach using multi-level models is drafted next. Finally, effects in regard to model transformation are discussed that arise when moving to multi-level modeling.

Automated Data Center Deployment

Figure 13 depicts an example of a simple metamodel of data centers. Instances that describe particular data centers are transformed via code generation. The resulting artifacts constitute the basis for an automated infrastructure as a service (IaaS) deployment. Technologies that realize such deployment from bare metal and that can be leveraged at this stage comprise for example MAAS[4] and JuJu[5] (cf. [11]). Following a model-based approach, the metamodel and conforming models are agnostic towards these technologies. Further technologies such as Fully Automatic Installation (FAI) [10] or TripleO[6] can be supported via additional transformation templates.

Moving to a Multi-Level Model

In the presented metamodel, nodes and disks can be categorized by setting a specific type with the help of an attribute. For instance, a particular node can be classified as a storage node. Storage nodes, typically, aggregate a large number of storage capacity and therefore comprise various disks. A disk, in turn, can be a traditional, mechanical hard disk drive (HDD) or a solid-state drive (SSD).

Instead of typing instances of nodes and disks according to the metamodel and as an alternative to such modeling there exists the possibility to leverage specialization via inheritance relationships. The multi-level model depicted in Figure 14 defines a Node as an instance of (its powertype) NodeType. Using inheritance relationships, specialized concepts such as StorageNode, ComputeNode, and NetworkNode can then be used directly for instantiation (some Products as examples). Finally, a data center (DC) is defined with further instances such as an availibility zone (AZ), a rack, and different nodes.

Collaboration Aspects & Effects on Model Transformation

Using a multi-level modeling approach, part of the multi-level model can be created by a domain expert. That is, the latter is empowered to describe the concepts for the particular domain of expertise.

A data center expert may propose to further distinguish different StorageNodes: ConfidentialStoreNodes shall host confidential, and thus sensitive, data. PerformantStorageNodes comprise SSDs to a good extent while BackupStorageNodes make use of low priced storage (i.e., HDDs).

Delegating (at least parts of) the modeling from a language engineer to a domain expert is interesting; particularly in case of large (domain-specific) models and taxonomies. This is in contrast to a two-level modeling approach where DSL workshops often facilitate the language engineering together with domain experts.

As model transformations operate on models they typically have to relate to concepts for exploiting the semantics within a model. For this reason, in case a metamodel is changed, co-evolution of respective model transformations usually needs to be performed. In a Multi-level modeling approach the relation towards more specialized or instantiated concepts needs to be clarified. That is, while it would be possible in the given example to transform all of the StorageNodes equally, it is not clear how to capitalize specializations without

[4] http://maas.io
[5] http://jujucharms.com
[6] http://wiki.openstack.org/TripleO

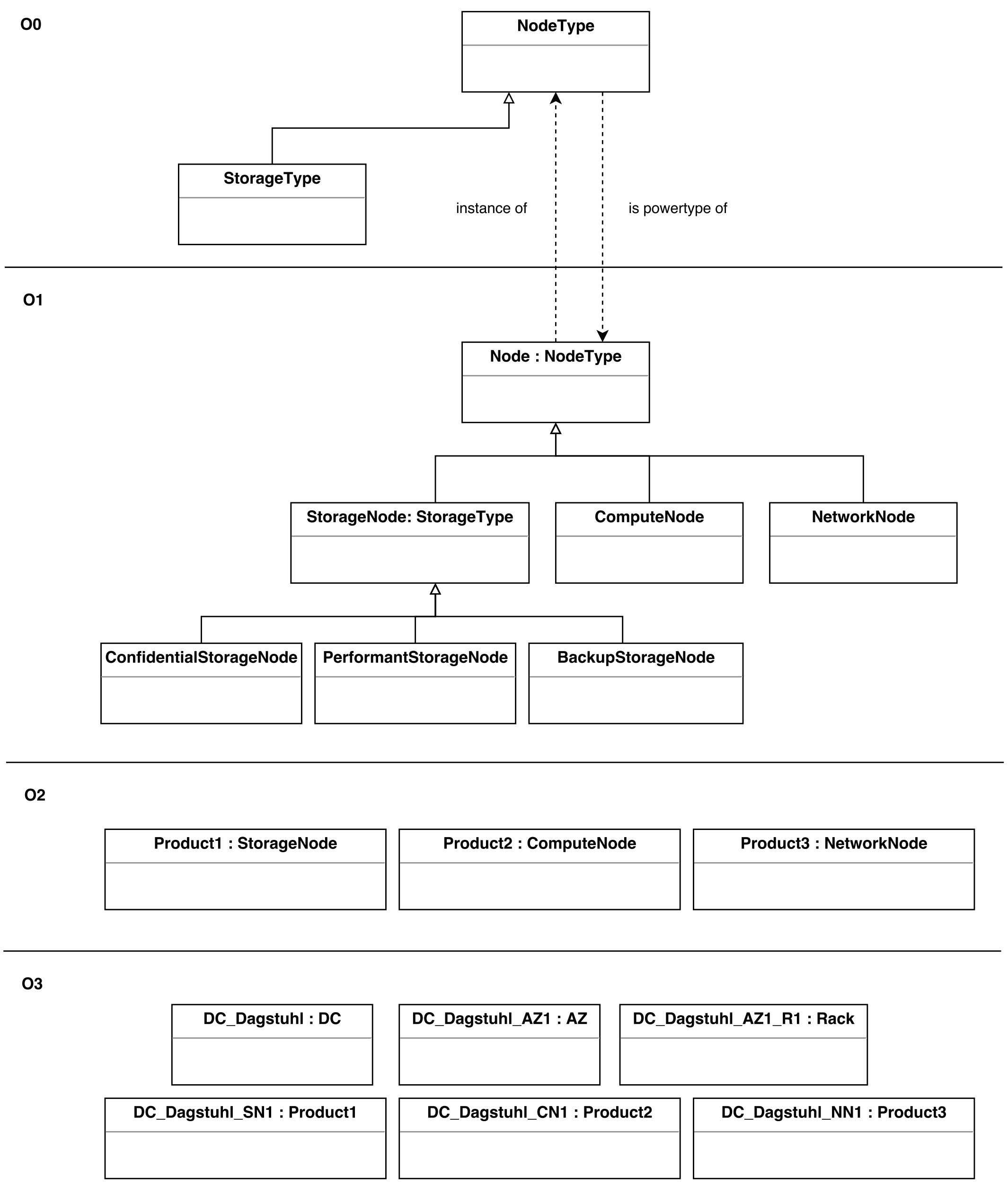


Figure 14 A Multi-Level Model Describing a Data Center.

further domain knowledge. In fact, during deployment, all of such nodes shall be clustered or pooled by a storage solution according to their type. To provide for this, an appropriate model transformation needs to be implemented for the model-based approach. In addition, ConfidentialStoreNodes shall be located in a special (and protected) network.

Thus, model transformation may be subject to particular requirements that need to be communicated by the domain expert in regard to model elements. For this, a multi-level modeling tool support could offer domain experts the possibility to associate such transformation requirements while specifying the semantics of various model elements. This in turn could help language engineers to identify objectives for a model transformation.

Ideally the domain expert would be able to not only specify (parts of) the multi-level model but also (part of) some model transformation, probably using a DSL suited for this very purpose.

Another distinction of multi-level modeling is that a multi-level model, usually, is a standalone model. In case of many multi-level models, standardization of at least the model elements that are used and referenced by model transformations needs to be ensured across all the multi-level models (and transformations).

In the following section we investigate model transformation approaches with a focus on multi-level modeling.

Technologies Relevant to Multi-Level Modeling

In this section, we collect existing work on querying and manipulating multi-level models. The outlined approaches may help in solving the aforementioned case. However, concrete case studies have to be performed in the future to better understand the state-of-the-art in multi-level model transformation. At the end of this section, we discuss some open points that we have already identified by performing a first initial literature study.

In [13, 8] we have developed the generative programming language Genoupe as an extension to C#. With Genoupe we introduced and realized static type checks for dynamically generated types. In [6] we introduced reflective constraint writing that makes possible constraints across the level of the modeling infrastructure (linguistic levels). In [3, 7] we developed a model-based object-relational mapping tool that supports model/database evolution by differentiating model versions and generating data transformation scripts. In [9] we have provided an efficient and effective automatic modularization of ATL transformations.

One of the first papers discussing model transformations dealing with multiple levels is the work by Pataricza and Varro [17]. They introduce generic and meta-transformations based on the VIATRA [16] framework. In particular, meta-transformations allow to define transformations on a particular level, but the execution of these transformations is not done on the next level below, but on two levels below. This kind of transformation has been demonstrated on typical linguistic hierarchies. The work by Guerra and de Lara [5] proposed an extension for the Epsilon Transformation Language (ETL) to support multi-level models. For instance, they introduce deep transformations which are related to meta-transformations as introduced by Pataricza and Varro, but they are more general in the sense that the concrete level can be specified on which the transformation is finally applied. Another interesting aspect which is discussed by Guerra and de Lara is the possibility to refine rules to deal with the specifics of more concrete instances. This would be indeed beneficial for the introduced case. An interesting line for future research would be to compare the extension relationships between transformation rules when applied for two-level models and multi-level models. Finally, Atkinson et al. proposed an extension for ATL to deal with multi-level models [2]. Especially, for the input and output patterns of rules dedicated extensions allow to not only match and generate instances of classes in a two-level fashion as it is provided by standard ATL, but more options are provided to select a particular level. Based on these extensions, deep transformations are possible to be defined. However, no means for redefinition of rules are provided.

Synopsis.

While there exist approaches for out-place transformations based on VIATRA [16], ETL, and ATL, to the best of our knowledge, there are no dedicated transformation approaches for in-place transformations. Such kind of transformations would be of interest for several scenarios such as refactorings, co-evolution repairs, automating edit operations, and defining operational semantics. However, based on the extensions done for out-place transformation scenarios, a deeper comparison and analysis of the new features may also ease the development of extensions for in-place transformation languages. Another direction worth to follow would be the investigation of multi-level programming languages which may be employed for transformation implementations. An experiment may be to combine existing transformation languages and multi-level languages which reside on the same technology. For instance, combining SiTRA [1] – a transformation library for Java – with DeepJava [12] or RubyTL [4] with DeepRuby [15] may show how the concepts of multi-level modeling and model transformation primitives work together.

References

1 D. H. Akehurst, B. Bordbar, M. J. Evans, W. G. J. Howells, and K. D. McDonald-Maier. Sitra: Simple transformations in Java. In O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, editors, *Model Driven Engineering Languages and Systems*, pages 351–364, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

2 C. Atkinson, R. Gerbig, and C. V. Tunjic. Enhancing classic transformation languages to support multi-level modeling. *Software & Systems Modeling*, 14(2):645–666, May 2015. doi:10.1007/s10270-013-0384-y.

3 B. Bordbar, D. Draheim, M. Horn, I. Schulz, and G. Weber. Integrated model-based software development, data access and data migration. In *Model Driven Engineering Languages and Systems*, volume 3713 of *LNCS*, pages 382–396, Berlin Heidelberg, 2005. Springer Berlin Heidelberg.

4 J. S. Cuadrado, J. G. Molina, and M. M. Tortosa. RubyTL: A practical, extensible transformation language. In A. Rensink and J. Warmer, editors, *Model Driven Architecture – Foundations and Applications*, pages 158–172, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

5 J. de Lara and E. Guerra. *Domain-Specific Textual Meta-Modelling Languages for Model Driven Engineering*, pages 259–274, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-31491-9_20.

6 D. Draheim. Reflective constraint writing. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, 24:1–60, 2016.

7 D. Draheim, M. Horn, and I. Schulz. The schema evolution and data migration framework of the environmental mass database IMIS. In *Proceedings of SSDBM 2004 – the 16th International Conference on Scientific and Statistical Database Management*, pages 341–344. IEEE Press, 2004.

8 D. Draheim, C. Lutteroth, and G. Weber. Generative programming for C#. *ACM SIGPLAN Notices*, 40(8):29–33, 2005.

9 M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi. Model transformation modularization as a many-objective optimization problem. *IEEE Transactions on Software Engineering*, 43(11):1009–1032, 2017.

10 M. Gärtner, T. Lange, and J. Rühmkorf. The fully automatic installation of a Linux cluster. Technical Report 379, Computer Science Department, University of Cologne, December 1999. [accessed in February 2018]. URL: http://e-archive.informatik.uni-koeln.de/id/eprint/379.

11 T. Holmes. MING: Model- and view-based deployment and adaptation of cloud datacenters. In M. Helfert, D. Ferguson, V. Méndez Muñoz, and J. S. Cardoso, editors, *Cloud Computing and Services Science, CLOSER 2016, Revised Selected Papers*, volume 740 of

Communications in Computer and Information Science, pages 317–338. Springer, 2016. URL: `doi:10.1007/978-3-319-62594-2_16`.

12 T. Kuehne and D. Schreiber. Can programming be liberated from the two-level style: Multi-level programming with Deepjava. In *Proceedings of the 22Nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications*, OOPSLA '07, pages 229–244, New York, NY, USA, 2007. ACM. URL: `doi:10.1145/1297027.1297044`.

13 C. Lutteroth, D. Draheim, and G. Weber. A type system for reflective program generators. *Journal Science of Computer Programming*, 76(5):392–422, 2011.

14 M. Wimmer, M. Brambilla, and J. Cabot. *Model-Driven Software Engineering in Practice, 2nd edition*. Morgan & Claypool Publishers, 2017. Synthesis Lectures on Software Engineering.

15 B. Neumayr, C. G. Schütz, C. Horner, and M. Schrefl. DeepRuby: Extending Ruby with dual deep instantiation. In *MODELS*, 2007.

16 D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, and Z. Ujhelyi. Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework. *Software & Systems Modeling*, 15(3):609–629, Jul 2016. `doi:10.1007/s10270-016-0530-4`.

17 D. Varró and A. Pataricza. *Generic and Meta-transformations for Model Transformation Engineering*, pages 290–304, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. `doi:10.1007/978-3-540-30187-5_21`.

Participants

- João Paulo Almeida
Federal University of Espírito Santo – Vitória, BR
- Mira Balaban
Ben Gurion University – Beer Sheva, IL
- Tony Clark
Sheffield Hallam University, GB
- Victorio Albani de Carvalho
Federal Institute of Espírito Santo – Colatina, BR
- Dirk Draheim
Talinn University of Technologies, EE
- Philipp Martin Fischer
DLR – Braunschweig, DE
- Ulrich Frank
Universität Duisburg-Essen, DE
- Cesar Gonzalez-Perez
CSIC – Santiago de Compostela, ES
- Georg Grossmann
University of South Australia – Mawson Lakes, AU
- Ta'id Holmes
Deutsche Telekom – Darmstadt, DE
- Manfred Jeusfeld
University of Skövde, SE
- Agnes Koschmider
KIT – Karlsruher Institut für Technologie, DE
- Anne Koziolek
KIT – Karlsruher Institut für Technologie, DE
- Thomas Kühne
Victoria University of Wellington, NZ
- Vinay Kulkarni
Tata Consultancy Services – Pune, IN
- Wendy MacCaull
St. Francis Xavier Univ. – Antigonish, CA
- Bernd Neumayr
Universität Linz, AT
- Chris Partridge
Brunel University, GB
- Iris Reinhartz-Berger
Haifa University, IL
- Michael Schrefl
Universität Linz, AT
- Matt Selway
University of South Australia – Mawson Lakes, AU
- Maarten Steen
BiZZdesign – Enschede, NL
- Friedrich Steimann
Fernuniversität in Hagen, DE
- Manuel Wimmer
TU Wien, AT
- Dustin Wüest
Fachhochschule Nordwestschweiz – Windisch, CH

Report from Dagstuhl Seminar 17502

Testing and Verification of Compilers

Edited by

Junjie Chen[1], Alastair F. Donaldson[2], Andreas Zeller[3], and Hongyu Zhang[4]

1 **Peking University, CN,** `chenjunjie@pku.edu.cn`
2 **Imperial College London, GB,** `alastair.donaldson@imperial.ac.uk`
3 **Universität des Saarlandes, DE,** `zeller@cs.uni-saarland.de`
4 **University of Newcastle, AU,** `hongyujohn@gmail.com`

Abstract

This report documents the Dagstuhl Seminar 17502 "Testing and Verification of Compilers" that took place during December 10 to 13, 2017, which we provide as a resource for researchers who are interested in understanding the state of the art and open problems in this field, and applying them to this and other areas.

Seminar December 10–13, 2017 – http://www.dagstuhl.de/17502
1998 ACM Subject Classification D.2.4 Software/Program Verification, D.2.5 Testing and Debugging, D.3.4 Processors
Keywords and phrases code generation, compiler testing, compiler verification, program analysis, program optimization
Digital Object Identifier 10.4230/DagRep.7.12.50
Edited in cooperation with Hugues Evrard and David R. MacIver

1 Executive Summary

David R. MacIver (Imperial College London, GB)

This report documents the Dagstuhl Seminar 17502 "Testing and Verification of Compilers".

Compilers underpin all software development, but bugs in them can be particularly hard to notice if they result in "silent failure", where a program appears to work but is subtly miscompiled. Thus a compiled program may behave erroneously even when the source form of it appears entirely correct.

Despite the common wisdom that "it is never the compiler's fault", bugs in compilers are in fact relatively common, and finding them is a challenging and active area of research.

This seminar brought together researchers in that area with a broader group of researchers and practitioners in software testing and verification, and in compiler development itself, to share their experiences and discuss the open questions and challenges that the field presents. The goal was to brainstorm new ideas for how to approach these challenges, and to help foster longer-term collaborations between the participants.

The seminar involved a number of talks from participants about their particular areas of work and research, followed by working groups where various specific challenges were discussed. It then concluded with an open panel session on the challenges and concepts of compiler testing and verification.

This report presents the collection of abstracts associated with the participant presentations, followed by notes summarising each discussion session and the concluding panel, which we provide as a resource for researchers who are interested in understanding the state of the art and open problems in this field.

2 Table of Contents

3 Overview of Talks

3.1 How Google Tests Compilers

Edward E. Aftandilian (Google Research - Mountain View, US)

Google has several compiler teams that are responsible for shipping new and updated compilers to the rest of Google on a regular basis.

In this talk, I will discuss how Google's compiler teams validate new and updated compilers, what works well in this process, and where it could be improved.

3.2 Compiler testing for safety-critical applications

Marcel Beemster (Solid Sands - Amsterdam, NL)

Now that so much software is used to control our cars, we have to consider the "safety" of the compilers that are used to generate the machine code from the sources. Processes exist to ascertain so called "Functional Safety". These are based on empirical findings over the past 150 years.

With our SuperTest test suite for C and C++, we can make these processes work for compilers as well, as we can demonstrate the connection between the language specification and the test suite.

We also demonstrate a run-time failure of the formally proven CompCert compiler, thus showing the need for testing alongside prove-techniques. As Knuth already wrote: "Beware of bugs in the above code; I have only proved it correct, not tried it."

3.3 Learning to accelerate compiler testing

Junjie Chen (Peking University, CN)

Main reference Junjie Chen, "Learning to Accelerate Compiler Testing", in Proc. of the 40th Int'l Conf. on Software Engineering (ICSE 2018), to appear.

It is well known that compilers are one of the most important software infrastructures. Compiler testing is an effective and widely-used way to assure the quality of compilers.

While many compiler testing techniques have been proposed to effectively detect compiler bugs, these techniques still suffer from the serious efficiency problem. This is because these compiler testing techniques need to run a large number of randomly generated test programs on the fly through automated test-generation tools (e.g., Csmith).

To accelerate compiler testing, it is desirable to schedule the execution order of the generated test programs so that the test programs that are more likely to trigger compiler bugs are executed earlier. Since different test programs tend to trigger the same compiler bug, the ideal goal of accelerating compiler testing is to execute the test programs triggering different compiler bugs in the beginning. However, such perfect goal is hard to achieve, and thus in this work, we design two steps to approach the ideal goal through learning, in order to largely accelerate compiler testing.

References

1 Junjie Chen, Yanwei Bai, Dan Hao, Yingfei Xiong, Hongyu Zhang, and Bing Xie. Learning to Prioritize Test Programs for Compiler Testing. In: Proceedings of the 39th International Conference on Software Engineering (ICSE 2017), pages 700–711

2 Junjie Chen, Wenxiang Hu, Dan Hao, Yingfei Xiong, Hongyu Zhang, Lu Zhang, Bing Xie. An Empirical Comparison of Compiler Testing Techniques. In: Proceedings of the 38th International Conference on Software Engineering (ICSE 2016), pages 180–190

3 Junjie Chen, Yanwei Bai, Dan Hao, Yingfei Xiong, Hongyu Zhang, Lu Zhang, Bing Xie. Test Case Prioritization for Compilers: A Text-Vector based Approach. In: Proceedings of the 9th International Conference on Software Testing, Verification and Validation (ICST 2016), pages 266–277

3.4 A Generator of Highly Effective Fuzz Testers

Eric Eide (University of Utah - Salt Lake City, US)

A fuzz tester, or “fuzzer,” is effective if it can continually create test cases that reveal defects throughout the system under test. It is difficult to create effective fuzzers for programming language compilers and interpreters because these systems have highly structured inputs. Our goal is to reduce the time and human effort needed to implement effective fuzzers for programming language implementations, and to this end, we are creating Xsmith, a new generator of fuzz testers. Xsmith will generate language fuzzers from specifications, and more importantly, it will inject sophisticated program-generation techniques into the fuzzers it creates. In this talk I will summarize the current status of the Xsmith project. Ultimately, Xsmith will be successful if it permits highly effective fuzz testers to be constructed with significantly less ad hoc code, and thus significantly less effort, than if they had been constructed from scratch.

3.5 Automated Testing of Graphics Shader Compilers

Hugues Evrard (Imperial College London, GB)

We present an automated technique for finding defects in compilers for graphics shading languages. A key challenge in compiler testing is the lack of an oracle that classifies an output as correct or incorrect; this is particularly pertinent in graphics shader compilers where the output is a rendered image that is typically under-specified. Our method builds on recent successful techniques for compiler validation based on metamorphic testing, and leverages existing high-value graphics shaders to create sets of transformed shaders that should be semantically equivalent. Rendering mismatches are then indicative of shader compilation bugs. Deviant shaders are automatically minimized to identify, in each case, a minimal change to an original high-value shader that induces a shader compiler bug. We have implemented the approach as a tool, GLFuzz, targeting the OpenGL shading language, GLSL. Our experiments over a set of 17 GPU and driver configurations, spanning the main 7 GPU designers, have led to us finding and reporting more than 60 distinct bugs, covering all tested configurations. As well as defective rendering, these issues identify security-critical vulnerabilities that affect WebGL, including a significant remote information leak security bug where a malicious web page can capture the contents of other browser tabs, and a bug whereby visiting a malicious web page can lead to a "blue screen of death" under Windows 10. Our findings show that shader compiler defects are prevalent, and that metamorphic testing provides an effective means for detecting them automatically.

3.6 Introduction to Coccinelle

Julia Lawall (INRIA - Paris, FR)

Main reference Yoann Padioleau, Julia L. Lawall, René Rydhof Hansen, Gilles Muller: "Documenting and automating collateral evolutions in linux device drivers", in Proc. of the 2008 EuroSys Conference, Glasgow, Scotland, UK, April 1-4, 2008, pp. 247–260, ACM, 2008.
URL http://dx.doi.org/10.1145/1352592.1352618

Coccinelle is a program matching and transformation tool for C code. The guiding principle behind Coccinelle is the use of a patch-like notation, ie fragments of source code annotation with - to indicate code removal and + to indicate code addition, in order to express transformation rules. Coccinelle was originally developed to automate evolutions in Linux kernel code, and today has been used in over 6000 commits to the Linux kernel, by both the Coccinelle research group and a wide range of Linux kernel developers. Coccinelle is also used on other software, such as wine, qemu, and systemd. This talk gives an overview of Coccinelle and its potential applicability to the maintenance of compiler code.

3.7 Differential Testing of Interactive Debuggers

Daniel Lehmann (TU Darmstadt, DE)

To understand, localize, and fix programming errors, developers often rely on interactive debuggers. However, as debuggers are software, they may themselves have bugs, which can make debugging unnecessarily hard or even cause developers to reason about bugs that do not actually exist in their code. The problem of analyzing debuggers is fundamentally different from the well-studied problem of testing compilers because debuggers are interactive and because they lack a specification of expected behavior.

In this talk, we present an automated analysis technique for interactive debuggers. Our approach, called DBDB, generates debugger actions to exercise the debugger and records traces that summarize the debugger's behavior. By comparing traces of multiple debuggers with each other, we find diverging behavior that points to bugs and other noteworthy differences.

We evaluate DBDB on the JavaScript debuggers of Firefox and Chromium, finding 19 previously unreported bugs, six of which are already confirmed and fixed. Beyond finding bugs, our work is a first step toward agreeing on and specifying the expected behavior of interactive debuggers.

3.8 CUDA Compiler Verification

Thibaut Lutz (NVIDIA - Redmond, US)

CUDA is a widely used programming language for general purpose computation on GPUs. The implementation of the CUDA compiler involves many intermediate stages and components which can each potentially introduce a bug. The programming model of CUDA, which involves parallelism and distributed memory, also makes it more difficult to implement efficient testing.

In this talk, we examine the challenges faced when testing this complex compilation pipeline. An overview of the testing strategies and tools is then presented. Finally, we discuss the lessons learnt from our testing and future directions to improve coverage and find more intricate bugs.

3.9 An Introduction to Software Verification with Whiley

David J. Pearce (Victoria University - Wellington, NZ)

Main reference David J. Pearce, Lindsay Groves: "Designing a verifying compiler: Lessons learned from developing Whiley", Sci. Comput. Program., Vol. 113, pp. 191–220, 2015.
URL http://dx.doi.org/10.1016/j.scico.2015.09.006

In this talk, I'll employ live coding to demonstrate the Whiley programming language and its accompanying "verifying compiler". The language is focused on ensuring programs meet their specifications. Whiley programs can be verified at compile time, and doing this prevents a range of common errors impossible (e.g. divide-by-zero, array out-of-bounds, etc). Sophisticated specifications can be written using a quantifiers and other apparatus from first-order logic. Programming in Whiley feels surprisingly natural and the goal is to make it comparable to interacting with a type checker. The language has been used for the last three years to teach a large undergraduate class about program specification, and we have benefited considerably from this experience.

3.10 TreeFuzz: Learning Probabilistic Models of Input Data for Fuzz Testing

Michael Pradel (TU Darmstadt, DE)

Joint work of JIbesh Patra, Michael Pradel

Fuzzing is a popular technique to create test inputs for software that processes structured data. It has been successfully applied in various domains, ranging from compilers and interpreters over program analyses to rendering engines, image manipulation tools, and word processors. Existing fuzz testers are either tailored to a specific data format, rely on whitebox analysis of the software under test, or infer a context-free grammar of the input format. This talk presents TreeFuzz, an approach to learn probabilistic models of input data from a corpus of examples and to fuzz-generate new data from the inferred models. The approach supports any data format that can be represented as a tree and learns models without any human intervention. To support a wide range of different properties of input data, TreeFuzz is designed as a framework with an extensible set of generative models. The learned models are more expressive than context-free grammars, producing test inputs that reach deep into the software under test. Our evaluation applies TreeFuzz to a programming language, JavaScript, and a markup language, HTML. The results show that the approach creates mostly correct data: 96.3% of the generated JavaScript programs are syntactically valid and there are only 2.06 validation errors per kilobyte of generated HTML. Furthermore, we show that the performance of both learning and generation scales linearly with respect to the size of the corpus, making it easily applicable to tens of thousands of examples. Finally, we show that the fuzz-generated data are useful for testing: Using TreeFuzz-generated JavaScript programs for differential testing of JavaScript engines exposes various inconsistencies among browsers, including browser bugs and unimplemented language features.

3.11 A Few Stories about Compiler Bugs

John Regehr (University of Utah - Salt Lake City, US)

Main reference Nuno P. Lopes, David Menendez, Santosh Nagarakatte, John Regehr: "Provably correct peephole optimizations with alive", in Proc. of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015, pp. 22–32, ACM, 2015.
URL http://dx.doi.org/10.1145/2737924.2737965

Engineering a successful compiler is difficult because the major goals for a compiler – fast compile times, few compiler bugs, and high-quality generated code – are individually difficult and also conflict with each other. Formal methods can help solve all of these problems, but the formal methods must be developed in such a way that they are maintainable over a long time period and are usable and understandable by compiler developers. One promising approach is translation validation, because it runs to the side of a compiler, requiring few if any modifications to the tool.

3.12 Automatic non-Functional Testing of Configurable Generators

Gerson Sunyé (University of Nantes, FR)

Generative software development has paved the way for the creation of multiple code generators and compilers that serve as a basis for automatically generating code to a broad range of software and hardware platforms. With full automatic code generation, the user is able to easily and rapidly synthesize software artifacts for various software platforms. In addition, modern generators (i.e., C compilers) become highly configurable, offering numerous configuration options that the user can use to easily customize the generated code for the target hardware platform.

In this context, it is crucial to verify the correct behavior of code generators. Numerous approaches have been proposed to verify the functional outcome of generated code but few of them evaluate the non-functional properties of automatically generated code, namely the performance and resource usage properties.

In this presentation, I address two problems:

1. Non-functional testing of code generators: We benefit from the existence of multiple generators with comparable functionality (i.e., code generator families) to automatically test the generated code. We leverage the metamorphic testing to detect inconsistencies in code generators families by defining metamorphic relations as test oracles. We define the metamorphic relation as a comparison between the variations of performance and resource usage of code, generated from the same code generator family.
2. Handling the diversity of software and hardware platforms in software testing: Running tests and evaluating the resource usage in heterogeneous environments is tedious. To handle this problem, we benefit from the recent advances in lightweight system virtualization, in particular container-based virtualization, in order to offer effective support for automatically deploying, executing, and monitoring code in heterogeneous environment, and collect non-functional metrics (e.g., memory and CPU consumptions).

We evaluate our approach by analyzing the performance of Haxe, a popular code generator family. Experimental results show that our approach is able to automatically detect several inconsistencies that reveal real issues in this family of code generators.

3.13 Learning to Synthesize Programs

Yingfei Xiong (Peking University, CN)

In many scenarios including testing we need to find the most likely program under a local context,where the local context can be the program under test, an incomplete program, a partial specification, natural language description, etc. We call such problem program estimation. In this paper we propose an abstract framework, learning to synthesis, or L2S in short, to address this problem. L2S combines four tools to achieve this: syntax is used to define the search space and search steps, constraints are used to prune off invalid candidates at each search step, machine-learned models are used to estimate conditional probabilities for the candidates at each search step, and search algorithms are used to find the best possible solution. The main goal of L2S is to lay out the design space to motivate the research on program estimation.

We have performed a preliminary evaluation by instantiating this framework for synthesizing conditions. On 4 projects from Defects4J, we can successfully synthesize the correct conditions at top 10 in 64.7%-85.7% of the cases by training only on the source code of the project, and the precision is related to the size of the projects.

3.14 Debugging Debug Information

Francesco Zappa Nardelli (INRIA - Paris, FR)

Hidden, obscure, and badly specified components lurk at the very heart of our computing infrastructure. Consider debugging informations. Debugging informations are obviously relied upon by debuggers and play a key role in the implementation of program analysis tools, but, more surprisingly, debugging informations can be relied upon by the runtime of high-level programming languages (e.g. to unwind the stack and implement C++ exceptions). Unfortunately debugging informations themselves can be pervaded by subtle, undebuggable, bugs. We will describe how to perform validation and synthesis of the DWARF stack unwinding debug tables, and we will report on ambitious plans we might build on reliable debug information.

3.15 Tutorial: Compiler Optimisations and Shared Memory Concurrency

Francesco Zappa Nardelli (INRIA - Paris, FR)

Main reference Robin Morisset, Pankaj Pawan, Francesco Zappa Nardelli: "Compiler testing via a theory of sound optimisations in the C11/C++11 memory model", in Proc. of the 34th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2013), pp. 187–196, ACM, 2013.
URL https://doi.org/10.1145/2499370.2491967

Compiler optimisations introduce unexpected behaviours in shared memory concurrent programs; programming languages thus define memory models to specify precisely which behaviours can be observed and, indirectly, which optimisations a compiler can apply. Taking the C11/C++11 standard as running example, we will investigate how to reason about the correctness of compiler optimisations (or lack thereof), and how to build a tool to fuzzy test mainstream compilers against the memory model. Au passage, we will describe our failure to specify a reasonable memory model for general purpose programming languages, and how this failure paved the way for exciting research on validation and implementation of compilers.

3.16 Fuzzing with Inferred Grammars

Andreas Zeller (Universität des Saarlandes, DE)

Fuzzing a program is greatly simplified if one has a grammar that describes the lexical and syntactical (possibly even semantical) properties of the input. I show how grammar and code coverage can successfully guide test generation, and how tracking how a program processes input characters can produce grammars that again can be fed into test generation.

4 Working groups

4.1 Benchmarking Compiler Testing

Junjie Chen (Peking University, CN)

This group discussed the creation of *benchmarks* for compiler testing.

These benchmarks would take known buggy versions of a compiler and curate a set of bugs, along with test cases triggering them, a bug fix, and information about the bug (e.g. location of it, conditions required to trigger it).

The working group concluded that such benchmarks would be worthwhile, as they would significantly speed the development of new compiler testing techniques by evaluating their effectiveness in finding a known set of bugs.

4.2 Beyond Compiler Testing

Michael Pradel (TU Darmstadt, DE)

A working group chaired by Michael Pradel discussed how to adapt ideas from compiler testing beyond compilers.

The group went through a set of developer tools that also take source code as their input, such as lint-like bug finding tools, debuggers, code search, static analyses, and symbolic execution engines. For each tool, a set of challenges were identified

One recurring challenge is that some tools require further inputs in addition to program source code, e.g., user interactions for debuggers or search queries for code search engines.

Another recurring challenge is that the desired behavior of several tools is only informally specified. For example, testing whether a bug finding tool works as expected requires knowledge about which code the tool is supposed to (not) flag as buggy.

The working group concludes that

1. Testing other development tools than compilers is an exciting research direction.
2. Existing work on generating test programs could be reused
3. There remain additional challenges to be addressed in future work.

4.3 Test-Case Reduction

David R. MacIver (Imperial College London, GB)

This working group discussed common issues and themes of test-case reduction, and open questions for future work in the area.

The two issues common to test case reduction identified were:

- It is very difficult to write good predicates for test-case reduction—the most common issue users of C-Reduce face is when they have written a predicate for their test case that is "too broad" and the bug slips off the originally identified bug into a less interesting one.
- It is also very difficult to write test-case reducers.

The group proposed that the solution to the latter was to make it easier to extend existing test-case reducers (C-Reduce and StructureShrink were both mentioned) to add new "reduction passes".

It is unclear that there is a well defined notion of what test-case reduction *does*. Existing literature has focused a great deal on size, but often the smallest example is not the clearest, and often even among the smallest size examples more reduction can be performed to improve readability. For example, C-Reduce always inlines functions even if this makes the test-case larger. Work such as [2, 3] on combining test-case reduction with generalisation was highlighted as a potential solution to this problem.

We also observed that there are common idioms in test-case reduction that are currently under-documented–e.g. when to be greedy vs non-greedy, trying large reductions first and then backing off, combining multiple shrinks together when single ones fail.

Open questions we considered interesting for further research were:

- How best can we present "readable" examples to end users, and what role does pure test-case reduction have in that?
- In light of that how can we best determine when one test-case is "better" than another?
- How can we cope with situations where shrinks must be chained together given that it is potentially very expensive to identify such chains?

References

1 John Regehr, Yang Chen, Pascal Cuoq, Eric Eide, Chucky Ellison, and Xuejun Yang. " Test-Case Reduction for C Compiler Bugs". In Proceedings of 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2012), Beijing, China, June 2012.

2 Groce, Alex, Josie Holmes, and Kevin Kellar. "One test to rule them all." Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, 2017.

3 Braquehais, Rudy, and Colin Runciman. "Extrapolate: generalizing counter-examples of functional test properties." (2017).

4.4 Program Generation

David R. MacIver (Imperial College London, GB)

This was a working group about the problem of generating programs to test compilers.

The group identified two key properties which are in tension with each other. These were summarised as "Semantic Validity" and "Completeness" (or "Coverage").

Semantic validity is the property of making sure you only generate "good" inputs–for example, ensuring that a C program is free of undefined behaviour.

Completeness on the other hand is the property of reaching into the dark corners of the compiler to find interesting behaviour rather than just shallowly testing the surface–finding interesting programs that trigger unusal behaviours.

These are in tension because a generator of valid programs must necessarily be somewhat conservative in what it emits. However, they can also mutually support each other, as program generator that generates too many invalid programs will tend to to succeed only in generating trivial valid programs.

It was felt that most interesting open questions were about how to that improve completeness while preserving semantic validity.

The two main approaches suggested for pursuing this were to provide better tooling for describing language to attempt to use machine learning to discover the range of valid programs from the behaviour of the compiler.

The group also identified the problem of generating *idiomatic* programs–a common problem is that generated test cases are ignored by compiler developers because they look unnatural, and the group believed that generating test cases that look like normal code would help with this.

5 Panel discussions

5.1 Challenges and Concepts of Compiler Testing and Verification

Yingfei Xiong (Peking University, CN)

In the discussion session three topics were discussed.

First, all participants brainstormed the current open challenges in compiler testing and verification.

Second, the name of the field was discussed. The participants agreed that there is no universal good name to the field. For example, "compiler" captures most of the work in this field and is easy to explain to outsiders, but has problem of omitting some subjects such as debuggers. As a result, we do not try to name the field for now, and keep the name "compiler testing and verification" for future editions of the seminar. Here "compiler" is broadly defined to include software tools that take programs as input.

Third, some confusing terms were discussed. For example, program could refer to the input program to the compiler and the compiler itself. Tests could refer to the tests of the compiler or the tests in the compiled program. However, no good solution came out from the discussion and each paper should clearly define the terms before use.

The panel identified the following challenges:

1. Input program generation
   a. For advanced type systems
   b. Testing subtyping and other complex relations
   c. Generating valid programs
   d. Generating invalid yet effective test programs
   e. Covering new language features
   f. Dealing with undefined behavior
2. Finding effective test oracles
3. Efficient testing
4. Undefined semantics
5. Reproducibility of the test cases
6. Dealing with concurrency and nondeterminism
7. Readability of test cases
8. Generalization of test cases
9. Finding *important* bugs.
   a. What is importance?
   b. How can we communicate it?
10. Verifying compiler optimizations
11. Using translation validation for better testing.
12. Providing efficient and easy-to-use tool implementations

Participants

- Edward E. Aftandilian
Google Research –
Mountain View, US
- Marcel Beemster
Solid Sands – Amsterdam, NL
- Junjie Chen
Peking University, CN
- Nathan Chong
ARM Ltd. – Cambridge, GB
- Eric Eide
University of Utah –
Salt Lake City, US
- Hugues Evrard
Imperial College London, GB
- Dan Hao
Peking University, CN
- John Hughes
Chalmers University of
Technology – Göteborg, SE
- Dan Iorga
Imperial College London, GB
- Julia Lawall
INRIA – Paris, FR
- Daniel Lehmann
TU Darmstadt, DE
- Thibaut Lutz
NVIDIA – Redmond, US
- David MacIver
Imperial College London, GB
- Jessica Paquette
Apple Computer Inc. –
Cupertino, US
- David J. Pearce
Victoria University –
Wellington, NZ
- Michael Pradel
TU Darmstadt, DE
- John Regehr
University of Utah –
Salt Lake City, US
- Raimondas Sasnauskas
SES Engineering –
Luxembourg, LU
- Marija Selakovic
TU Darmstadt, DE
- Gerson Sunyé
University of Nantes, FR
- Nikolai Tillmann
Facebook – Seattle, US
- Yingfei Xiong
Peking University, CN
- Francesco Zappa Nardelli
INRIA – Paris, FR
- Andreas Zeller
Universität des Saarlandes, DE
- Hongyu Zhang
University of Newcastle, AU