Towards Rapid Design of Compartmental Models

Zahra Fiyouzisabah
zahra.fiyouzisabah@umontreal.ca
Université de Montréal
Canada

Jessie Galasso
jessie.galasso-carbonnel@mcgill.ca
McGill University
Canada

Marios Fokaefs
fokaefs@yorku.ca
York University
Canada

Michalis Famelis
famelis@iro.umontreal.ca
Université de Montréal
Canada

ABSTRACT

In times of crisis, epidemiologists can come under great pressure to model rapidly evolving diseases and to produce analyses about the effects of potential public health interventions. Taking previously developed, tested, and validated model components as the base on which to prototype new infectious disease models can save precious time and effort. However, there is currently no systematic process for quickly navigating a corpus of existing epidemiological models or identifying and reusing their most useful components. In this paper, we propose a vision to accelerate the creation of prototype compartmental models for infectious diseases. We outline a semi-automated process that epidemiologists can use to create prototypes that have been partially completed with reused fragments from existing models. Epidemiologists can thus focus on modelling the novel aspects of an ongoing public health crisis, as opposed to aspects of it that are already more or less well understood in previous work. Our approach comprises five steps in total, including identifying useful components in a corpus of infectious disease models, generating potential candidate prototypes, and organizing them in a formal data structure that allows navigation and exploration by the modellers. We outline 13 challenges ahead and discuss potential solutions based on formal modelling techniques.

CCS CONCEPTS

• **General and reference** → **Design**; • **Software and its engineering** → **Reusability**; **Software prototyping**; **Software evolution**; *Software version control*; Maintaining software; **Collaboration in software development**; **Model-driven software engineering**; **Abstraction, modeling and modularity**; **Domain specific languages**; *Rapid application development*; • **Applied computing** → **Life and medical sciences**.

KEYWORDS

Domain Specific Modelling, Reuse, Evolution, Prototyping, Formal Concept Analysis, Scientific Computing, Compartmental Models

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3688340

ACM Reference Format:
Zahra Fiyouzisabah, Jessie Galasso, Marios Fokaefs, and Michalis Famelis. 2024. Towards Rapid Design of Compartmental Models. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3652620.3688340

1 INTRODUCTION

Epidemiology is the study of distribution, patterns, and causes of health events in human populations [23]; infectious disease modelling [19] is a critical aspect of it [13]. During health crises, such as the early stages of the COVID-19 pandemic, scientists can face intense pressure and tight deadlines to create and validate disease models to help policy makers address emerging threats with public health measures under conditions of "deep uncertainty" [8]. Despite each epidemic's uniqueness, epidemiological models are often highly stylized and share many common basic concepts. Moreover, following the rise of computational science, they are typically software artifacts, defined at various levels of abstraction [23]. The software reuse principle thus indicates that reusing components from existing models can save time [21], allowing scientists to focus on the unique aspects of their study. However, there is no systematic method for reusing existing epidemiological models, forcing scientists to manually search for, understand, and integrate parts from existing models, making the process cumbersome and prone to errors. Here, we propose a vision for tool-supported systematic model reuse for the rapid development of partially completed prototype epidemiological models.

In this paper, we present a vision to use independently evolved models to aid the development of new compartmental models. Our approach is novel in leveraging software engineering techniques like systematic reuse and model merging to refine and evolve epidemiological models for emerging public health challenges.

We explain our vision with an example inspired from real COVID-19 models [31]. Carol, an epidemiology researcher, wants to study the COVID-19 pandemic. She begins with an basic SIR model m_0, shown in Figure 1; this is a *compartmental model*, widely used in epidemiology [32]. It has three *compartments*, each representing a segment of the overall population: (a) **S**usceptible: the segment of the population that is susceptible to the virus but not yet sick, (b) **I**nfectious: the segment of the population that has fallen ill, and (c) **R**ecovered: those that have recovered from the illness. The population in each compartment can grow or shrink during the epidemic through *flows*, represented as arrows, labeled with the

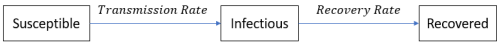


Figure 1: Initial basic SIR compartmental model m_0.

rates of these transitions, e.g., transmission and recovery rates. In the rest of the paper, whenever we refer to "models", we always refer to such compartmental models.

To further evolve her model, Carol examines models created by others, hoping to reuse them in her model. In her research, she finds various models, fragments of which are relevant to her study: the model m_1 in Figure 2a includes exposed and vaccinated groups; the model m_2 in Figure 2b includes compartments for different levels of infection severity based on symptoms (asymptomatic, mildly symptomatic, and severely symptomatic); and the model m_3 in Figure 2c details the hospitalization status of patients. These models contain parts relevant and similar to Carol's study. Reusing them would be beneficial as they have already been tested and validated, reducing the likelihood of errors and saving her significant time and effort in evolving her prototype model.

However, Carol is faced with a software reuse and model evolution problem. A key challenge is the significant time needed to explore and understand existing models to find useful parts to reuse [27], making the process inefficient and prone to overlooking important modelling ideas. Another challenge is that manually integrating parts and ideas from various models is challenging and error-prone. Ideas in different models might conflict with each other, or specific rules might need to be applied to combine different parts smoothly [29]. These challenges are exacerbated under the "deep uncertainty" of a public health crisis, when scientists must rapidly propose informed analyses to the authorities operating under pressure.

We envision aiding Carol to evolve her prototype model by providing her with a tool to explore and integrate fragments from multiple models. This would allow Carol to: *a)* explore ideas present in existing models; *b)* navigate designs based on these ideas; *c)* select relevant fragments from various models; and *d)* get a composite model incorporating them. We envision this resulting composite model as an *inherently incomplete basis* from which to develop the prototype. Carol can further develop and refine it with her unique additions to develop her complete solution.

Carol would first need to specify a corpus of relevant models. This could be an explicit enumeration or a query to some public or private repository of models. The tool might start with fundamental model elements like *Susceptible*, *Exposed*, and *Recovered*, which are common to all models. It then presents additional elements from the corpus for selection, such as *Infectious*, *Vaccinated*, and *Dead*, for Carol to select and refine her prototype. For example, if Carol selects *Vaccinated*, the tool would display models that include the *Vaccinated* compartment alongside *Susceptible*, *Exposed*, and *Recovered*. Moreover, new related elements would appear, allowing Carol to further refine her model. Carol might store interesting versions of her prototype in a version-controlled repository and continue exploring the design space until she has identified the set of elements that are closest to her needs. For example, she might eventually arrive at a composite model m_p shown in Figure 3 . She can then complete and build upon m_p, adding her own unique features and focusing on novel aspects specific to her study.

2 PROPOSED APPROACH

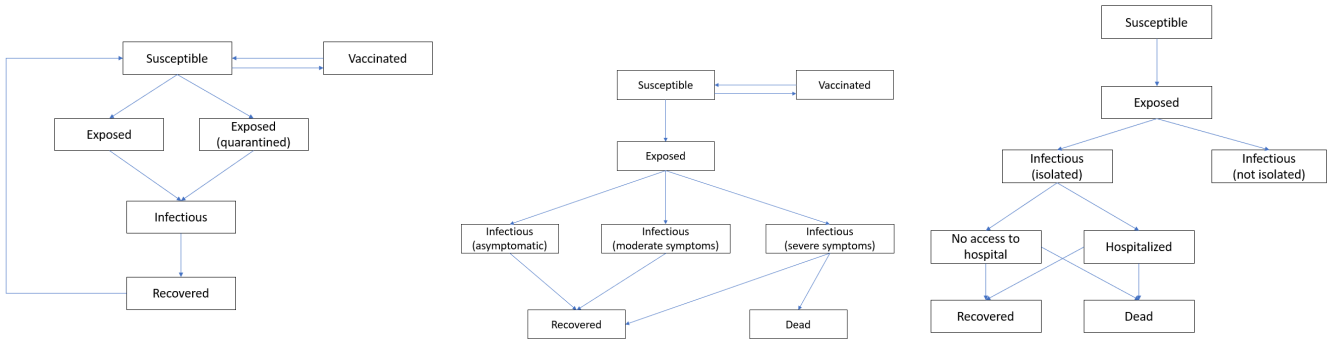
Our vision is to help epidemiologists explore modelling ideas in a corpus of existing models and reusing them to rapidly generate a composite model. This model can then be used as the basis for further evolution and refinement. In other words, we want to help Carol integrate ideas from $m_1 - m_3$ into her m_0, to get rapidly to m_p, which she can then use as the basis for further development. We propose an approach with five steps: 1) Feature identification and clustering, 2) dependencies identification, 3) fragment merging, 4) creation of concept lattice, and 5) interactive exploration of the concept lattice. We discuss the challenges in each one below, along with our preliminary ideas.

Feature Identification and Clustering. The first step is to treat the existing models as *variants* and identify distinct *modelling ideas* within them. Carol's corpus comprises the models m_1, m_2, m_3, and her base model m_0. In our example, m_0 is too trivial, so for the rest of the paper, we assume that Carol discards it. To identify variant ideas and keep the level of abstraction appropriately high, we need to group models' elements that occur frequently together. For example, in model m_2, the *Infectious (asymptomatic)*, *Infectious (moderate symptoms)*, and *Infectious (severe symptoms)* compartments are related and appear together, so they can be clustered as one modelling idea (e.g., "Infectious with varying degrees of symptomaticity").

The first challenge **(C-1)** is thus *feature identification* [4]. As a universally accepted definition of what features *are* remains elusive [7], and no feature extraction techniques exist specifically for compartmental models, we aim to develop a technique for identifying them at a level of abstraction appropriate for epidemiologists. In this paper, we take a "feature" (or a "modelling idea") to mean a reusable set of elements shared by different models. As a first step, we limit features to compartments, using clustering algorithms to identify recurring sets of them. Obviously, flows between compartments and rate parameters are also crucial. For this, we are currently working on an approach based on *Pattern structures* [16], which can extract clusters of complex and non-boolean descriptions [10].

Using a naive model differencing approach, we identify clusters of compartments that appear together. Given that different names might have been used for the same compartments across different models, natural language processing (NLP) techniques would help to recognize synonyms or terminology variations between compartments. Identified clusters likely constitute *features* [7] of a model; further domain expert input is needed for confirmation. For Carol's corpus, we find the clusters: SER {*Susceptible*, *Exposed*, *Recovered*};V {*Vaccinated*}; EI {*Exposed (quarantined)*, *Infectious*}; I_{sym} {*Infectious (asymptomatic)*, *Infectious (moderate symptoms)*, *Infectious (severe symptoms)*}; D {*Dead*}; IH {*Infectious (isolated)*, *Infectious (not isolated)*, *Hospitalized*, *No access to hospital*}.

The initial clustering may group unrelated compartments, such as *Infectious (isolated)* and *Hospitalized*, despite representing different modelling ideas. The second challenge **(C-2)** is adjusting the clustering to reflect semantic relationships between distinct features. This could involve an interactive step where users refine



(a) m_1 **(b)** m_2 **(c)** m_3

Figure 2: Corpus of additional compartmental models to consult.

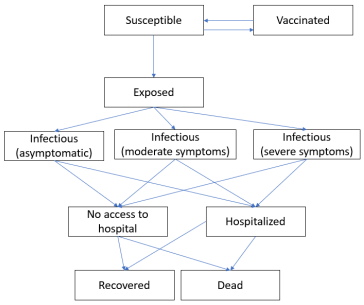


Figure 3: Desired prototype m_p, with ideas from $m_1 - m_3$.

clusters. For example, Carol might: (a) Split EI into two distinct clusters: EQ for *Exposed (quarantined)* and Inf for *Infectious*. (b) Divide the IH cluster into I_{iso} for *Infectious (isolated and non-isolated)* and H for *Hospitalized* and *No Access to Hospital*. We show the resulting clusters in Table 1. We note that this decomposition might not always be perfect. For example, both SER and EQ contain concepts related to exposure. A third challenge **(C-3)** is to support the user in this task, e.g., using machine learning techniques based on linguistic similarity. A possibility would be to leverage specialized ontologies (e.g., the Genomic Epidemiology Ontology [18]) to find candidate cluster refinements based on semantic similarity.

Dependencies identification. The next step is to identify semantic dependencies between clusters, such as when some are alternatives and cannot coexist in the same model. For example, clusters I_{sym} and I_{iso} both model how the infection manifests itself in a population; the former segments patients by symptoms while the latter does so by isolation status. The challenge **(C-4)** is how to identify such semantic dependencies. One approach is to use semantic similarity techniques, supported by an epidemiology ontology, to match clusters based on their names or contents. In our example, mapping the content names of the clusters Inf, I_{sym}, and I_{iso} to terms within the ontology would reveal a semantic relationship. But for other clusters, such as *SER*, *D*, and *H*, no semantic dependencies would be found beyond ordering. However, we note that fully automated detection of semantic similarity might not be sufficient. For example, *SER* and *EQ* clusters both contain

Cluster Label	Containing Compartments
SER	Susceptible, Exposed, Recovered
V	Vaccinated
EQ	Exposed (quarantined)
Inf	Infectious
I_{sym}	Infectious (asymptomatic), Infectious (moderate symptoms), Infectious (severe symptoms)
D	Dead
I_{iso}	Infectious (isolated), Infectious (not isolated)
H	Hospitalized, No access to hospital

Table 1: Clusters of features in Carol's input corpus

exposure-related compartments but they are not alternatives as 'Exposed' and 'Exposed (quarantined)' appear together in m_1. This highlights the need **(C-5)** for an iterative, interactive process to refine clusters and their dependencies based on specific use cases.

Fragment Merging. Our goal is to help modellers identify useful combinations of ideas from the input corpus. We extend this corpus with models that combine identified clusters, enriching it with new feature combinations. We propose generating new models from existing ones using algebraic model merging [9]. For example, a *merge* function can combine the models, m_1 and m_2 to produce the model, m_{12}, in Figure 4.

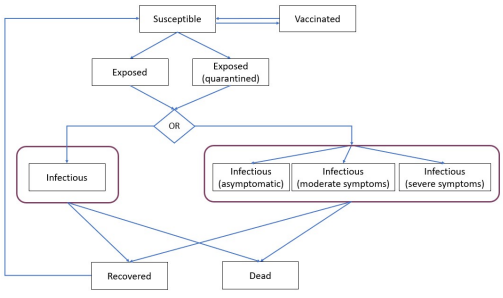


Figure 4: Model m_{12}: the result of merging m_1 and m_2, integrating features from both, while users can choose between features with semantic dependencies–similar to resolving merge conflicts in code. (e.g., epidemiologists can choose to either study the entire infectious group or focus on varying levels of infection severity based on symptoms.)

	SER	V	EQ	Inf	I_{sym}	I_{iso}	D	H
m_1	X	X	X	X				
m_2	X	X			X		X	
m_3	X					X	X	X
m_{12}	X	X	X	X	X		X	
m_{13}	X	X	X	X		X	X	X
m_{23}	X	X			X	X	X	X
m_{123}	X	X	X	X	X	X	X	X

Table 2: Formal context mapping models to clusters.

One challenge **(C-6)** is selecting the right merging algorithm, such as superimposition [6] or n-way merging [28] and potentially adapting it for compartmental models. Here, We illustrate this with a simple approach that concatenates clusters, representing alternative dependencies with an 'or' condition. In the model m_{12}, the clusters {SER, V, EQ} of m_1 are combined with the clusters {SER, V, I_{sym}, D} from m_2 to get the clusters {SER, V, EQ, [INF | I_{sym}], D}, where we show alternative clusters using brackets and the "|" symbol. We also get:

- m_{23} with clusters {SER, [I_{sym} | I_{iso}], H, V, D};
- m_{13} with {SER, V, EQ, [Inf | I_{iso}], H, D}; and
- m_{123} with {SER, V, EQ, [Inf | I_{sym} | I_{iso}], H, D}.

This exercise highlights the challenge **(C-7)** to provide appropriate metamodelling support for the merged variants. In our example, we use a simple partial modelling approach [14] that retains the information about alternative clusters within the model itself. Other metamodel relaxation approaches could include those used for model families [3] or for model transformation languages [30].

Yet another challenge **(C-8)** is the strategy for enriching the input corpus with merged models. A brute force approach of generating everything will risk a combinatorial explosion as the corpus size increases. A potential solution is to generate a limited number of models on demand as the modeller explores the combinations.

Creating a Concept Lattice. By identifying interesting modelling ideas and their dependencies in the input corpus and generating multiple combinations of them, we have created a large design space. A key challenge **(C-9)** is structuring this space for querying, exploration, and drawing conclusions. One promising approach is to use Formal Concept Analysis (FCA), a mathematical framework for data analysis, information management, and knowledge representation [17] that classifies objects by their shared attributes. FCA has been applied to uncover relationships in various domains, including software product lines [10] and code snippets [15].

FCA takes as input a *formal context* $K = (O, A)$, where O is a set of objects (in our case, compartmental models) and A is a set of attributes (in our case, clusters) that characterize each object. We set O to be the three initial models m_1, m_2, m_3 and the merged models $m_{12}, m_{13}, m_{23}, m_{123}$, while A is the set of clusters in Table 1. We show Carol's formal context in Table 2, where "X" signifies an object has the corresponding attribute.

The output of FCA is a *concept lattice*, like the one in Figure 5, generated using Latviz [1] for Carol's formal context. Each numbered node, called a *formal concept*, represents a maximal set of objects (in black) that share a maximal set of attributes (in blue). This means that the models in a node are the only ones sharing all the attributes of that node, and the attributes in the node are the only ones shared by all the models of that node. This characteristic provides hierarchical relationships and commonalities among the

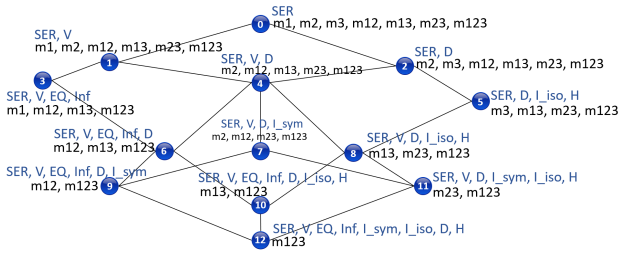


Figure 5: Concept lattice of the formal context in Table 2.

models, allowing Carol to navigate between them based on features and view them in different levels of abstraction.

The top node of the lattice (#0) contains the compartments SER, common to all the models in O. Visiting its sub-concepts refines the set of models, adding more specificity. For example, moving to node #1 also adds V, thus excluding model m_3. Alternatively, moving from #0 to #2 adds D, excluding model m_1. Node #4 is the common sub-concept of #1 and #2, containing the union {SER, V, D} of their attributes and the intersection of their models, $O - \{m_1, m_3\}$. From #4, we can navigate to #8 to include I_{iso} and H, thus narrowing the set of models. Conversely, if we want to make our model set more general, we can move from #8 to #5 by excluding V, and only view the models that have the features SER, D, I_{iso}, and H. In other words, starting from any concept in the lattice, the structure allows us to include more attributes/narrow the set of models by navigating to a sub-concept. Conversely, we can navigate to a super-concept to remove attributes and explore a broader, less specific set of models.

One challenge **(C-10)** is the scalability and incrementality of the lattice creation algorithm, which domain-specific optimizations might improve. Another challenge **(C-11)** is that the lattice does not include all possible feature combinations; they are distributed based on their occurrence patterns in the input corpus, limiting customizability. To address this, we are exploring pattern structures [16], an FCA extension that uses similarity operators to enable more flexible and tailored feature combinations.

Concept Lattice Exploration. The concept lattice supports an interactive workflow for visually exploring different feature combinations. We discuss the envisioned functionalities of this tool-supported workflow.

Starting point: The first challenge **(C-12)** is picking a good starting point for exploration, like the top node of the lattice, a node closest to their base model, or one with desired quality attributes.

Navigation: At each step, users are dynamically offered a list of features (e.g., V, D, ...). Selecting one navigates to the corresponding lattice node, displaying candidate models. Users can navigate back and forth, adding or removing features to see how model recommendations change.

Output: Once a desired combination of features is found, users can export a model as a partially completed prototype (e.g., Carol's m_p). If not already generated (cf. challenge **(C-8)**), the tool then produces the model by merging the desired feature fragments.

To ensure the ongoing evolution of the prototype model, we propose linking it to the input variant models by storing them in a version control system **(C-13)** . The prototype can be treated as a dependent branch or submodule, allowing seamless integration of

updates from the variant models and ensuring the prototype stays in sync with the latest changes.

3 RELATED WORK

Although no reuse methodology exists for epidemiological models, software engineering techniques have been applied to scientific domains [24]. For example, a modeling framework that links domain knowledge to code implementation accelerates compartmental model development [25]. Curzi-Laliberté et al. [12] use model-driven engineering and a graphic domain-specific language to create and simulate compartmental models. A notable feature of their approach is the grouping of semantically related compartments to create layers of abstraction.

The challenges we face are closely related to fork-based development, where different software variants evolve independently, and changes in forks can be proposed for integration into the parent repository through a pull request [20]. Reusing fragments from multiple models is similar to reusing features from forks. Zhou et al. [33] identify features in forks with clustering and labeling. Identifying redundancies in fork-based development [27] and providing an overview of what exists in the whole community prevents developers from developing a feature already existing in previous work. Non-systematic reuse by cloning and owning a repository is a well known ad-hoc technique, that faces various inefficiency challenges in fork-based development [34] and generally [22].

Design space exploration (DSE) entails identifying optimal design solutions from a range of potential options to meet specific requirements and has been used to manage design uncertainty and conflicting objectives [11]. DSE has been extensively studied in different application objectives, use cases, targeted software systems, and application domains [2]. Mjeda et al. [26] studied design exploration and analysis in the context of software product lines (SPLs). In turn, feature-oriented SPLs aim at systematizing code reuse at a high level of abstraction [5][7]. Our vision draws from ideas in DSE and SPLs; a key difference is that (a) we do not assume any kind of existing reuse infrastructure, and (b) the end result of our proposed approach is an incomplete prototype, rather than a fully configured and optimised product.

4 CONCLUSION

We outlined a vision for the rapid development of partially completed prototype compartmental models, by reusing modelling ideas from a given input corpus. The vision entails identifying and clustering ideas of interest from other relevant models and their dependencies, merging useful fragments to produce a design space, and organizing this design space in a concept lattice, allowing interactive exploration. We pointed out 13 challenges and discussed preliminary approaches.

Acknowledgements: This work was supported by a grant from the Wellcome Trust [226107/Z/22/Z].

REFERENCES

[1] M. Alam, T.N. Nguyen Le, and A. Napoli. 2016. Latviz: A new practical tool for performing interactive exploration over concept lattices. In *Proc. of CLA'16*.
[2] J. Alves Pereira, M. Acher, J. Martin, J-M Jézéquel, G. Botterweck, and A. Ventresque. 2021. Learning software configuration spaces: A systematic literature review. *J. of Systems and Software* 182 (2021), 111044.
[3] S. Alwidian and D. Amyot. 2017. Relaxing metamodels for model family support. In *Proc. of ME'17*, Vol. 2019. 60–64.
[4] G. Antoniol and Y-G Guéhéneuc. 2006. Feature identification: An epidemiological metaphor. *IEEE TSE* 32, 9 (2006), 627–641.
[5] S. Apel, D. Batory, C. Kästner, and G. Saake. 2013. Feature-oriented software product lines. (2013).
[6] S. Apel and C. Lengauer. 2008. Superimposition: A language-independent approach to software composition. In *Proc. of ICST'08*. Springer, 20–35.
[7] T. Berger, D. Lettner, J. Rubin, P. Grünbacher, A. Silva, M. Becker, M. Chechik, and K. Czarnecki. 2015. What is a feature? a qualitative study of features in industrial software product lines. In *Proc. of SPLC'15*. 16–25.
[8] A. Boin, M. Lodge, and M. Luesink. 2020. Learning from the COVID-19 crisis: an initial analysis of national responses. *Policy Design and Practice* 3, 3 (2020), 189–204.
[9] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. 2006. A manifesto for model merging. In *Proc. of GIMM'06*. 5–12.
[10] J. Carbonnel, M. Huchard, and C. Nebut. 2019. Towards complex product line variability modelling: Mining relationships from non-boolean descriptions. *J. of Systems and Software* 156 (2019), 341–360.
[11] J. M. P. Cardoso, J. G. de Figueiredo Coutinho, and P. C Diniz. 2017. *Embedded computing for high performance: Efficient mapping of computations using customization, code transformations and compilation.* Morgan Kaufmann.
[12] B. Curzi-Laliberté, M. Fokaefs, M. Famelis, and M. Hamdaqa. 2024. EpiMDE: A Model Driven Engineering Platform for Epidemiological Modeling. In *Proc. of MODELS 2024*. to appear.
[13] C. Ding, X. Liu, and S. Yang. 2021. The value of infectious disease modeling and trend assessment: a public health perspective. *Expert review of anti-infective therapy* 19, 9 (2021), 1135–1145.
[14] M. Famelis, R. Salay, and M. Chechik. 2012. Partial models: Towards modeling and reasoning with uncertainty. In *Proc. of ICSE'12*. IEEE, 573–583.
[15] J. Galasso, M. Famelis, and H. Sahraoui. 2022. Fine-Grained Analysis of Similar Code Snippets. In *Proc. of ICSR'22*. Springer, 3–21.
[16] B. Ganter and S. Kuznetsov. 2001. Pattern structures and their projections. In *Proc. of ICCS*. Springer, 129–142.
[17] B. Ganter and R. Wille. 2012. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media.
[18] E. Griffiths, D. Dooley, M. Graham, G. Van Domselaar, F. Brinkman, and W. Hsiao. 2017. Context is everything: harmonization of critical food microbiology descriptors and metadata for improved food safety and surveillance. *Frontiers in Microbiology* 8 (2017), 1068.
[19] A. B Gumel, E. A Iboi, C. N Ngonghala, and E. H Elbasha. 2021. A primer on using mathematics to understand COVID-19 dynamics: Modeling, analysis and simulations. *Infectious Disease Modelling* 6 (2021), 148–168.
[20] M. Hadian, S. Brisson, B. Adams, S. Ghari, E. Noei, M. Fokaefs, K. Lyons, and S. Zhou. 2022. Exploring trends and practices of forks in open-source software repositories. In *Proc. of CASCON '22*. IBM Corp., 120–129.
[21] Charles W. Krueger. 1992. Software reuse. *ACM Comput. Surv.* 24, 2 (jun 1992), 131–183. https://doi.org/10.1145/130844.130856
[22] J. Krüger and T. Berger. 2020. An empirical analysis of the costs of clone- and platform-oriented software reuse. In *Proc. of ESEC/FSE '20*. 432–444.
[23] E. Kuhl. 2021. *Computational Epidemiology: Data-Driven Modeling of COVID-19*. Springer.
[24] D. Leroy, J. Sallou, J. Bourcier, and B. Combemale. 2021. When Scientific Software Meets Software Engineering. *Computer* 54, 12 (2021), 60–71.
[25] S. Libkind, A. Baas, M. Halter, E. Patterson, and J. P Fairbanks. 2022. An algebraic framework for structured epidemic modelling. *Phil. Trans. of the Royal Society* 380, 2233 (2022), 20210309.
[26] A. Mjeda, A. Wasala, and G. Botterweck. 2017. Decision spaces in product lines, decision analysis, and design exploration: an interdisciplinary exploratory study. In *Proc. of VaMoS '17*. 68–75.
[27] L. Ren, S. Zhou, C. Kästner, and A. Wąsowski. 2019. Identifying Redundancies in Fork-based Development. In *Proc. of SANER'19*. 230–241.
[28] J. Rubin and M. Chechik. 2013. N-way model merging. In *Proc. of FSE'13*. 301–311.
[29] J. Rubin, K. Czarnecki, and M. Chechik. 2013. Managing cloned variants: a framework and experience. In *Proc. of SPLC'13*. 101–110.
[30] E. Syriani, J. Gray, and H. Vangheluwe. 2013. Modeling a model transformation language. *Domain Engineering: Product Lines, Languages, and Conceptual Models* (2013), 211–237.
[31] A. R Tuite, D. N Fisman, and A. L Greer. 2020. Mathematical modelling of COVID-19 transmission and mitigation strategies in the population of Ontario, Canada. *Cmaj* 192, 19 (2020), E497–E505.
[32] C. Von Csefalvay. 2023. *Computational Modeling of Infectious Disease: With Applications in Python*. Elsevier.
[33] S. Zhou, Ş. Stănciulescu, O. Leßenich, Y. Xiong, A. Wąsowski, and C. Kästner. 2018. Identifying features in forks. In *Proc. of ICSE'18* (Gothenburg, Sweden). 105–116.
[34] S. Zhou, B. Vasilescu, and C. Kästner. 2019. What the fork: a study of inefficient and efficient forking practices in social coding. In *Proc. of ESEC/FSE 2019*. 350–361.