Requirements Engineering for Safety-Critical Molecular Programs

Robyn R. Lutz
Department of Computer Science, Iowa State University, Ames, IA USA
Email: rlutz@iastate.edu ORCID: 0000-0001-5390-7982

***Abstract*—The field of cyber-molecular systems is growing rapidly. In these nanotechnology applications the computational logic is encoded by developers into the molecules themselves. Many planned applications are safety-critical, including bio-compatible sensors, pollution trackers, and targeted drug-delivery devices. Requirements engineering (RE) activities and artifacts are essential to assuring the safety of molecular programs. However, molecular programmed devices offer challenges to traditional RE activities. Molecular programmed systems are nanoscale, so hard to monitor; execute at scale, typically 10^{10} devices in solution at once; and have probabilistic behavior. Toward safe molecular programs, we propose a new framework, RE4DNA, for their safety requirements discovery, specification, and verification. Its contribution is to bridge the cyber and the molecular in the requirements engineering process. Further, use of RE4DNA identifies building blocks that can contribute to a preliminary safety case. In this paper we introduce RE4DNA, describe how it handles some particular challenges of molecular programming, illustrate its use on a benchmark molecular program, and discuss future work.**

***Index Terms*—Requirements engineering, molecular programming, safety case**

I. Introduction

Tun et al. advised at RE'19 that requirements engineers "look out" more into the physical world. [29]. The emerging field of molecular programmed systems, also called cyber-molecular systems, bridges the computational and physical worlds. This field currently stands to benefit greatly from increased attention of requirements engineers to its particular challenges. Many of the planned applications are safety-critical. In this RE@Next! paper, we propose a new framework to bridge the cyber and the molecular in the requirements engineering (RE) of safety-critical molecular programs.

A striking advantage of molecular systems is that they are bio-compatible. Many will be designed to work *in vivo* (in the body). The public must be able to trust these systems. Our proposed framework is offered in anticipation that many molecular programmed systems will need to be certified either internally, or externally by a government agency. There is a thus a need for safety cases to be created for programmed molecular systems. While it cannot yet be known precisely what that regulatory landscape will look like [1], [8], it is incumbent on us to begin planning what belongs in the safety case. There is evidence that the incremental development of the safety case alongside the development of the system, starting in the requirements phase, yields better results and is preferred by regulators [15], [22].

The RE4DNA framework proposed here supports safety-aware requirements discovery, specification, and verification of molecular programs. This differs from most previous work on software engineering of molecular programs by focusing specifically on the *requirements engineering phase of computational programs implemented in DNA molecules.* We propose that a structured requirements engineering process can improve molecular program quality and safety, as well as significantly reduce costly design-and-test cycles in the laboratory. Moreover, for safety-critical molecular programs, analysis results from the RE4DNA activities can provide essential building blocks toward composing preliminary safety arguments [12].

Requirements engineering activities and artifacts are essential to assuring the safety of molecular programs. However, a molecular programmed system offers several challenges to traditional RE activities: (1) probabilistic behavior, inherent in its nanoscale physical implementation and operation in a molecular environment; (2) scalability to very large numbers (over 10^{10}) concurrently executing components; and (3) the need to use a shared, transdisciplinary model to communicate requirements specifications.

In RE4DNA we aim to provide a customization and extension of existing RE tools and techniques to handle these particular challenges of molecular programming. Toward this objective, we here report initial results from investigation of three research questions:

RQ1: What current RE techniques support requirements engineering for a safety-critical molecular programmed system?

RQ2: What extensions and domain-specific specializations of current RE techniques assist that effort?

RQ3: What RE4DNA artifacts serve as building blocks for a preliminary safety argument?

The contributions of this RE@Next! paper are fourfold:

1) It proposes a new framework, RE4DNA, to bridge the cyber and the molecular in the requirements engineering of safety-critical molecular programs.
2) It shows how RE4DNA handles some particular challenges of RE for molecular programming and illustrates its use on the requirements development for a small, safety-critical molecular programmed device.

2332-6441/22/$31.00 ©2022 IEEE
DOI 10.1109/RE54965.2022.00045

3) It demonstrates how RE4DNA identifies building blocks that can contribute to a preliminary safety case.
4) It outlines a research plan for needed future work on RE4DNA and suggests its potential for broader reuse in the RE of other very large, highly distributed, stochastic systems.

The paper is organized as follows. Sect. II describes background and related work. Sect. III describes our approach in RE4DNA, and Sect. IV reports from its application on an example molecular program. Sect. V concludes with research directions for future work by ourselves and others.

II. Background and Related Work

Molecular programming is transdisciplinary in that it is at the intersection of requirements engineering, computer science, mathematics, chemistry, molecular biology and physics.

Molecular programmed systems. Molecular programming uses the inherent information and computational capabilities of DNA to create programmable molecular devices and structures that self-assemble. These DNA nanotechnology systems are dynamic and able to respond to changes in their environment by changing their behavior. They are programmed by the developers' careful selection of the DNA strands that will achieve the system's functional and nonfunctional goals.

A simple example is a molecular (DNA) biosensor programmed to detect a target molecule of concern, such as a specific pollutant or tumor marker [34]. When the DNA molecular strands selected to implement the biosensor are combined in solution, they self-assemble into an open-V, tweezers-like shape that, if it encounters the target molecule of concern, closes to trap it. The change in shape is externally visible via microscopy or fluorescence, indicating whether or not the biosensor has detected the presence of the target molecule.

Stochastic chemical reaction networks (CRNs) are often used in molecular programming to specify the program requirements [10], [26]. The stochastic CRN model is Turing universal [25]. A CRN represents the program's behavior as a set of reactions over a set of abstract molecular species (here, abstract DNA strands). For example, the reaction $X + Y \xrightarrow{k} 2Z + S$ specifies that, when an X molecule collides with a Y molecule, they are consumed and two Z molecules and an S molecule are produced as a result, where k is the rate constant at which this reaction occurs. A CRN's state is a vector specifying the number of each molecular species present, and the CRN's execution is a continuous time Markov process executing at a rate determined by its rate constants [3].

RE for molecular programmed systems. While there have been significant recent advances in the verification of CRN specifications, there has been only limited attention to requirements discovery for molecular programs. Related work on specifying goals in uncertain environments has formulated failure patterns [28] and formalized the required probability of goal satisfaction [5]. However, molecular systems may have such a large number of individual components in solution that failures with any significant probability almost certainly will occur in many individual components. Probabilistic model checking has been used to verify molecular programs, including DNA nanorobots [11], [13], [18], [19]. Satisfiability Modulo Theories (SMT) also have been used to analyze DNA computing [33]. Alloy has been used to model a class of small CRNs relevant to synthetic biology [31]. Simulation and testing of CRNs offer additional approaches to verification of CRNs [14].

Early composition of safety arguments. Composing a safety argument is core to any safety case and thus has been widely studied [4], [16], [17], [23], [24]. Assurance cases for synthetic biology already have been developed [9]. Beginning this effort early in the development of a new safety-critical product is recommended and motivates our work here [22], [32]. The process of composing a safety case also continues throughout development. For example, laboratory experiments and, for some systems, clinical trials, will be essential, later sources of evidence that safety goals of a molecular program are met. However, in this paper we focus on the requirements and the contributions that the RE4DNA activities can make to the early safety argument.

III. RE4DNA

This section describes RE4DNA, our proposed framework to bridge the cyber and the molecular in the requirements engineering of safety-critical molecular programs. We provide an illustrative example of its use in Section IV. In RE4DNA we aim to provide a customization and extension of existing RE tools and techniques to handle the particular needs of molecular programming, specifically, its stochasticity, inherent in its physical implementation and operation in a molecular environment; its scalability to very large numbers of concurrently executing devices; and its transdisciplinary communication needs. Although each of these needs is not unique to molecular programming, the combination creates some particular difficulties in requirements discovery, specification and verification for these systems. Additionally, RE4DNA provides the building blocks and traceability needed toward early, incremental safety arguments.

RE4DNA uses an underlying Traceability Information Model (TIM), shown in Figure 1, that defines the artifact types, permitted trace types, and the traceability paths. Such a TIM is fairly standard in the development of safety-critical systems [7], [22], [27] and especially useful for multi-disciplinary teams of developers, as occurs with molecular programs.

A. Requirements discovery.

Intent discovery. The system-to-be and its purpose are initially described informally in this step, often in multiple cross-disciplinary discussions. We have found that cross-trained students with knowledge of both molecular chemistry and computer science are valuable assets in facilitating these discussions and providing needed contextual information.

Goal modeling. Goal modeling is used in RE4DNA as an effective technique to refine, represent and jointly review the requirements for a new molecular program [30]. A molecular

program's probabilistic behavior both enables and constrains its computation. An advantage of goal modeling is that it supports capture of the domain properties and environmental assumptions simultaneously with the goals. Determination of which failures are feasible in a molecular environment and what their effects will be is strongly rooted in the domain properties (e.g., that there is a finite amount of each of the molecules used) and of environmental assumptions (e.g., that the solution is well-mixed, and that a signal requires a threshold number of signal molecules to be present). The safety requirements that will need to be shown to be satisfied in the planned system are then specified in Continuous Stochastic Logic (CSL) in RE4DNA to enable their subsequent formal verification.

Safety analysis. Early analysis of risks at the goal level, based on the Identify/Assess/Control approach in [5], [30], helps discover which obstacles to achievement of goals can occur in the molecular environment. For safety-critical molecular programs [21], our framework employs fault tree analysis to identify, assess, and mitigate risks to achieving safety requirements. Use of fault tree analysis helps communication as well. We have found that molecular biologists are familiar with fault trees because they use similar diagnostic reasoning in their lab notebooks.

B. Requirements specification.

The operationalized behavioral requirements for the molecular program are specified by assigning the goals to the specific molecular agents (the abstract DNA strands) responsible for them [30]. We express these program requirements as a chemical reaction network (CRN), a widely used model for molecular programming described in Section II. A CRN defines the rules of interaction for the molecular agents.

CRN specification of the requirements serves as an effective communication bridge across the disciplines involved in molecular programming. We have found CRNs useful for both communicating and reviewing the behavioral requirements. Requirements review by stakeholders representing the diversity of fields from software engineers to the laboratory scientists who will implement the requirements creates more robust molecular programs and finds problems earlier. Finding requirements errors, omissions and physical infeasibilities early will reduce the design-and-test cycles in the laboratory [28]. Moreover, specification of the requirements as a CRN enables automated verification and compilation into concrete DNA strands for implementation.

C. Requirements verification.

Requirements verification of molecular programs faces challenges of scalability. With requirements verification we seek evidence that we have "gotten the requirements right" in the specification. However, there are usually very many devices (copies of the system) operating in parallel in the solution. To claim that the safety requirements hold across a range of molecular population counts, diversity of evidence is needed [20]. Use of CSL to specify the safety requirements supports probabilistic model checking in the PRISM tool [18], [19] for small populations (in the hundreds) and simulation using Ordinary Differential Equations in MatLab's SimBiology tool for very large populations.

D. Requirements evolution.

Due to the complexities of probabilistic behavior at scale (i.e., the need for safe and robust behavior from inherently unreliable molecular devices), and the fact that this is an emerging field, changes to requirements, assumptions, and domain properties occur frequently. Many changes result from new domain knowledge in theoretical (e.g., improved modeling of energy wells) or experimental (e.g., more automated microfluidic laboratory equipment) advances. Many changes also are perfective, i.e., new opportunities for improved performance (e.g., reduced molecular leaks). These changes also can cause emergent risks that need to be discovered and mitigated by new or modified requirements. Additionally, argument structures and their building blocks must be updated accordingly [2]. RE4DNA is designed to support the incremental, evolving requirements engineering needed for molecular programs; however, much work remains, especially in terms of automated traceability and safe reuse of requirements, risk and argument patterns.

E. Traceability to safety argument building blocks

Safety cases should be built incrementally as development progresses [22], [32]. RE4DNA assists by identifying building blocks that contribute to a preliminary safety argument that the safety requirements, together with the specified domain properties and environmental assumptions, are satisfied by the CRN specification. The argument thus documents the reasoning about safety. This safety-aware focus in RE4DNA responds to the need for better integration of safety arguments into the development of safety-critical molecular programs.

The TIM in Figure 1 shows the trace links available in RE4DNA for achieving this safety-focused traceability from its artifacts to the safety argument building blocks and relations. The elements shown in square brackets in the artifacts in Fig. 1 correspond to the partial safety argument composed in the illustrative example in Fig. 4. The safety argument uses the core Goal Structuring Notation (GSN) v.3 [23] and its six core safety argument building blocks. These are: Goal (safety claim, expressed as a proposition), Context (operating environment), Assumption (made in the argument), Strategy (how a higher-level goal is inferred from lower-level goals), Justification (rationale), and Solution (composition of evidence) [17]. We maintain consistency with the description of preliminary safety arguments in [12] to facilitate potential future use of their AdVoCATE assurance case toolset.

IV. Application

RE4DNA is a framework that will provide a clear and coherent structuring of the requirements engineering process that others can use to develop their own, new molecular programs. This paper shows how the RE4DNA framework can

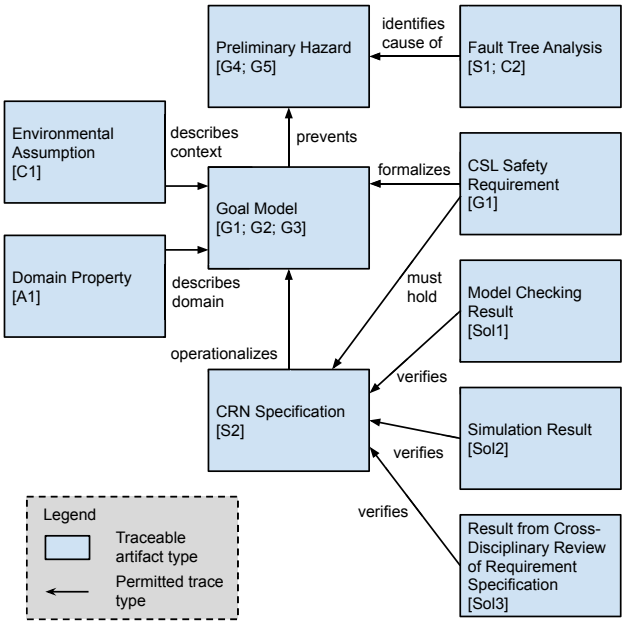

Fig. 1. The Traceability Information Model (TIM) for the artifacts and associations created and used in the RE4DNA framework are displayed in the figure. The annotations in square brackets trace forward to the safety-argument building blocks derived from each artifact type, and map to the example molecular program whose development is described in Section IV.

be used to systematize and promote more fully what needs to be done during RE to justifiably rely on a safety-critical molecular program.

We illustrate RE4DNA by describing its use on an example molecular system. In prior work we developed a molecular oscillator [13], and we use it here as our illustrative example. Oscillators have been used in molecular programming as benchmarks [6], [26], in part due to their ubiquity in both natural (e.g., gene regulatory networks) and synthetic (molecular programming) biology. In many cases oscillators are safety-critical.

Some artifacts used by RE4DNA in our example appear in [13], including formal requirements, and CRN simulation and model checking results. The RE4DNA framework is imposed retrospectively on those artifacts toward evaluating its feasibility. Other artifacts used by RE4DNA in our example are newly created for this paper, including the TIM and the safety argument. These new artifacts add traceability and formally tie the safety requirements to the safety case. While prior work did not integrate these fully into the RE process, RE4DNA incorporates an updated and improved understanding of their essential roles.

A. Requirements discovery

Intent discovery. The intent of our molecular system-to-be was initially described informally in discussions among multi-disciplinary stakeholders. We proposed to develop a programmed molecular component to self-monitor and report its health status. Our innovation in [13] was to add an output capability to a standard oscillator such that it regularly sends

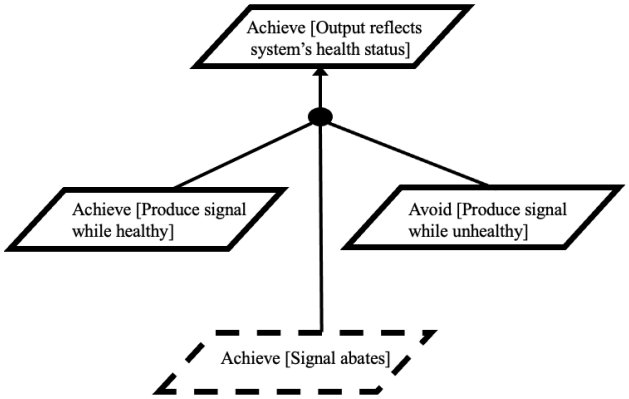

Fig. 2. Incremental Goal Model software artifact developed for the molecular oscillator during requirements discovery [13], [30]. The solid lines represent the original goal model. The dashed lines represent an additional, molecular domain-specific goal subsequently identified during safety analysis.

an "I'm OK" signal to adjacent systems if it is healthy. However, if the oscillator fails, the signal must stop. The absence of a signal for a period of time indicates to adjacent systems that the oscillator has failed. It is safety-critical because other adjacent molecular components, some of which are safety-critical, will depend on it.

Goal modeling. Figure 2 shows part of the Goal Model for the self-monitoring molecular oscillator. The top-level goal is that the presence or absence of an "I'm OK" health signal reflect its actual health status. As shown in Fig. 2, the top-level goal is then initially refined into the AND of two subgoals: ACHIEVE [Produce signal while healthy] and AVOID [Produce signal while unhealthy]. These subgoals are subsequently formalized as safety requirements in Continuous Stochastic Logic (CSL) for use in formal reasoning. For example, the first subgoal above is formalized in CSL as $\mathcal{P}_{\geq 1}[\Box(\text{healthy} \implies \mathcal{P}_{\geq 1-\delta_1}[\Diamond_{\leq t_1}((\text{sigHigh} \vee \neg\text{healthy})])]$.

Safety analysis. Following the Identify/Assess/Control obstacle analysis steps, in the Identify step RE4DNA uses fault-tree analysis [21] to identify two scenarios that cause the oscillator to fail. In the first safety scenario, the oscillator fails because the component runs out of one of the three molecules that drive its phased oscillations. In the second safety scenario, the oscillator fails because the count of each of the three molecules converges toward equilibrium, disrupting the oscillations. In the Assessment step these are both found to be feasible and likely obstacles to achieving the goals represented in the Goal Model. In the Control step, the needed mitigation adds a *newly discovered safety subgoal* to the Goal Model. This captures the molecular reality that stopping a molecular signal requires not only stopping the production of signal molecules but also actively removing (consuming) them. This new goal is represented in Fig. 2 as the subgoal Achieve [Signal abates] in the dashed box.

B. Requirements specification

Specification of the behavioral requirements as a chemical reaction network (CRN) model is used by RE4DNA to

operationalize them and to enable their automated verification. Operationalization assigns the goals and environmental assumptions to specific molecular agents responsible for them.

The CRN below specifies the three phases of the oscillator and its "I'm OK" signal, S. The initial condition is that the count of only one of the three molecules, X, is high (i.e., it starts in Phase X), and the count of Y and Z are similar and low. Thus, the first and third reactions are equally likely to occur. When either occurs, it increases the rate of the first reaction and decreases the rate of the third reaction. This goes on until the count of Y is highest (i.e., it enters Phase Y). Similarly, the oscillator continues on to Phase Z. The count of S molecules increases in the first reaction, and decreases (abates) in the last two reactions.

$$X + Y \xrightarrow{k} 2Y + S$$
$$Y + Z \xrightarrow{k} 2Z$$
$$Z + X \xrightarrow{k} 2X$$
$$Z + S \xrightarrow{k_2} Z$$
$$S \xrightarrow{k_2} \emptyset$$

C. Requirements verification

As noted in Sect. II, CRNs are an increasingly common lingua franca in which to specify a molecular program's requirements. In RE4DNA the CRN serves as our chosen way to communicate the molecular requirements specifications, via a shared, transdisciplinary model. This also enables the CRN to be input to existing requirements verification tools, including simulation, model checking, proof assistants, and model testing. To analyze and verify the oscillator's requirements we used two of these toolsets. The first was MATLAB's SimBiology package. Figure 3 shows some simulation results that were part of our verification that the CRN specification for the oscillator satisfies the subgoal ACHIEVE [Produce signal while healthy]. The second was the probabilistic model checker, PRISM [18], [19]. PRISM checked that the CRN specification satisfied the formal CSL safety requirements.

D. Requirements evolution.

We have described the RE4DNA framework in a sequential manner. However, the activities were often performed iteratively and incrementally. For example, simulation was done repeatedly as the goal model and CRN specification evolved both to correct errors and to improve the oscillator. Requirements engineering was complicated by the fact that the oscillator is a molecular system. For example, the "OK" signal itself is molecular, meaning that the oscillator must verifiably produce a sufficient count of signal molecules while it is healthy. Similarly, although the signal molecules will never entirely disappear, their count must verifiably abate over time when the oscillator has failed. The probabilistic molecular environment often defeated intuition or enforced de-idealizations. This complexity, even for a seemingly simple system such as the oscillator, encouraged us to continue integrating formal requirements verification into RE4DNA. Formal reasoning to detect when goals were infeasible and what environmental assumptions were missing from the specification was very useful in getting the requirements right.

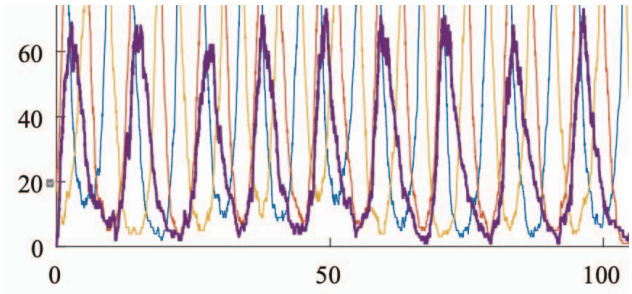

Fig. 3. Simulation with MATLAB's SimBiology is used by RE4DNA to verify that the requirements specification for the oscillator meets its goals. Purple peaks in bold font show the regular issuance of the "OK" signal when the oscillator is healthy. The other three colors show the count of X, Y and Z molecules, respectively, in the oscillator's three phases. The X axis is simulation time in seconds; the Y axis is total molecular count [13].

E. Traceability to safety argument building blocks.

Figure 4 shows how the artifacts created by the RE4DNA activities can form part of the safety argument that the top-level goal Achieve [Output reflects system's health status] is satisfied. The unique IDs on the node labels provide traceability between the building blocks in the argument and the artifacts in the TIM in Figure 1. The mapping here was done manually but can be automated in the future, perhaps using [12]. Additionally, tooling might support broader potential use of RE4DNA's approach for non-molecular, highly distributed networks with probabilistic behavior that are safety-critical.

F. Discussion

This RE@Next! paper reports early results in our ongoing investigation of three research questions:

RQ1: What current RE techniques support requirements engineering for a safety-critical molecular programmed system? We have shown that goal modeling, safety analysis (specifically, fault tree analysis, which is familiar to lab scientists), and formal specification using chemical reaction network modeling as a standard bridge to a variety of automated requirements verification tools (model checking, simulation) can assist in addressing the challenges of probabilistic behavior, scalability and transdisciplinary stakeholders.

RQ2: What extensions and domain-specific specializations of current RE techniques assist that effort? We have described the primacy of accurate domain properties and environmental assumptions in creating safe and robust molecular programs, suggested the need for additional domain-specific obstacle patterns to address the particularities of the molecular environment (specifically, signal abatement and signal leakage, as well as the certainty that at this very large scale some molecular components will fail), and noted that rigorous requirements analysis can reduce costly design-and-test cycles in the laboratory.

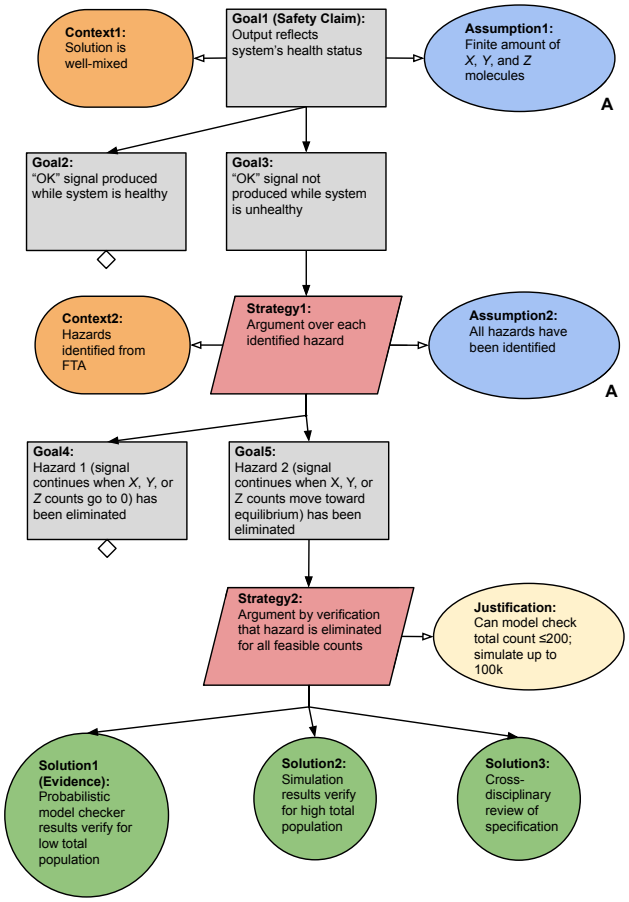

Fig. 4. A portion of the safety argument for the self-monitoring oscillator, showing the goals, strategies, solutions, and associated building blocks generated as RE4DNA artifacts. The unique labels on the nodes trace to the TIM in Figure 1

.

RQ3: What RE4DNA artifacts serve as building blocks for a preliminary safety argument? We have proposed a TIM for RE4DNA and traced artifacts from the RE4DNA activities to the structural elements and relations needed for a preliminary safety argument.

V. Directions for Future Work and Conclusion

This paper has proposed a framework for the requirements engineering of safety-critical molecular programs that incorporates best-practice RE techniques and tools while addressing some particular challenges of the cyber-molecular domain. Additionally, our framework uses the artifacts produced by its requirements engineering activities to identify building blocks for safety arguments toward a preliminary safety case.

The long-term objective of our work is to provide a requirements engineering structure and guide that is useful and used by both molecular programming research and industrial groups when developing new molecular programmed systems. We are especially concerned with providing support for safety-critical applications, such as DNA drug-delivery capsules and biosensors. There is much future work to do toward this. Next steps, many of which offer opportunities for new collaborative work, include identifying how best to specify the domain properties and assumptions for ease of comprehension by all stakeholders (software engineers to lab scientists); offering domain-tailored display options; creating additional obstacle patterns from recurring molecular program difficulties; incorporating rationales into the safety argument; and automating the composition and update of safety arguments as RE artifacts evolve.

VI. Acknowledgment

This work is supported in part by NSF Grant FET #1900716.

References

[1] *ANSI/AAMI/IEC 62304:2006/A1:2016 Medical Device Software*, 2016.

[2] A. Agrawal, S. Khoshmanesh, M. Vierhauser, M. Rahimi, J. Cleland-Huang, and R. R. Lutz, "Leveraging artifact trees to evolve and reuse safety cases," in *Proc. 41st Int'l Conf on Software Engineering*. IEEE/ACM, 2019, pp. 1222–1233.

[3] D. F. Anderson and T. G. Kurtz, "Continuous time Markov chain models for chemical reaction networks," in *Design and Analysis of Biomolecular Circuits*, H. Koeppl et al., Ed. Springer, 2011, pp. 3–42.

[4] R. Bloomfield and J. Rushby, "Assurance 2.0: A manifesto," in *Proc. 29th Safety-Critical Systems Symposium*, 2021, pp. 85–108.

[5] A. Cailliau and A. van Lamsweerde, "Assessing requirements-related risks through probabilistic goals and obstacles," *Requirements Engineering*, vol. 18, no. 2, pp. 129–146, 2013.

[6] L. Cardelli, "Artificial biochemistry," in *Algorithmic Bioprocesses*, ser. Natural Computing Series, A. Condon, D. Harel, J. N. Kok, A. Salomaa, and E. Winfree, Eds. Springer, 2009, pp. 429–462.

[7] J. Cleland-Huang, M. P. E. Heimdahl, J. H. Hayes, R. R. Lutz, and P. Maeder, "Trace queries for safety requirements in high assurance systems," in *Proc. 18th REFSQ*, ser. LNCS, vol. 7195, 2012, pp. 179–193.

[8] K. Cobbaert and G. Box, *Software as a Medical Device, Regulatory and Market Access Implications*. RAPS Publications, 2021.

[9] M. B. Cohen, J. Firestone, and M. Pierobon, "The assurance timeline: Building assurance cases for synthetic biology," in *Proc. SAFECOMP*, 2016, pp. 75–86.

[10] M. Cook, D. Soloveichik, E. Winfree, and J. Bruck, "Programmability of chemical reaction networks," in *Algorithmic Bioprocesses*, A. Condon et al., Ed. Springer, 2009, pp. 543–584.

[11] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield, "DNA walker circuits: Computational potential, design, and verification," in *Proc. 19th Int'l Conf on DNA Computing and Molecular Programming*, ser. LNCS, vol. 8141. Springer, 2013, pp. 31–45.

[12] E. Denney and G. Pai, "Tool support for assurance case development," *Autom. Softw. Eng.*, vol. 25, no. 3, pp. 435–499, 2018.

[13] S. J. Ellis, T. H. Klinge, J. I. Lathrop, J. H. Lutz, R. R. Lutz, A. S. Miner, and H. D. Potter, "Runtime fault detection in programmed molecular systems," *ACM Trans on Software Eng Methodology*, vol. 28, no. 2, pp. 6:1–6:20, 2019.

[14] M. C. Gerten, A. L. Marsh, J. I. Lathrop, M. B. Cohen, A. S. Miner, and T. H. Klinge, "Inference and test generation using program invariants in chemical reaction networks," in *Proc. 44th Int'l Conf on Software Engineering*, 2022.

[15] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. L. Jones, "Certifiably safe software-dependent systems: challenges and directions," in *Proc. Future of Software Engineering*, 2014.

[16] T. Kelly and R. Weaver, "The goal structuring notation–a safety argument notation," in *Proc. DSN Workshop on Assurance Cases*, 2004, p. 6.

[17] J. Knight, *Fundamentals of Dependable Computing for Software Engineers*. CRC Press, 2012.

[18] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd CAV*, ser. LNCS. Springer, 2011, pp. 585–591.

[19] M. R. Lakin, D. Parker, L. Cardelli, M. Kwiatkowska, and A. Phillips, "Design and analysis of DNA strand displacement devices using probabilistic model checking," *Journal of the Royal Society Interface*, vol. 9, no. 72, pp. 1470–1485, 2012.

[20] J. Lathrop, J. Lutz, R. Lutz, H. Potter, and M. Riley, "Population-induced phase transitions and the verification of chemical reaction networks," *Natural Computing*, 2021.
[21] N. G. Leveson, *Safeware, System Safety and Computers*. Addison Wesley, 1995.
[22] P. Mäder, P. L. Jones, Y. Zhang, and J. Cleland-Huang, "Strategic traceability for safety-critical projects," *IEEE Softw.*, vol. 30, no. 3, pp. 58–66, 2013.
[23] SCSC Assurance Case Working Group, *Goal Structuring Notation, Version 3*, 2021.
[24] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri, "A method to generate reusable safety case argument-fragments from compositional safety analysis," *J. Syst. Softw.*, vol. 131, pp. 570–590, 2017.
[25] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with finite stochastic chemical reaction networks," *Natural Computing*, vol. 7, no. 4, pp. 615–633, 2008.
[26] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," in *Proc. 14th Int'l Meeting on DNA Computing*, ser. LNCS, vol. 5347. Springer, 2009, pp. 57–69.
[27] J. Steghöfer, B. Koopmann, J. S. Becker, M. Törnlund, Y. Ibrahim, and M. Mohamad, "Design decisions in the construction of traceability information models for safe automotive systems," in *29th IEEE Int'l Req Eng Conf*, 2021, pp. 185–196.
[28] T. Tun, R. Lutz, B. Nakayama, Y. Yu, D. Mathur, and B. Nuseibeh, "The role of environmental assumptions in failures of DNA nanosystems," in *Workshop Complex Faults and Failures in Large Software Systems*, 2015.
[29] T. T. Tun, M. Jackson, R. Laney, B. Nuseibeh, and Y. Yu, "Are your lights off? using problem frames to diagnose system failures," in *17th IEEE Int'l Req Eng Conf*, 2009, pp. 343–348.
[30] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley, 2009.
[31] M. Vasic, D. Soloveichik, and S. Khurshid, "CRNs exposed: A method for the systematic exploration of chemical reaction networks," in *26th Int'l Conf on DNA Computing and Molecular Programming*, 2020.
[32] T. Viger, R. Salay, G. M. K. Selim, and M. Chechik, "Just enough formality in assurance argument structures," in *39th SAFECOMP*, ser. LNCS, vol. 12234. Springer, 2020, pp. 34–49.
[33] B. Yordanov, C. M. Wintersteiger, Y. Hamadi, and H. Kugler, "SMT-based analysis of biological computation," *NASA Formal Methods*, vol. 7871, pp. 78–92, 2013.
[34] B. Yurke, A. J. Turberfield, A. P. Mills, F. C. Simmel, and J. L. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, no. 6796, pp. 605–608, 2000.