The Assurance Recipe: Facilitating Assurance Patterns

Justin Firestone(✉) and Myra B. Cohen

Department of Computer Science and Engineering, University of Nebraska - Lincoln, Lincoln, NE 68588-0115, USA
{jfiresto,myra}@cse.unl.edu

Abstract. As assurance cases have grown in popularity for safety-critical systems, so too has their complexity and thus the need for methods to systematically build them. Assurance cases can grow too large and too abstract for anyone but the original builders to understand, making reuse difficult. Reuse is important because different systems might have identical or similar components, and a good solution for one system should be applicable to similar systems. Prior research has shown engineers can alleviate some of the complexity issues through modularity and identifying common patterns which are more easily understood for reuse across different systems. However, we believe these patterns are too complicated for users who lack expertise in software engineering or assurance cases. This paper suggests the concept of lower-level patterns which we call recipes. We use the safety-critical field of synthetic biology, as an example discipline to demonstrate how a recipe can be built and applied.

Keywords: Assurance case · Assurance pattern · Synthetic biology iGEM

1 Introduction

Assurance cases have grown in popularity to reason about safety-critical systems. They are commonly used in domains such as aviation, nuclear energy, railways, and offshore drilling [6]. Recent work has proposed using assurance cases in non-traditional domains where engineering safety is also paramount, such as in synthetic biology and medical devices [1,5]. However, complex systems may lead to complex assurance cases that are hard to understand and build for novice users. One way to make assurance cases easier to share and use across domains has been to abstract similarities for reuse via *patterns* [3]. Patterns are meta-models of common argument structures and can be described and catalogued for retrieval and reuse. While useful for an expert in building assurance cases, these patterns may not be usable by the novice who may have too many degrees of freedom to concretize the abstraction.

Instead we propose to provide better guidance through a new abstraction, a template-like model that we call an *assurance recipe*. The user is provided a

B. Gallina et al. (Eds.): SAFECOMP 2018 Workshops, LNCS 11094, pp. 22–30, 2018.
https://doi.org/10.1007/978-3-319-99229-7_3

structure/pattern for the assurance case and is guided to select *ingredients* for a set of options. Although, not as general as a pattern, recipes can be customized for a domain and then parameterized for easy user instantiation.

In this paper we focus on a non-traditional domain (synthetic biology) where users will be non-experts in building assurance cases. We first perform a pre-study to understand if there is sufficient commonality across solutions. We examine projects submitted to the International Genetically Engineered Machine (iGEM) competition between 2015 and 2017 and examine their approaches to safety. We use this data to develop common recipes. We then present a feasibility study to demonstrate how we can apply the recipes in practice. Although the recipes shown only apply to synthetic biology and are small in size, we believe that recipes can be useful for other disciplines and be a starting point for further research, development, and discussion.

2 Background and Related Work

Researchers have recognized a need for modularity and reuse of patterns which can facilitate design and understanding of assurance cases. To address the difficulty of assuring and certifying electronics systems in the aerospace industry, Ruiz et al. [9] suggest combining Case-Based Reasoning (CBR) as a way to represent, retrieve, and reuse previously assured safety cases. For the general safety-critical domains Conmy and Bate [2] propose a method to understand reuse of software components in different contexts, such as when a software module needs to be verified on new hardware. Evidence used to verify a software module in one context is not necessarily sufficient or appropriate in another. They combine Component-Based Software Engineering (CBSE) concepts with argument fragments to guide the user.

Hawkins and Kelly [4] propose a catalog of argument patterns to describe claims that could apply to *any* software assurance case. Similar to software design, argument patterns are abstractions of common strategies (or best practices). This concept was expanded by Szczygielska and Jarzebowicz [10] who proposed an online repository for assurance patterns with a focus on universal application and uniformity across different industry domains.

The *recipes* in this work are influenced heavily by the work of Denney and Pai, who developed a foundation for a theory of assurance patterns, or a *pattern for patterns* [3]. Their work formalizes definitions of argument structures and argument patterns, and provides an algorithm for instantiating their patterns. We use the Claim Formalization Pattern developed by them (shown in Fig. 1) for this work. The field of synthetic biology has been growing rapidly, and uses engineering design techniques to develop models to then program living organisms (*chassis*), such as *Escherichia coli* (*E. coli*), to perform modified or novel functionality [7]. Given that synthetic biology is designing and programming a safety-critical system (these are living organisms that may be released eventually into the environment or used for medicinal purposes), the need to assure their safety is important. Recent work has suggested that assurance cases can be

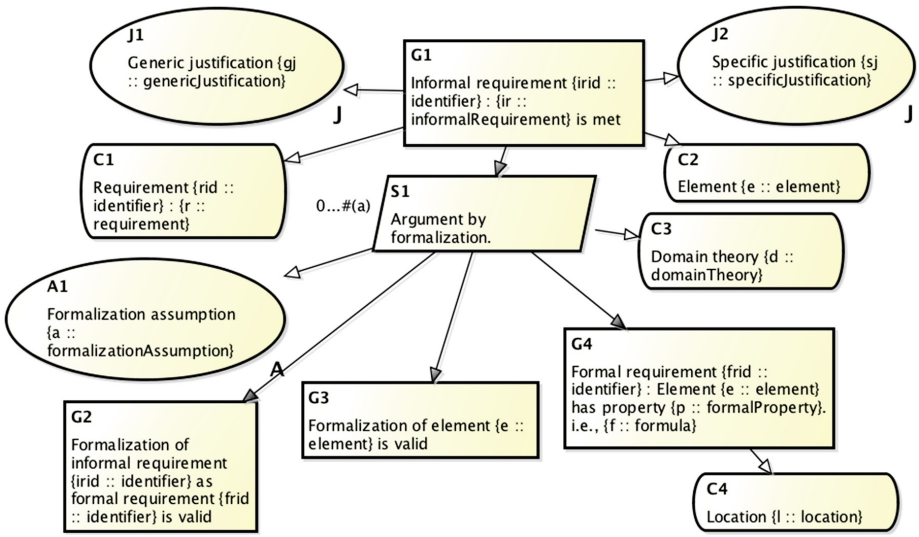


Fig. 1. A high-level pattern based on [3].

used for this purpose [1]. We use the acronym *SEBO* to represent synthetically engineered biological organism.

3 Pre-study

We examined three years of projects from the iGEM competition, held yearly for teams of students from high school through to graduate school. This competition has grown rapidly in popularity, with 314 teams from over 40 countries in 2017 [11]. The iGEM competition provides a plentiful source of real-world projects because teams are encouraged to share information in an open-source database. Teams must also explicitly discuss safety of their projects.

Teams can attempt to earn a bronze, silver, or gold medal for satisfying increasingly stringent criteria. Gold-medal teams must accomplish Integrated Human Practices, which asks teams to "consider whether their projects are safe, responsible, and good for the world" [13]. We thus limited our investigation to gold-medal teams from 2015–2017, assuming they addressed safety.

The most common chassis for iGEM projects is the K-12 strain of *E. coli*, a bacterium which does not inherently produce toxins and can be handled in a Safety Level One lab. By far, most experiments are performed in a Safety Level One lab using organisms which are *generally regarded as safe* (GRAS) [17]. However, even the relatively safe K-12 *E. coli* could pose safety-critical risks outside the lab, since SEBOs have added functionality and behavior.

3.1 Categorization Methodology

We manually investigated the safety pages of every gold-medal iGEM team from 2015–2017. There were 334 projects which we grouped into the following categories of safety features.

- **Containment (Con)**: the organisms and their products are safely contained in the lab and the SEBOs are not intended for release into the environment.
- **Kill-switch (KS)**: upon unwanted evolution or mutation, a process is triggered to actively kill the SEBOs, often through lysis of cell membranes.
- **Auxotrophy (Aux)**: the organisms cannot survive without the presence of a specific chemical or food source.
- **Degradation (Deg)**: the organisms or their products degrade naturally over time when exposed to certain environmental conditions.
- **Sterility (Ster)**: the organisms' offspring are engineered to be sterile.
- **GRAS**: the organisms and their products are in Risk Group 1 as defined by the FDA or the NIH.

Some of the categorization required making subjective determinations about which safety features were used. Also, some teams used terminology inconsistently. These subjective decisions represent a threat to the validity of our categories and statistics, but we nevertheless believe them to be a fair representation.

Table 1. Most-common safety features from recent iGEM competitions.

Year	# Gold medals	KS	Con	GRAS	Aux	Deg	Ster
2015	114	7.02%	14.04%	50.00%	3.51%	1.75%	0.88%
2016	111	12.61%	20.72%	43.24%	2.70%	5.41%	0.00%
2017	109	13.76%	16.51%	44.04%	6.42%	0.00%	0.00%

Most teams base the safety of their projects on the mere fact their organisms are GRAS (43–50%), but some teams add safety features into their projects. Table 1 shows usage of safety features. The percentages do not add up to 100% for a few reasons. First, many teams did not complete their webpage stubs for the Safety category, or it was otherwise unclear what their approach for safety was. It is likely those teams were relying on GRAS *Only* as their safety feature, meaning the percentages from Table 1 for GRAS *Only* are probably under-reported. Second, some of the teams considered more than one safety feature. We list all that they use. Finally, some teams entered projects which were *in silico*, such as software or hardware improvements. Table 1 demonstrates that we can group safety techniques into a small number of categories, with a Kill-switch (7–14%) and Containment (14–17%) being the most popular mechanisms. We show in the next section how we developed recipes for these.

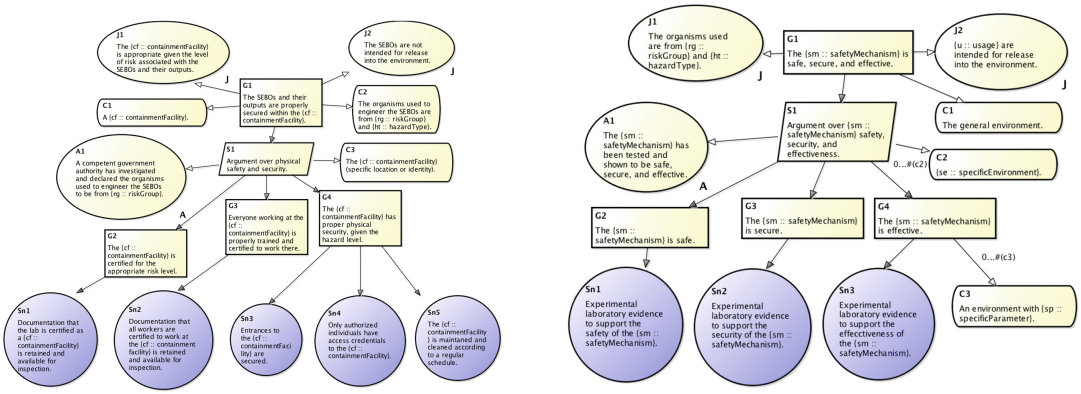
4 Assurance Recipes

To build an assurance recipe we begin with a pattern. We use the Claim Formalization Pattern from [3] for our recipes. We use goal structuring notation [6] and leave the evaluation compared to alternative modeling languages as future work.

We build recipes for the two most-common safety features in Table 1. The first, the *Containment Recipe*, assumes SEBOs are not intended for release into the environment. The second, the *Safety Mechanism Recipe*, assumes SEBOs will be applied outside of the laboratory.

4.1 Containment Recipe

For this recipe we focus on the safety levels and risks. We show this recipe in Fig. 2(a). It is intended to address the four lab safety levels and the risks associated with organisms requiring those safety levels. The assumption is that a competent government agency has declared the organisms to be from a specific risk group. The evidence needed will mostly be documentation of adequate training and physical measures to certify the lab. Table 2 suggests ingredients for the recipe from which the user can select the appropriate choices.



(a) Containment (b) Safety

Fig. 2. Recipes for (a) Containment and (b) Safety.

4.2 Safety Mechanism Recipe

The recipe in Fig. 2(b) addresses the most-common SEBO safety mechanisms. The assumption is that the SEBOs will be released into the environment and they pose some risk of harm. The evidence will heavily rely on wet-lab experimental data. The safety and security of the SEBOs are included as sub-goals, but there may not be sufficient experimental evidence to support them. Table 3 suggests ingredients for the recipe.

5 Feasibility

We used these recipes on four real projects. The first is the 2017 University of Nebraska - Lincoln team. They thoroughly documented the safety training each member completed before working in the lab, therefore we used the Containment

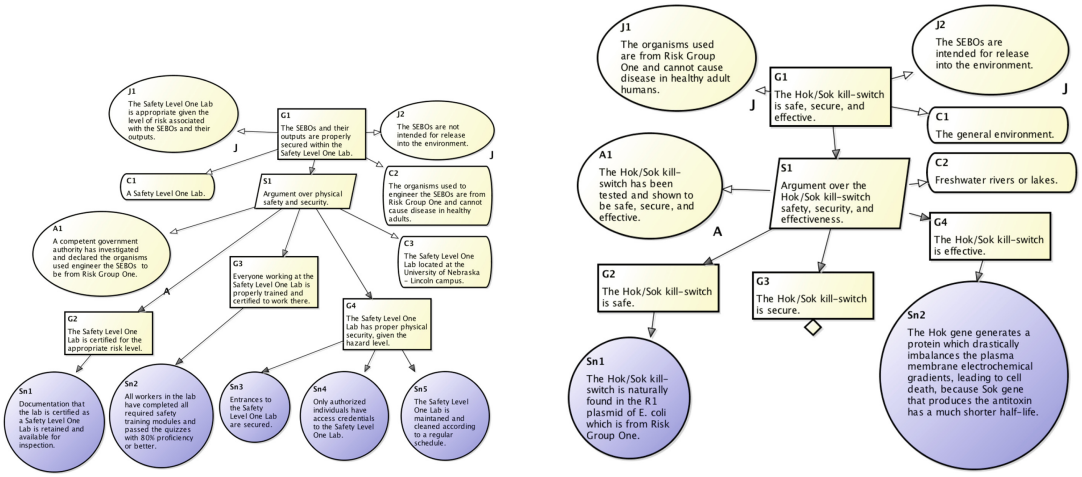
Table 2. Suggested ingredients for the Containment Recipe.

Variable	Ingredients
cf :: containmentFacility	1. Safety Level One Lab 2. Safety Level Two Lab 3. Safety Level Three Lab 4. Safety Level Four Lab
rg :: riskGroup	1. Risk Group One 2. Risk Group Two 3. Risk Group Three 4. Risk Group Four
ht :: hazardType	1. Cannot cause disease in healthy adults; 2. Can cause treatable or preventable disease in humans; 3. Can cause serious disease in humans which might not have a treatment or vaccine; 4. Can cause serious disease in humans which has no known treatment or vaccine

Table 3. Suggested ingredients for the Safety Mechanism Recipe.

Variable	Ingredients
sm :: safetyMechanism	1. Kill-switch 2. Auxotrophy 3. Degradation 4. Sterility
u :: usage	1. The SEBOs 2. Only the SEBOs' outputs
se :: specificEnvironment	1. Soil 2. Water table 3. [Specific species habitat] 4. Atmosphere 5. Rivers 6. Freshwater rivers or lakes 7. Saltwater lakes or oceans 8. Human body 9. [Non-human] body
sp :: specificParameter	1. [Temperature Range] 2. [pH Range] 3. [Aerobic/Anaerobic] environment 5. [Natural/Specific frequencies/Absence] of light 7. [Presence/Absence] of nutrients 8. [Altitude range]

Recipe. They listed themselves as a Safety Level 1 lab, in Risk Group 1, and were thus the first hazardType. For evidence we need to demonstrate that the students followed protocols for containment. We were able to use their documentation to fill in evidence such as Sn2 (of the recipe). They wrote: "We took a total of 6 safety modules including a Biosafety Level 1 course before we were allowed to work in the lab. All modules that required a quiz to be taken had to be completed with 80% proficiency" [15]. The assurance case is shown in Fig. 3(a).



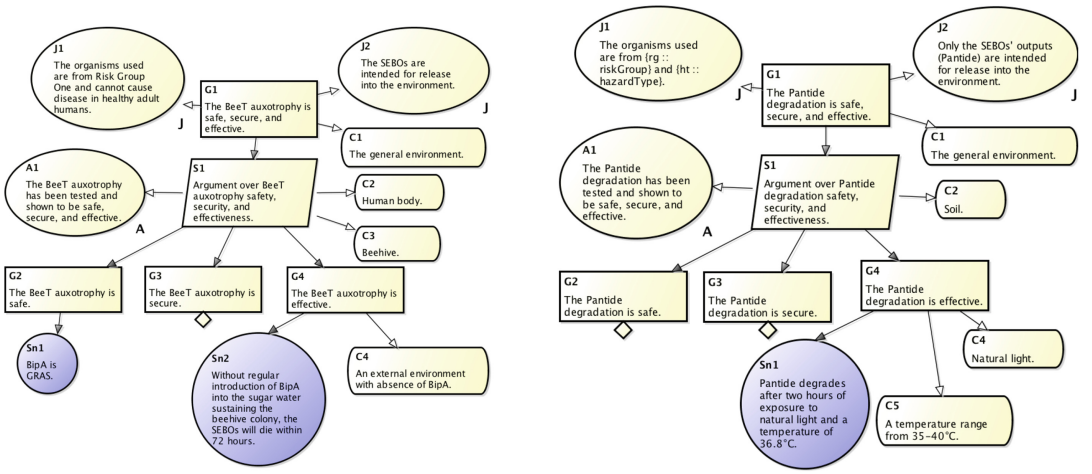
(a) containment (b) kill-switch

Fig. 3. Assurance case fragments for (a) Containment and (b) Kill-switch.

Since kill-switches are one of the most common approaches to non-containment safety we built a recipe for the Hok/Sok kill-switch from a University of Maryland team, one of the most commonly implemented kill-switches [14]. The key feature is the evidence based on wet-lab experiments showing that the kill-switch will trigger within 30 s of mutation. This is shown in Fig. 3(b).

We also implemented an auxotrophy assurance case based on the 2016 Wageningen team [16] (see Fig. 4(a)). They implemented the auxotrophic system developed in [8]. They designed SEBOs to produce a chemical which helps bees defend against mites. The SEBOs need a synthetic amino acid, BipA, to survive, which beekeepers add to the sugar water feeding the bee colony. If the SEBOs escape the beehive, they will die within 72 h without BipA.

Last, we built a degradation assurance case based on the 2016 Formosa team which developed a pesticide called Pantide [12]. Because it was a novel toxin, the team performed experiments to determine it sufficiently degrades within two hours of exposure to natural light at 36.8C. This is shown in Fig. 4(b).



(a) auxotrophy (b) degradation

Fig. 4. Assurance case fragments for (a) Auxotrophy and (b) Degradation.

6 Conclusions and Future Work

In this paper we have presented the idea of an assurance recipe and demonstrated how it can be applied in the non-traditional domain of synthetic biology. Although our recipes are based on the most common safety features of SEBOs from iGEM projects, they are intended for general application to any SEBOs, and should be helpful for other safety-critical disciplines. Assurance recipes and ingredients can facilitate use and reuse by domain experts who lack expertise with building assurance cases. In future work we will apply these more generally and are working on an interactive software system to help iGEM users build assurance cases using recipes.

Acknowledgments. This work was supported in part by the National Institute of Justice grant 2016-R2-CX-0023 and the National Science Foundation Grant CCF-1745775.

References

1. Cohen, M.B., Firestone, J., Pierobon, M.: The assurance timeline: building assurance cases for synthetic biology. In: Skavhaug, A., Guiochet, J., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2016. LNCS, vol. 9923, pp. 75–86. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45480-1_7
2. Conmy, P., Bate, I.: Assuring safety for component based software engineering. In: 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE), pp. 121–128. IEEE (2014)
3. Denney, E.W., Pai, G.J.: Safety case patterns: theory and applications (2015)
4. Hawkins, R., Kelly, T.: A Software Safety Argument Pattern Catalogue. The University of York, York (2013)

5. Jee, E., Lee, I., Sokolsky, O.: Assurance cases in model-driven development of the pacemaker software. In: Margaria, T., Steffen, B. (eds.) ISoLA 2010. LNCS, vol. 6416, pp. 343–356. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16561-0_33
6. Kelly, T., Weaver, R.: The goal structuring notation-a safety argument notation. In: Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, p. 6. Citeseer (2004)
7. Levskaya, A., Chevalier, A.A., Tabor, J.J., Simpson, Z.B., Lavery, L.A., Levy, M., Davidson, E.A., Scouras, A., Ellington, A.D., Marcotte, E.M., et al.: Synthetic biology: engineering *Escherichia coli* to see light. Nature **438**(7067), 441 (2005)
8. Mandell, D.J., Lajoie, M.J., Mee, M.T., Takeuchi, R., Kuznetsov, G., Norville, J.E., Gregg, C.J., Stoddard, B.L., Church, G.M.: Biocontainment of genetically modified organisms by synthetic protein design. Nature **518**(7537), 55–60 (2015)
9. Ruiz, A., Habli, I., Espinoza, H.: Towards a case-based reasoning approach for safety assurance reuse. In: Ortmeier, F., Daniel, P. (eds.) SAFECOMP 2012. LNCS, vol. 7613, pp. 22–35. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33675-1_3
10. Szczygielska, M., Jarzębowicz, A.: Assurance case patterns on-line catalogue. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds.) DepCoS-RELCOMEX 2017. AISC, vol. 582, pp. 407–417. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-59415-6_39
11. igem.org
12. http://2016.igem.org/Team:NCTU_Formosa/Safety
13. http://2017.igem.org/Human_Practices
14. 2015.igem.org/Team:UMaryland/HokSok
15. 2017.igem.org/Team:UNebraska-Lincoln/Safety
16. http://2016.igem.org/Team:Wageningen_UR/Safety
17. http://osp.od.nih.gov/wp-content/uploads/NIH_Guidelines.html