Adopting Patterns by Analogy Through Drama

Aleksandra Vranić
aleksandra.vranic@paneurouni.com
Pan-European University
Faculty of Media
Slovakia

Valentino Vranić
valentino.vranic@paneurouni.com
Pan-European University
Faculty of Informatics
Slovakia

Zishan Rahman
zishanfrahman@outlook.com
King's College London
Department of Informatics
United Kingdom

ABSTRACT

When staging plays with drama patterns, participants are never provided with their general description. All instructions they get are always in terms of the scene to be played, i.e., particular pattern instance—or instances, if the scene is composed out of several drama patterns. Yet, they tend to recognize the pattern they played before when they play it again in a completely different context and even to apply it elsewhere on their own—by analogy. Adopting patterns by analogy extends across domains. Drama patterns, which are—as instances—directly comprehensible even to small children, can be seen as metaphorical analogies of design patterns, which are not so easy to grasp. Participating in a drama pattern helps adopt the core idea of the design pattern it corresponds to—again, by analogy. In the focus group reported here, we built a series of different scenes using the same drama patterns: *Reflection Play*, *Reversed Advantage*, *Thoughts Reflecting Environment*, and *Mediator*. We observed and discussed how the participants picked up these patterns directly from their application and were able to apply them in different contexts by analogy. We discussed how these drama patterns represent metaphorical analogies of certain design patterns, namely *Reflection*, mock object patterns, *Observer*, and *Mediator*. We also report the analogy of drama with nonlinear systems where drama patterns can be seen as regulators.

CCS CONCEPTS

• **Software and its engineering → Patterns**.

KEYWORDS

drama patterns, design patterns, organizational patterns, reflection, mocking, nonlinear systems

ACM Reference Format:
Aleksandra Vranić, Valentino Vranić, and Zishan Rahman. 2024. Adopting Patterns by Analogy Through Drama. In *Proceedings of 31st Conference on Pattern Languages of Programs, People, and Practices (PLoP 2024).* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

Alexander's idea of a pattern, originally observed in building architecture [2, 4], shows to pertain virtually all areas of human activity and life in general. Good, humane design can always be followed and established through patterns. Good drama is also made of patterns—drama patterns [20]. Drama patterns are a special kind of organizational patterns [23]. A drama pattern represents a particular intriguing dramatic situation that tends to recur in different contexts and gives a clue how to resolve it [20].

The organizers of this focus group—Aleksandra and Valentino—explored drama patterns extensively over the past six years [9, 14–25] finding them capable of providing an instant experience of the idea of a pattern in general [14, 17, 18] and of software patterns in particular [15, 19, 23–25].

When staging plays with drama patterns, participants are never provided with their general description. All instructions they get are always in terms of the scene to be played, i.e., particular pattern instance—or instances, if the scene is composed out of several drama patterns. Yet, they tend to recognize the pattern they played before when they play it again in a completely different context and even to apply it elsewhere on their own—by analogy.

Adopting patterns by analogy extends across domains. Drama patterns, which are—as instances—directly comprehensible even to small children, can be seen as metaphorical analogies of design patterns [15, 19, 24], which are not so easy to grasp. Participating in a drama pattern helps adopt the core idea of the design pattern it corresponds to—again, by analogy.

Apart from the organizers, the focus group attracted five participants, one of whom (Zishan) took part in writing this report, too. In our focus group, we built a series of different scenes using the same drama patterns. We played most of these scenes with the participants. We also discussed how the drama patterns we played represent metaphorical analogies of certain design patterns. Sections 2–5 describe this in detail. Section 6 describes the analogy with nonlinear systems we identified after the focus group. Section 7 concludes the report.

2 REFLECTION PLAY

We started our focus group with a relaxing scene called Mirrors. This is actually an instance of the *Reflection Play* drama pattern [20]: the reflected object doesn't believe that what has shown is real and that it is its reflection, *but* the reflection behaves the same. This conflict of contradicting forces is resolved by the reflected object accepting the existence of what appears to be its reflection.

One of the participants played the reflected object, while others played its reflections. We played it in two variants. In one, the reflections repeated the moves of the reflected object upon being approached. In the other one, they all repeated the moves of the reflected object, who was centrally positioned, at once. Roughly, this was all the participants were said to do. No general description of the pattern was provided to them.

PLoP 2024, October 13–16, 2024, Skamania Lodge, Columbia River Gorge, Washington, USA.

Hillside ISBN 978-1-941652-20-6

An immediate metaphorical analogy of the *Reflection Play* drama pattern to reflection in programming is apparent. This mechanism that enables to inspect and even manipulate the elements of a program through other elements that *reflect* them has also been recognized as a design pattern called simply *Reflection* [6]. In the Mirror scene, obviously, the focus is on inspection rather than on manipulation. However, if the Marionette drama pattern [20] is subsequently applied so that the reflection takes the role of the marionettist and starts to control the moves of the reflected object, we could also cover the reflection's manipulation aspect.

3 REVERSED ADVANTAGE

The next scene we played was *The Real Red Riding Hood.* This scene—or rather a short play—is based on the *Reversed Advantage* drama pattern [20]: advantage claimers want to benefit from the acknowledgment of their advantage by others, *but* it occurs that their advantage endangers them when advantage usurpers appear. This conflict of contradicting forces is resolved by advantage claimers (publicly) reversing their claims in order to protect themselves, yet not yielding their advantage internally (keeping it to themselves).

Three participants played Red Riding Hoods who meet at a glen and argue over who is the real one. Then the wolves come—played by the other four participants—asking who among them is the real Red Riding Hood. The advantage of being the real Red Riding Hood suddenly reverses and the Red Riding Hoods make excuses and leave the glen. The wolves remain confused. Again, the participants were directed with respect to the scene without explaining the pattern in general.

We also played this scene two times and then built another scene called *The Wallet* by analogy (see Figure 1). One participant played a person who initially found somebody's wallet on the sidewalk and wanted to keep it. Three other participants were entering the scene one by one claiming the wallet is theirs. The advantage of having the wallet suddenly reveres as the police officers—played by two participants—enters the scene asking who stole the wallet. Just the alleged Red Riding Hoods, the alleged wallet owners deny they have anything to do with the wallet and leave the scene. Just like the wolves, the police officers remain confused.

By this, the participants experienced how effective is to learn patterns from examples and how easy it is to apply them by analogy. While generalized pattern descriptions is certainly useful, it is not necessary for learning and applying patterns.

We discussed applying *Reversed Advantage* in two other scenes (we didn't actually play them). One of them we call At the Dentist's [20]. Patients wait at the dentist's, arguing who should go first, but when the dentist appears asking who's next, they make excuses and leave. The other one takes place in a restaurant, where two or more people argue over the last dish, but when the cook announces it contains something they don't like or doesn't contain something they want, they make excuses and give it up. Aleksandra developed this scene when adapting *Salt More Than Gold,*[1] a famous Slovak folk tale, for a play with the children she was teaching in an elementary arts school she worked at.

As Zishan pointed out during the focus group, *Reversed Advantage* could be seen as a metaphorical analogy of mocking in software testing, which has been elaborated as a pattern language [5]. Mocking is a technique in software testing where certain dependencies in an object or class are isolated, such that only the behaviors that we expect from said object are tested. Mockito is one such library for implementing mocking in JUnit tests [7], The equivalence to the Reversed Advantage drama pattern, among mocking, is that in a unit test that uses mocks, the mock objects replicate the behaviors of a class with particularly high dependencies without actually being an instance of the class, allowing developers to quickly test the expected behaviors of a mocked class (against the actual behaviors) without relying on the actual class's possibly complicated dependencies. Similarly, in the *Reversed Advantage* pattern, the advantage claimers act as if they bear advantages they do not actually possess. This can be seen in The Wallet scene, for instance, where multiple people are claiming to be the owner of a wallet left on the sidewalk and all of them act like they do when in fact none of them own the wallet.

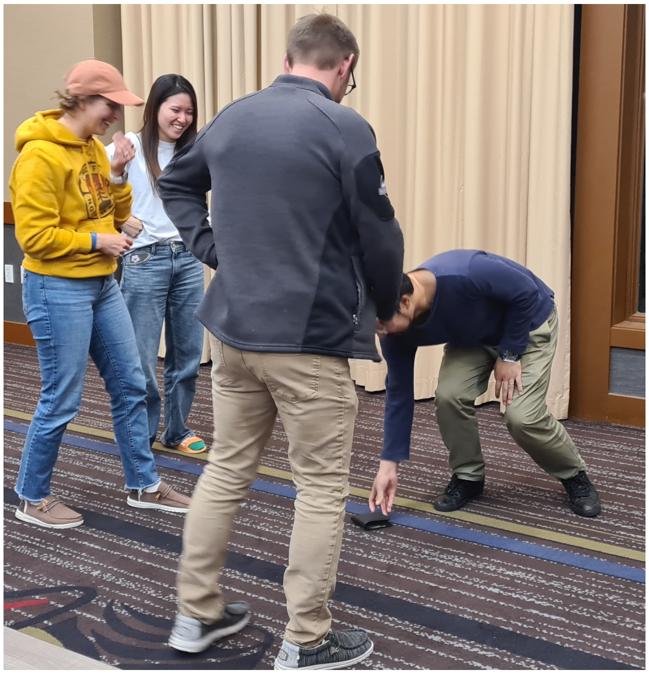

Figure 1: The Wallet scene.

4 THOUGHTS REFLECTING ENVIRONMENT

Next, we played another scene from the extended version of *The Real Red Riding Hood,* which is called *Birds* (see Figure 2). It takes place before the glen scene. Red Riding Hood is walking through the woods. She is scared and sounds of birds start to change to human voices in her mind saying "I'm afraid. I can't find my way. I'm cold. . . ", which is exactly what is presented to the audience. One participant played Red Riding Hood, while all other played the birds.

This is an instance of the *Thoughts Reflecting Environment* drama pattern [20]: the environment—the birds in this case—reflects the protagonist's thoughts. There is a need to express the protagonist's

[1] The original Slovak title is *Soľ nad zlato.*

Figure 2: The Birds scene.

thoughts, *but* without having the protagonist directly express them. This conflict of contradicting forces is resolved by the environment reflecting the protagonist's thoughts by modulating the sounds or visual expressions it already makes.

This is a metaphorical analogy of *Observer* [8]: the protagonist is the subject, while the birds are the observers [24].

By analogy, We also played the *Traffic Sounds* scene, developed by Aleksandra with her university students: a person is lost in the city and traffic sounds change to voices. When acting out the scene, the scene dragged on until eventually Zishan was given the cue to end the scene on a lighter note, preventing the tension in the scene from otherwise being unresolved. This was briefly discussed after the scene was acted out and concluded, although we have not determined an equivalence to this among software design patterns yet.

5 MEDIATOR

The last scene we played was *Family Argument* (see Figure 3). A father has an argument with his two sons. However, he's not talking to them directly, but through their mother, who acts as a mediator.

This is the *Mediator* drama pattern [15]: the parties need to talk to each other, but can't understand each other or won't listen to each other. This conflict of contradicting forces is resolved by a mediator, who understands each of the parties, translating or transfers what's been said between the parties.

It happens that there is a design pattern with the same name: *Mediator* [8]. It enables to resolve the situation when colleague objects need to interact, but this requires including special methods for each new kind of a colleague in their interface. For this, a mediator object, which translates the messages between the colleague objects, is introduced.

6 NONLINEAR SYSTEMS AS DRAMA

Without a doubt, a drama is a complex system—just as nonlinear systems are. Patterns could be seen as regulators in nonlinear systems. We are adding them to change the system output. There could be different variants of such a system, some of which would represent its simplifications. While *Reversed Advantage* is essential for *The Real Red Riding Hood*, there are other drama patterns that contribute to this play, one of which is *Amazing Similarity*. However, if our Red Riding Hoods are not similar to each other, we simplify the play by excluding this pattern. If applied, *Amazing Similarity* can be made stronger by applying *Reflection Play*, but this is not inevitable. This is similar to a decomposed control of trajectories in nonlinear systems [11, 13] with possible simplifications [10], which may even involve blockchain [12]. It is interesting to note that Alexander also worked on mathematical foundations of patterns in early days [1], where he saw them in graphs and set theory, getting back to this in a more comprehensive way in is work on the nature of order [3].

Consequently, drama patterns could be used to build metaphorical analogies of nonlinear systems. The participants would have an opportunity to feel their essence from inside. By applying or retracting different drama patterns, they could experience the control of nonlinear systems.

7 CONCLUSIONS

In the focus group reported here, we built a series of different scenes using the same drama patterns: *Reflection Play*, *Reversed Advantage*, *Thoughts Reflecting Environment*, and *Mediator*. We observed and discussed how the participants picked up these patterns directly from their application and were able to apply them in different contexts by analogy. We also discussed how these drama patterns represent metaphorical analogies of certain design patterns, namely *Reflection*, mock object patterns, *Observer*, and *Mediator*. We also report the analogy of drama with nonlinear systems where drama patterns can be seen as regulators.

ACKNOWLEDGMENTS

We would like to thank Rio Nitta, Zishan Rahman, Hiroaki Tanaka, Mindi Weik, and Matthew Weik for participating in the focus group reported here. We would also like to thank Volodymyr V. Khilenko for a fruitful discussion on drama patterns and nonlinear systems, as well as to Michael Weiss and Takashi Iba for their suggestions that helped us improve this report.

Figure 3: The Family Argument scene.

REFERENCES

[1] Christopher Alexander. 1964. *Notes on the Synthesis of Form.* Harvard University Press.

[2] Christopher Alexander. 1979. *The Timeless Way of Building.* Oxford University Press.

[3] Christopher Alexander. 2003. New Concepts in Complexity Theory Arising from Studies in the Field of Architecture: An Overview Of the Four Books of the Nature of Order with Emphasis on the Scientific Problems which are Raised. Katarxis № 3, katarxis3.com/SCIENTIFIC INTRODUCTION.pdf.

[4] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson, and Ingrid Fiksdahl-King. 1977. *A Pattern Language.* Gustavo Gili.

[5] Matthew A. Brown and Eli Tapolcsanyi. 2003. Mock Object Patterns. In *Proceedings of 10th Conference on Pattern Languages of Programs, PLoP 2003.* Allerton Park, Monticello, IL, USA. Available at hillside.net/plop/plop2003/papers.html.

[6] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1995. *Pattern-Oriented Software Architecture, Volume 1, A System of Patterns.* Addison-Wesley.

[7] Szczepan Faber et al. 2025. Mockito. site.mockito.org.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

[9] Patrik Honíšek and Valentino Vranić. 2020. Mining Drama Patterns in Dramatic Situations. In *Proceedings of 27th Conference on Pattern Languages of Programs Online, PLoP 2020.* ACM.

[10] Volodymyr V. Khilenko. 1999. State Estimation and Filtering in Stochastic Systems Using Adequate Simplification. *Cybernetics and Systems Analysis* 35, 5 (1999), 764–768.

[11] Volodymyr V. Khilenko. 2001. Application of algorithms of decomposition for calculating the linear stochastic models. *Kibernetika i Sistemnyj Analiz* 37, 4 (2001), 159–163.

[12] Volodymyr V. Khilenko. 2020. Application of Blockchain Technologies for Improving the Quality of ACS of Complex Dynamic Systems. *Kibernetika i Sistemnyj Analiz* 56, 2 (2020), 181–186.

[13] Volodymyr V. Khilenko. 2022. An Algorithm for Decomposition Control and Prediction of Trajectories of Nonlinear Stochastic Systems Under Different-Speed Processes in Their Dynamics. *Cybernetics and Systems Analysis* 58, 2 (2022), 220–224.

[14] Aleksandra Vranić. 2019. Experience Drama Patterns. Research seminar, AdvanSD Research Group, Slovak University of Technology, val.poetisania.com/advansd/#drama.

[15] Aleksandra Vranić and Valentino Vranić. 2024. Understanding Design Patterns Through Drama Metaphors. In *Proceedings of 2024 IEEE 17th International Scientific Conference on Informatics, Informatics 2024.* IEEE, Poprad Slovakia.

[16] Aleksandra Vranić, Valentino Vranić, and Branislava Vranić. 2019. Drama Patterns: Programs, People, and Practices. Focus group, 10th Asian Conference on Pattern Languages of Programs, People, and Practices, AsianPLoP 2024, val.poetisania.com/pub/#aplop24.

[17] Aleksandra Vranić, Valentino Vranić, and Branislava Vranić. 2019. Drama Patterns: Seeing the Patterns from Within. In *Proceedings of 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019.* ACM, Irsee, Germany.

[18] Aleksandra Vranić, Valentino Vranić, and Branislava Vranić. 2019. Understanding Humane in Patterns Through Drama. PLoP 2022 PLoPourri Event. 29th Conference on Pattern Languages of Programs Online, PLoP 2022, val.poetisania.com/pub/#drama-plop22.

[19] Valentino Vranić. 2024. Drama and Software Patterns. In *Proceedings of 2024 IEEE 17th International Scientific Conference on Informatics, Informatics 2024.* IEEE, Poprad Slovakia.

[20] Valentino Vranić and Aleksandra Vranić. 2019. Drama Patterns: Extracting and Reusing the Essence of Drama. In *Proceedings of 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019.* ACM, Irsee, Germany.

[21] Valentino Vranić and Aleksandra Vranić. 2021. Patterns: Code–People–Drama. Series of Erasmus+ mobility educational activities. Sinergija University, Bijeljina, Republic of Srpska, Bosnia and Herzegovina.

[22] Valentino Vranić and Aleksandra Vranić. 2024. Creativity Through Patterns: Interconnecting Writing, Programming, and People. Series of Erasmus+ mobility educational activities. Sinergija University, Bijeljina, Republic of Srpska, Bosnia and Herzegovina.

[23] Valentino Vranić, Aleksandra Vranić, and Waheedullah Sulaiman Khail. 2020. Growing Organizations with Patterns: Lessons from Drama. In *Proceedings of 25th European Conference on Pattern Languages of Programs, EuroPLoP 2020 Online.* ACM.

[24] Valentino Vranić, Aleksandra Vranić, and Branislava Vranić. 2023. Dramatizing Software Patterns: Focus Group Report. In *Proceedings of 28th European Conference on Pattern Languages of Programs, EuroPLoP 2023.* ACM, Irsee, Germany.

[25] Valentino Vranić, Aleksandra Vranić, and Branislava Vranić. 2023. Learning about Organizational Patterns from Drama Patterns: Focus Group Report. In *Proceedings of 30th Conference on Pattern Languages of Programs, PLoP 2023.* ACM, Allerton Park, Monticello, IL, USA. Accepted. Available at hillside.net/plop/2023/papers/proceedings.