Spielerisch Softwaredesign verstehen: Die erweiterte Lernsoftware „Pattern Park“

Peer Stechert

Der Artikel stellt die neue Version der Lernsoftware „Pattern Park“ vor, die an der Universität Siegen entwickelt wurde, um die didaktischen Potenziale von Entwurfsmustern für den Informatikunterricht zu erschließen. Als lebensnahes Szenario dient ein Freizeitpark, der die Muster in einem vertrauten Kontext veranschaulicht. Die erste Version der Software umfasste acht Muster. 2021 wurde die Software von Studierenden der Hochschule Stralsund überarbeitet und um acht zusätzliche Muster ergänzt, darunter das Model-View-Controller-Muster.

1 Einleitung

Der Artikel ist wie folgt gegliedert: In Abschnitt 1 werden Architektur- und Entwurfsmuster kurz fachlich vorgestellt und in Abschnitt 2 ihr Potential aus Sicht der Informatikdidaktik analysiert. Die ursprüngliche Version der Lernsoftware Pattern Park wird in Abschnitt 3 mit den in ihr enthaltenen Mustern vorgestellt. Abschnitt 4 zeigt die neuen Muster in der erweiterten Version. Das neu hinzugefügte Architekturmuster Model-View-Controller (MVC) wird aufgrund seiner Relevanz für den Schulunterricht ausführlicher fachdidaktisch analysiert. Am Beispiel des Moduls zum MVC wird in Abschnitt 5 der generelle Aufbau der Lernmodule zu den Entwurfsmustern erläutert. Abschnitt 6 diskutiert, wie Pattern Park in den Informatikunterricht integriert werden kann. Der Artikel schließt mit einem Ausblick in Abschnitt 7.

In der Softwareentwicklung sind Architektur- und Entwurfsmuster wesentliche Werkzeuge, die bewährte Lösungen für wiederkehrende Designprobleme bieten. Sie ermöglichen es Entwickler/innen, Software nachhaltig und effizient zu gestalten, indem sie klar definierte Strukturen und Vorgehensweisen vorgeben. Diese Muster abstrahieren gängige Probleme und präsentieren Lösungen, die wiederholt und in unterschiedlichen Kontexten angewendet werden können.

Architekturmuster wie Model-View-Controller (MVC) sind auf die Strukturierung ganzer Anwendungen ausgerichtet. Entwurfsmuster hingegen, wie der Beobachter (Observer) oder der Dekorierer (Decorator), adressieren spezifischere Probleme auf Klassen- und Objektebene. Der „Observer“ dient beispielsweise dazu, eine lose Kopplung zwischen einem Subjekt und seinen Beobachtern herzustellen, wodurch Veränderungen in einem Objekt effizient an abhängige Objekte weitergegeben werden. Das Kompositum (Composite) hingegen vereinfacht den Umgang mit hierarchischen Strukturen, indem es zusammengesetzte und primitive Objekte einheitlich behandelt.

Die Ursprünge dieser Konzepte liegen in der Architekturtheorie, wo Christopher Alexander die Idee verallgemeinerter Problemlösungen erstmals vorstellte. In den 1990er-Jahren wurden diese Prinzipien durch die Gang of Four (Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides) auf die Softwareentwicklung übertragen. Ihr Buch *„Design Patterns: Elements of Reusable Object-Oriented Software“* stellt 23 zentrale Entwurfsmuster vor, die in der Informatik bis heute vielfach genutzt werden.

2 Warum Entwurfsmuster im Informatikunterricht behandeln?

Entwurfsmuster bieten nicht nur technische, sondern auch didaktische Vorteile. Sie machen abstrakte Prinzipien der Softwareentwicklung sowie fundamentale Ideen des Faches greifbar und vermitteln ein Verständnis für die Strukturierung und Gestaltung von Programmen (Stechert, 2009). Ihre Bedeutung wird durch Bildungspläne hervorgehoben: Im bayerischen Lehrplan für Informatik etwa werden Entwurfsmuster explizit im Unterricht der zwölften Jahrgangsstufe gefordert. Auf grundlegendem Anforderungsniveau wird im Jahrgangsstufenprofil das Muster Kompositum genannt (ISB, 2025a). Im Fachlehrplan heißt es: *„[Schüler/innen] erläutern die Grundidee des Architekturmusters Model-View-Controller“* (ISB, 2025b). Auf erhöhtem Anforderungsniveau heißt es allgemeiner: *„[Schüler/innen] verwenden Entwurfs- bzw. Architekturmuster […].“* (ISB, 2025c). In den Informatik-Fachanforderungen für die Oberstufe in Schleswig-Holstein steht: *„Des Weiteren implementieren die Schülerinnen und Schüler diese Modelle unter Verwendung geeigneter Strukturmuster und Architekturen“* (MBWK, 2021). Verbindlich sind das Observer-Muster und das MVC-Architekturmuster für das erhöhte Anforderungsniveau. Ebenso sind Muster immer wieder Prüfungsthema in den bundesweit zentralen Abschlussprüfungen der IT-Fachinformatiker/innen mit Fachrichtung Anwendungsentwicklung.

Für Schüler/innen bieten Entwurfsmuster eine strukturierte Herangehensweise an komplexe Problemstellungen. Sie lernen nicht nur, wie sie Programme effizient entwerfen, sondern auch, wie sie bestehende Software analysieren und verbessern können. Diese Kompetenzen sind nicht nur im Studium, sondern auch in der Berufspraxis von unschätzbarem Wert.
Dadurch, dass Entwurfsmuster typischerweise als Klassendiagramme visualisiert werden, erweitern sie auch die Fähigkeit

der Schüler/innen im Umgang mit diesen Visualisierungen. Sie bieten die Chance, anhand einer Palette an typischen Beziehungen zwischen Klassen diese besser zu verstehen.

3 Die Ursprünge von Pattern Park

Pattern Park wurde ab 2007 an der Universität Siegen im Rahmen des Promotionsprojekts des Autors entwickelt, um Entwurfsmuster für die Sekundarstufe II didaktisch aufzubereiten (Stechert, 2009). Die zentrale Idee war, die oft abstrakten Konzepte der Entwurfsmuster durch ein lebensnahes Szenario anschaulich und nachvollziehbar zu machen. Um einen lebensweltnahen Bezug zur Thematik der Entwurfsmuster herzustellen, wurde ein Szenario gesucht, welches folgende Anforderungen erfüllen soll:

- Lebensweltbezug gemäß der Zielgruppe,
- Gendersensitivität,
- enthält möglichst viele Entwurfsmuster,
- veranschaulicht Fundamentale Ideen der Informatik (Schwill, 1993).

Hierfür wurde der Freizeitpark als Szenario gewählt, da er sowohl für Jugendliche als auch Erwachsene eine vertraute Umgebung darstellt (Abb. 1).

Die Software kombiniert deskriptive, interaktive und explorative Elemente, um Schüler/innen aktiv in den Lernprozess einzubinden. Jeder Bereich des Freizeitparks repräsentiert ein Modul, in dem ein spezifisches Entwurfsmuster behandelt wird. Die Schwerpunkte der Software liegen auf fundamentalen Konzepten der Informatik, wie Rekursion, Kapselung und Abstraktion, die durch die Muster veranschaulicht werden.

Aus den von Gamma et al. (1995) beschriebenen Entwurfsmustern wurden acht ausgewählt, wobei die Auswahl durch zwei wesentliche Kriterien geleitet wurde. Zum einen war entscheidend, ob die Muster zentrale informatische und objektorientierte Konzepte vermitteln, da diese den didaktischen Mehrwert der Software erheblich steigern. Diese Konzepte sollten den Kern der Lernsoftware bilden, während die spezifischen Anwendungsfälle der Entwurfsmuster eine eher untergeordnete Rolle spielen. Zum anderen war wichtig, dass jedes Muster anhand eines verständlichen und lebensnahen Beispiels veranschaulicht werden konnte. Diese Beispiele mussten sich nicht nur in das Szenario des Freizeitparks integrieren lassen, sondern auch für Schüler/innen leicht nachvollziehbar sein.

Die Module der ersten Version deckten folgende Entwurfsmuster ab (Tab. 1):

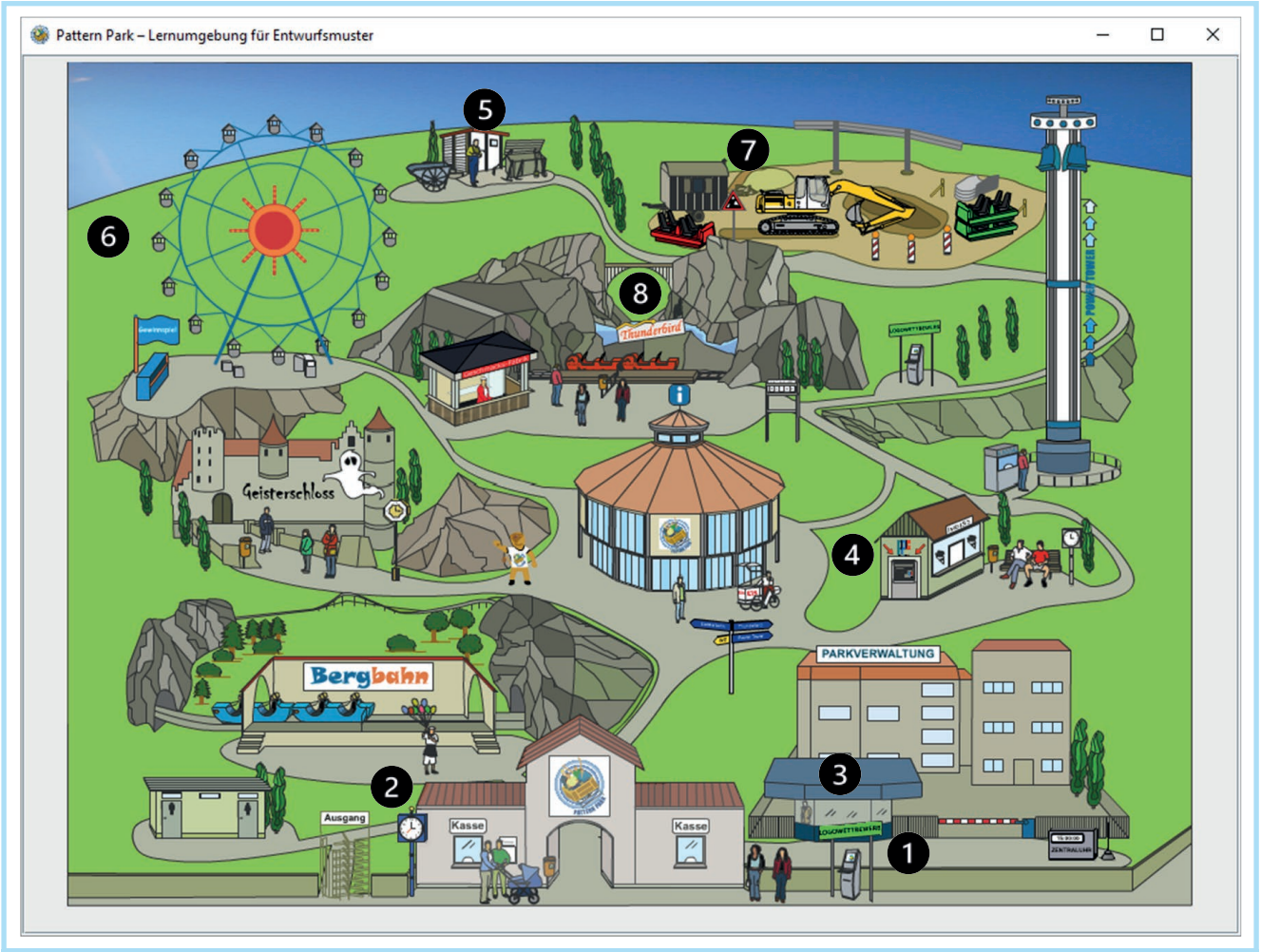

Abb.1. Die Übersichtskarte des Freizeitparks mit den Entwurfsmustern der ursprünglichen Version (Nummerierung siehe Tabelle 1)

Nr.	Entwurfsmuster	Beispiel	Konzepte
1	Kompositum	Grafik / Jugendgruppe	Rekursion, Datenstruktur Baum, Aggregation, Komposition
2	Beobachter	Eine Uhrzeit – mehrere Uhren	Assoziation (1-zu-n), Darstellung von Daten
3	Fassade	Pförtner	Datenkapselung, Schnittstelle, Delegation
4	Proxy	Geldkarte	Zugriffsschutz, Schnittstelle
5	Dekorierer	Werkzeuge und Arbeitskleidung von Mitarbeitern	Rekursion, Vererbung
6	Iterator	Durchlaufen einer Besucherliste	Datenstruktur Liste, Traversierung
7	Schablone	Aufbau von Bahnen	Vererbung
8	Zustand	Achterbahnfahrt	Zustandsdiagramm, Zustände, Bedingungen, Aktionen

Tab. 1. Übersicht über die Entwurfsmuster in der ersten Version von Pattern Park

- **Kompositum:** Behandlung von hierarchischen Strukturen, wie z. B. der Aufbau eines Logos aus Linien und Kreisen.
- **Beobachter:** Darstellung von Abhängigkeiten, z.B. zwischen einer zentralen Uhr und mehreren Parkuhren.
- **Fassade:** Erklärung der Schnittstellenreduktion, z.B. durch einen Pförtner im Verwaltungsbereich.
- **Proxy:** Visualisierung von Zugriffskontrollen, wie bei Geldkarten.
- **Dekorierer:** Dynamische Erweiterung von Funktionen, z.B. durch Arbeitskleidung.
- **Iterator:** Traversierung von Listen, wie Besucherlisten.
- **Schablonenmethode:** Wiederverwendung von Verhaltensmustern, am Beispiel Achterbahnen.
- **Zustand:** Zustandsübergänge, wie bei einer Achterbahnfahrt.

Jedes Modul besteht aus mehreren Lerneinheiten, darunter Animationen, Aufgaben ohne Diagramme der UML und interaktive UML-Puzzles, die den Einsatz der Muster in der Softwareentwicklung veranschaulichen.

4 Die Erweiterung von 2021

4.1 Übersicht über die neuen Entwurfsmuster

Im Jahr 2021 wurde Pattern Park von Studierenden der Hochschule Stralsund unter Leitung von Prof. Dr. Honekamp überarbeitet und um acht Entwurfsmuster erweitert (Tab. 2). Abbildung 3 zeigt, wo die neuen Muster im Pattern Park zu finden sind.

Besonders das MVC-Muster erweitert den Fokus der Software auf Architekturmuster und zeigt, wie komplexe Anwendungen strukturiert werden können. Es bietet Lernenden einen Einstieg in die Gestaltung moderner Softwareanwendungen. Da es in vielen Bundesländern in Lehrplänen genannt wird, wird es im Folgenden genauer für den Informatikunterricht analysiert.

Nr.	Entwurfsmuster	Beispiel	Konzepte
1	Model-View-Controller (MVC)	Turm mit Kontrolleur	Trennung von Datenhaltung, Darstellung und Steuerung
2	Singleton	Besucherzähler am Eingang	Sicherstellung der Existenz genau einer Instanz einer Klasse
3	Fabrik-Methode	Unterschiedliche Essenseinrichtungen (Geschmacks-Fabrik)	Flexible Erstellung von Objekten
4	Brücke	Unterschiedliche Pflanzenarten und Farben	Trennung von Abstraktion und Implementierung
5	Adapter	Neue Waggons werden mittels Kupplung mit vorhandenen verbunden	Anpassung inkompatibler Schnittstellen
6	Strategie	Verkaufsstrategien von Verkäufern	Austauschbare Algorithmen zur Lösung von Problemen
7	Memento	Zustände im Toilettenhaus	Erstellen von Zuständen eines Objekts, ohne dessen interne Struktur preiszugeben
8	Besucher	Besucher reagieren unterschiedlich auf unterschiedliche Attraktionen	Ermöglicht es, eine Operation auf einer Objektstruktur auszuführen, ohne die Klassen der Objekte zu verändern

Tab. 2. Übersicht über die ergänzten Entwurfsmuster in der aktuellen Version von Pattern Park

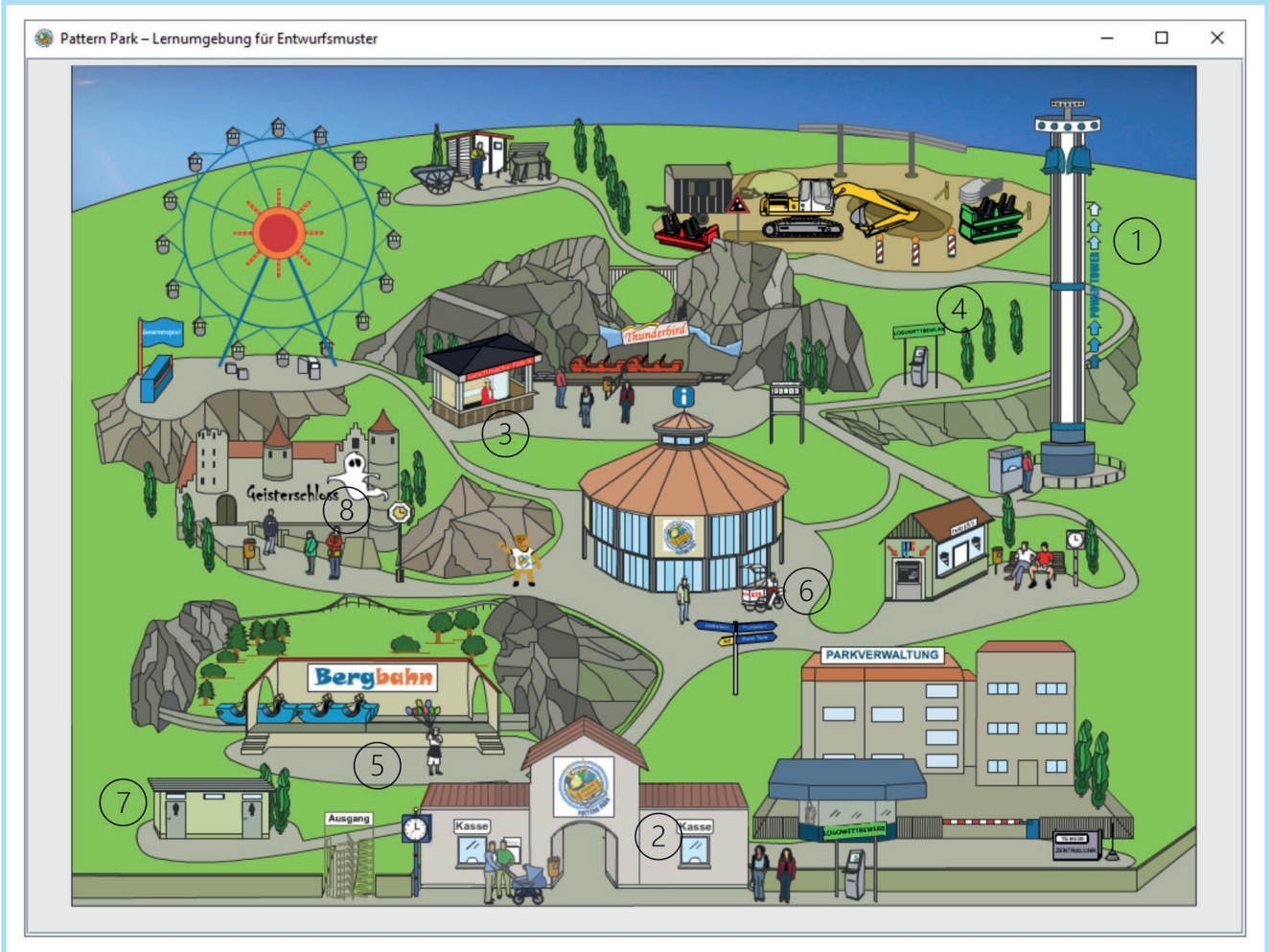

Abb. 3. Die Übersichtskarte des Freizeitparks mit den neuen Entwurfsmustern der erweiterten Version (Nummerierung siehe Tabelle 2)

4.2 Das MVC-Architekturmuster

Das Model-View-Controller-Architekturmuster wird von BUSCHMANN et al. (1996, 124) folgendermaßen beschrieben:
„Das Model-View-Controller-Muster (MVC) unterteilt eine interaktive Anwendung in drei Komponenten. Das Modell enthält die Kernfunktionalität und die Daten. Ansichten (engl. views) präsentieren dem Anwender Informationen. Steuerungskomponenten sind für Bedieneingaben verantwortlich. Ansichten und Steuerungskomponenten zusammen umfassen die Benutzerschnittstelle. Ein Mechanismus zur Benachrichtigung über Änderungen (engl. change-propagation mechanism) sichert die Konsistenz zwischen der Benutzerschnittstelle und dem Modell.“

Das Model-View-Controller (MVC)-Architekturmuster ist ein grundlegendes Designprinzip, das vor allem in interaktiven Anwendungen genutzt wird, um die Benutzeroberfläche von der zugrunde liegenden Logik zu trennen. Ziel des Musters ist es, die Flexibilität der Benutzeroberfläche zu erhöhen, indem sie unabhängig vom funktionalen Kern gehalten wird. Historisch entwickelte sich das MVC-Muster aus der zweischichtigen Architektur, in der bereits eine Trennung von Logik und Benutzeroberfläche vorgenommen wurde, um eine höhere Wiederverwendbarkeit von Oberflächenkomponenten zu erreichen. In den 1980er-Jahren wurde das Konzept durch die Programmiersprache Smalltalk bekannt und etabliert. Es wurde seitdem vielfach adaptiert und findet heute breite Anwendung in modernen Softwareentwicklungsprojekten.

Das MVC-Muster besteht aus drei zentralen Komponenten: dem Modell (Model), der Ansicht (View) und der Steuerung (Controller). Das **Modell** enthält die Anwendungsdaten und stellt Methoden bereit, um diese Daten zu bearbeiten. Es bildet den Kern des Systems, der unabhängig von der Darstellung und Interaktion bleibt. Beispielsweise kann das Modell als Datenbank oder als andere Form der Datenhaltung implementiert sein. Ansichten und Steuerungen greifen auf das Modell zu, ohne dessen innere Details zu beeinflussen. Die **Ansicht** ist für die Darstellung der im Modell gespeicherten Informationen verantwortlich. Sie kann in unterschiedlichen Formen existieren, um verschiedene Anforderungen der Benutzer zu erfüllen. So ist es möglich, dass mehrere Ansichten gleichzeitig auf ein Modell zugreifen, um die gleichen Daten auf unterschiedliche Weise zu präsentieren. Die **Steuerung** schließlich dient als Vermittler zwischen Benutzer und System. Sie verarbeitet Benutzereingaben, kontrolliert die Interaktion mit dem Modell und entscheidet, welche Änderungen an den Ansichten vorgenommen werden.

Ein grundlegendes Prinzip im MVC-Muster ist die Entkopplung der Komponenten. Durch die Trennung der Benutzeroberfläche vom funktionalen Kern wird das Geheimnisprinzip umgesetzt: Jede Komponente kennt nur die Details, die für ihre spezifische Aufgabe relevant sind. Dies erleichtert die Wiederverwendung und Modularisierung, da Elemente der Benutzeroberfläche, wie Schaltflächen, Listen oder Textfelder, in Bibliotheken zusammengefasst werden können. Dadurch lassen sie sich in verschiedenen Projekten mehrfach nutzen, ohne Anpassungen am Kernsystem vornehmen zu müssen.

Ein weiteres fundamentales Konzept im MVC ist die **EVA-Struktur** (Eingabe – Verarbeitung – Ausgabe). Die Steuerung übernimmt die Eingabe, indem sie Benutzeraktionen wie Klicks oder Tastendrücke verarbeitet und entsprechende Anfragen an das Modell weiterleitet. Die Verarbeitung der Daten erfolgt ausschließlich im Modell, das alle notwendigen Operationen durchführt. Die Ausgabe wird durch die Ansicht umgesetzt, die die aktuellen Daten des Modells präsentiert. Besonders bei Systemen mit mehreren Ansichten zeigt sich die Stärke dieser Trennung: Änderungen im Modell führen dazu, dass alle betroffenen Ansichten aktualisiert werden, was durch das Beobachter-Entwurfsmuster effizient umgesetzt werden kann.

Im Bereich der Benutzerinteraktion ist das MVC-Muster eng mit der Ereignisverarbeitung verbunden. In Programmiersprachen wie Java wird hierfür das Delegations-Ereignis-Modell verwendet. Benutzeraktionen wie Mausklicks oder Tastendrücke lösen Ereignisse aus, die von Ereignisquellen (z.B. grafische Elemente der Benutzeroberfläche) generiert und von Ereignisempfängern (z.B. Steuerungskomponenten) verarbeitet werden. Diese Beziehung zwischen Quelle und Empfänger spiegelt sich in der Verbindung zwischen Ansicht und Steuerung im MVC-Muster wider. Während im Delegations-Ereignis-Modell eine 1:n-Beziehung zwischen Quellen und Empfängern besteht, ist die Beziehung zwischen Ansicht und Steuerung im MVC typischerweise 1:1.

Darüber hinaus ermöglicht das MVC-Muster eine kritische Reflexion über die Darstellung von Daten. Verschiedene Ansichten können unterschiedliche Aspekte desselben Datenbestands hervorheben, was Lernenden ein Verständnis für die Vor- und Nachteile verschiedener Präsentationsformen vermittelt. Gleichzeitig eröffnet dies die Möglichkeit, zu diskutieren, wie Darstellungsformen die Wahrnehmung und Interpretation von Daten beeinflussen können.

Objektorientierte Prinzipien wie Vererbung und Assoziation spielen ebenfalls eine zentrale Rolle im MVC-Architekturmuster. Das Muster erlaubt es, gemeinsame Elemente in den Klassen „Ansicht“ und „Steuerung“ zu identifizieren und in einer Oberklasse zusammenzufassen. Diese Struktur fördert die Wiederverwendbarkeit und erleichtert die Wartung des Systems. Beispielsweise können Steuerungskomponenten durch Vererbung um zusätzliche Funktionalitäten erweitert werden, ohne die zugrunde liegende Struktur zu beeinträchtigen.

Zusammenfassend lässt sich sagen, dass das MVC-Muster eine effektive Trennung von Datenhaltung, Benutzerinteraktion und Darstellung ermöglicht. Diese Struktur bietet eine hohe Flexibilität und Wiederverwendbarkeit, da Änderungen an einer Komponente keine direkten Auswirkungen auf die anderen haben. Es ist besonders geeignet für Anwendungen, die komplexe Benutzerschnittstellen erfordern, und bildet die Grundlage für viele moderne Frameworks und Architekturen. Im Unterricht bietet das MVC-Muster eine hervorragende Gelegenheit, zentrale informatische Konzepte wie Modularisierung, Ereignisverarbeitung und objektorientiertes Design zu vermitteln und praxisnah anzuwenden.

Abbildung 4 zeigt ein Wirkungsdiagramm, in dem die Zusammenhänge zwischen den fundamentalen Ideen im MVC-Architekturmuster dargestellt werden (Ufer, 2007, 69).

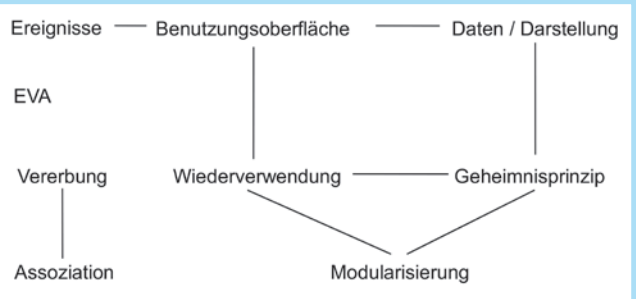

Abb. 4. Wirkungsdiagramm der fundamentalen Ideen im Model-View-Controller-Architekturmuster nach Ufer (2007, 69)

5 Aufbau der Module in Pattern Park am Beispiel des MVC-Architekturmusters

Jedes Modul in Pattern Park ist einem spezifischen Entwurfsmuster gewidmet. Der Aufbau sowie die Navigation innerhalb der Module sind durchgehend einheitlich gestaltet. Nach der Auswahl eines Moduls über die Freizeitparkkarte gelangt man zu den Inhalten des jeweiligen Entwurfsmusters.

Der Einstieg in ein Modul erfolgt über eine Übersichtsseite, die das Entwurfsmuster anhand eines Beispiels aus dem Freizeitpark kurz beschreibt. Diese Beschreibung umfasst eine prägnante Erläuterung der Musteridee sowie die Nennung ihrer Vor- und Nachteile. Ergänzend dient ein Bild sowohl zur Veranschaulichung der Funktionsweise als auch als visuelles Erkennungsmerkmal des Musters. Diese Seiten sollen den Benutzer/innen helfen, die grundlegenden Ideen der Entwurfsmuster auch zu einem späteren Zeitpunkt leicht wieder ins Gedächtnis zu rufen. Ein Klick auf das Symbol des Freizeitparks führt zurück zur Übersichtskarte, während über Reiter am oberen Bildschirmrand die weiteren Inhalte des Moduls zugänglich sind.

5.1 Einführungsanimationen und Übungen

Nach der Übersichtsseite folgt eine Einführung in das Muster durch einen Animationsfilm, der das Beispiel aus dem Freizeitpark anschaulich darstellt. Benutzer können zwischen einer Version mit Ton und einer Variante mit Untertiteln wählen, die über ein Informationssymbol zugänglich sind. Die Animationen

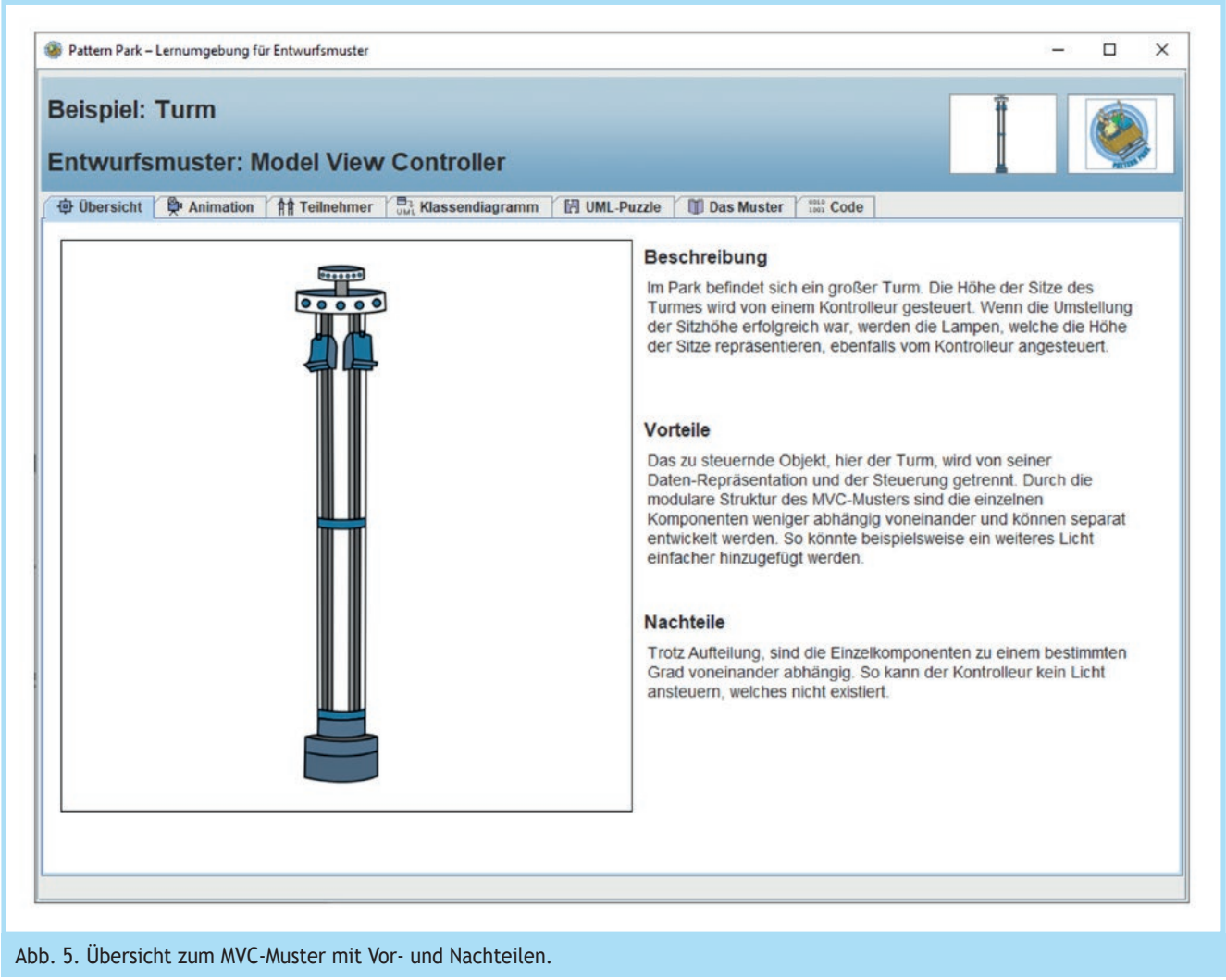

Abb. 5. Übersicht zum MVC-Muster mit Vor- und Nachteilen.

bieten einen verständlichen Einstieg in die Funktionsweise der Muster und sind ideal geeignet, um komplexe Zusammenhänge zu verdeutlichen.

Anschließend finden sich Übungsseiten, die gezielt auf das Muster abgestimmte Aufgaben enthalten. Jede Übung beginnt mit einer kurzen Beschreibung der Lernziele, Vorkenntnisse und der ungefähren Bearbeitungszeit. Zu den Übungen zählen Aufgaben ohne UML-Diagramme, bei denen die Schwerpunkte auf der Vermittlung der Musteridee und deren Anwendung liegen. Ergänzend dazu gibt es Übungen mit UML-Diagrammen, die das Muster aus objektorientierter Sicht mithilfe von Sequenz- oder Klassendiagrammen erklären. Für das Zustandsmuster wird abweichend ein Zustandsdiagramm zu einer Achterbahnfahrt behandelt.

5.2 UML-Puzzles

Ein besonderes Highlight jedes Moduls ist das UML-Puzzle, in dem Benutzer ein unvollständiges Klassendiagramm basierend auf dem Entwurfsmuster vervollständigen. Dazu können Klassen erstellt, Attribute und Methoden definiert sowie Beziehungen wie Assoziationen, Vererbungen, Kompositionen und Aggregationen hinzugefügt werden. Der Editor bietet Funktionen zum Rückgängigmachen von Arbeitsschritten sowie zur Speicherung und Wiederherstellung des aktuellen Diagrammzustands. Die Benutzer können ihre Ergebnisse überprüfen lassen, wobei das System Feedback zu möglichen Fehlern oder zur korrekten Lösung gibt.

5.3 Ergänzende Inhalte

Zu jedem Entwurfsmuster gibt es zudem eine allgemeine Beschreibung, die eine umgangssprachliche Definition, ein allgemeingültiges Klassendiagramm und den entsprechenden Java-Code (nur Methodenrümpfe und Attribute) umfasst. Diese Inhalte sind im Glossar verfügbar und bieten eine zusätzliche Referenz für die Vertiefung und Wiederholung der Konzepte.

Diese strukturierte Gestaltung der Module unterstützt Schüler/innen dabei, die grundlegenden Prinzipien der Entwurfsmuster zu verstehen und anzuwenden. Die Einheitlichkeit der Module gewährleistet eine intuitive Bedienbarkeit und ermöglicht einen effizienten Lernprozess.

6 Einbindung von Pattern Park in den Unterricht

Die Software eignet sich, um den Informatikunterricht praxisnah und interaktiv zu gestalten. Empfohlen sei hier eine Vor-

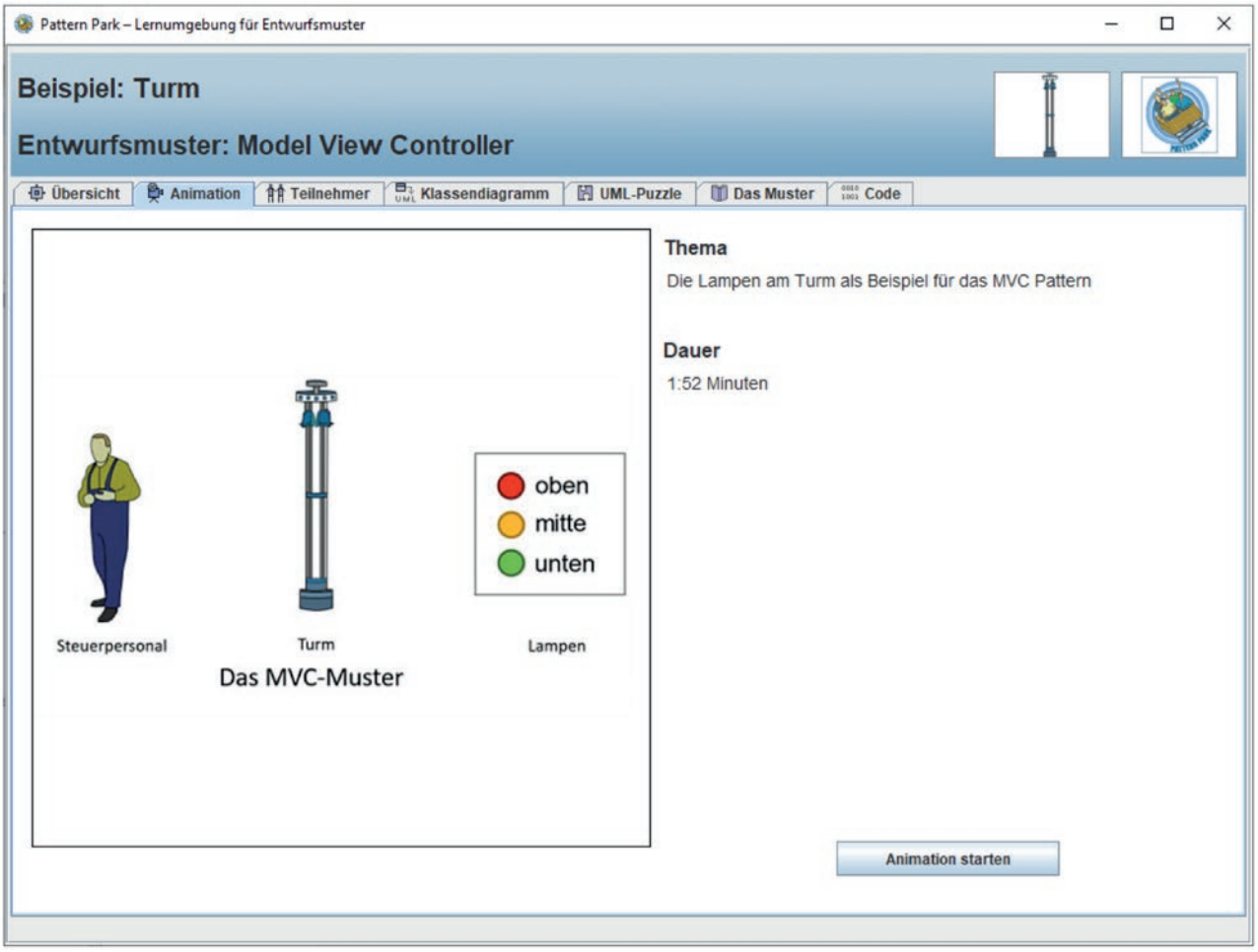

Abb. 6. Screenshot der Animationsseite des MVC-Musters

gehensweise gemäß der Semantic Wave Theory, nach der abstrakte Inhalte erst vorgestellt, dann auf ihren Kern reduziert und an einem anschaulichen Beispiel erarbeitet werden.

Danach erfolgt ein Transfer auf einen komplexeren Kontext (STECHERT, 2021). Ein Unterrichtsablauf kann wie folgt aussehen:

1. **Einleitung:** Theoretische Vermittlung eines Entwurfsmusters und seiner Anwendungen. Dies kann durch eine kurze Präsentation oder Diskussion erfolgen.

2. **Exploration mit der Software:** Schüler/innen bearbeiten eigenständig ein Modul, sehen sich die Animationen an, lösen Aufgaben und vervollständigen UML-Puzzles.

3. **Praktische Anwendung:** Im Rahmen des Use-Modify-Create-Ansatzes wenden die Lernenden das Muster in einer Programmieraufgabe an, modifizieren bestehenden Quellcode und entwickeln eigene Lösungen.

Dieser Ablauf fördert nicht nur das Verständnis, sondern auch die aktive Anwendung und Vertiefung der erlernten Inhalte in einem veränderten Kontext gemäß Semantic Wave Theory.

7 Ausblick: Grundlagen für Beruf und Studium schaffen

Der Schwerpunkt der Software liegt auf fundamentalen Konzepten der Informatik, wie Rekursion, Kapselung und Abstraktion, die einem Informatikunterricht in der gymnasialen Oberstufe gerecht werden.

Durch die intensive Arbeit mit Entwurfsmustern gewinnen Schüler/innen zudem ein vertieftes Verständnis für Softwaredesign und -architektur. Diese Kompetenzen sind essenziell, um komplexe Systeme zu analysieren und eigene Lösungen zu entwickeln.

Die in der Schule erworbenen Kenntnisse erleichtern im Sinne der Wissenschaftspropädeutik den Einstieg ins Informatikstudium und schaffen eine solide Basis für weiterführende Themen wie Software Engineering, objektorientiertes Design und moderne Architekturen.

Download und weitere Informationen

Die Lernsoftware Pattern Park ist als Open-Source-Projekt auf GitLab verfügbar:
https://gitlab.com/patternpark/pattern-park-app/-/releases.

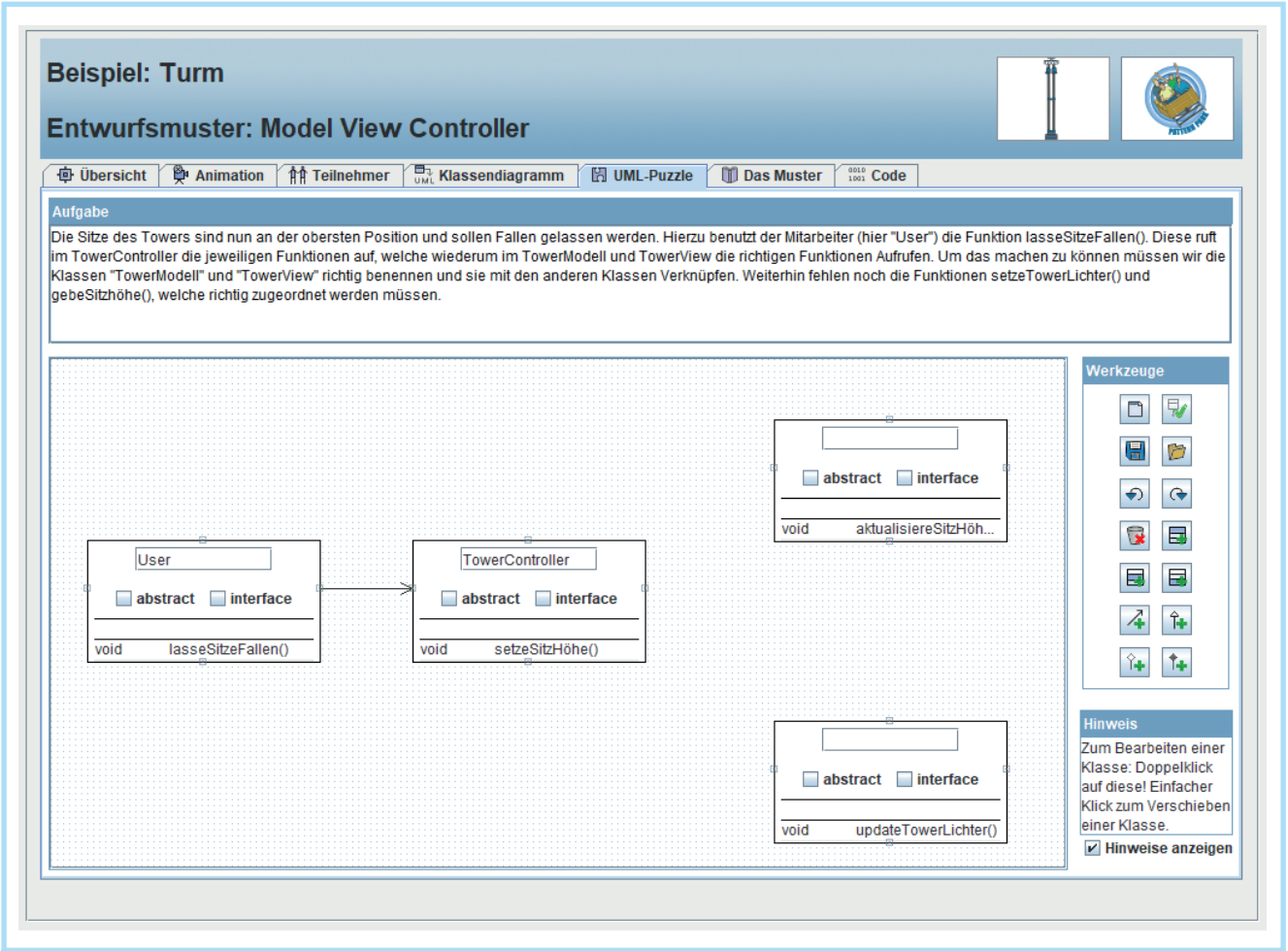

Abb. 7. Screenshot des UML-Puzzles zum MVC-Muster

Literatur

Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P. & Stal, M. (1998). *Pattern-orientierte Software-Architektur.* Addison-Wesley. München

Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. (1995). *Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA

ISB (2025a) Staatsinstitut für Schulqualität und Bildungsforschung München. https://www.lehrplanplus.bayern.de/jahrgangsstufenprofil/gymnasium/12/informatik

ISB (2025b). Staatsinstitut für Schulqualität und Bildungsforschung München. https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/informatik/grundlegend

ISB (2025c) Staatsinstitut für Schulqualität und Bildungsforschung München. https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/informatik/erhoeht

MBWK (2021). Ministerium für Bildung, Wissenschaft und Kultur des Landes Schleswig-Holstein. *Fachanforderungen Informatik. Allgemein bildende Schulen. Sekundarstufe I. Sekundarstufe II.* https://fachportal.lernnetz.de/sh/faecher/informatik/fachanforderungen.html

Schwill, A. (1993). Andreas Schwill: *Fundamentale Ideen der Informatik.* In: Zentralblatt für Didaktik der Mathematik 1.

Stechert, P. (2009). *Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht.* Dissertationsschrift. Fachbereich Elektrotechnik und Informatik, Universität Siegen, 256 Seiten.

Stechert, P. (2021). Semantic Waves als Unterrichtskonzept / Informatikdidaktik kurz gefasst Teil 28. https://www.youtube.com/watch?v=Io2KucSU6so

Ufer, J. (2007). *Architekturmuster als Beitrag zum Informatiksystemverständis.* Diplomarbeit, Siegen, 2007

Dr. Peer Stechert, peer.stechert@mnu.de, arbeitet als Landesfachberater und Studienleiter für Informatik am IQSH in Schleswig-Holstein und ist zuständig für die Aus-, Fort- und Weiterbildung im Fach Informatik.